UNIVERSIDAD DE BUENOS AIRES
Facultad de Ciencias Exactas y Naturales
Departamento de Matemática

Teoría de modelos, teoría de prueba, y aspectos computacionales de lógicas para razonar sobre árboles con datos

Tesis a presentar para optar al título de Doctor de la Universidad de Buenos Aires en el área Ciencias Matemáticas

Sergio Alejandro Abriola

Director de tesis: Santiago Figueira
Consejero de estudios: Román Sasyk

Lugar de trabajo: Instituto de Investigación en Ciencias de la Computación (ICC)

Contents

Teoría de modelos, teoría de prueba, y aspectos computacionales de lógicas para razonar sobre árboles con datos

Resumen. XPath es el lenguaje de consultas más ampliamente utilizado para documentos XML; es un estándar abierto y constituye una World Wide Web Consortium (W3C) Recommendation. Trabajamos con un fragmento de este lenguaje, apropiadamente reinterpretado como una lógica: $\mathrm{XPath}_{=}$, una lógica que puede ser vista como una extensión de la lógica modal básica, pero en el contexto de árboles con datos y, fundamentalmente, con la capacidad de realizar comparación de datos entre nodos.

Desarrollamos la teoría de modelos de $\mathrm{XPath}_{=}(\downarrow)$, que solo puede navegar el árbol descendiendo, y $\mathrm{XPath}_{=}(\uparrow\downarrow)$, que también puede navegar hacia arriba. Obtenemos resultados de definibilidad y separación para los dos tipos de fórmulas en $\mathrm{XPath}_{=}$: expresiones de nodo y expresiones de camino. También desarrollamos la noción de bisimulación binaria para ambos fragmentos, y demostramos un teorema de caracterización al estilo van Benthem para expresiones de camino de $\mathrm{XPath}_{=}(\downarrow)$.

Encontramos axiomatizaciones ecuacionales correctas y completas para $\mathrm{XPath}_{=}(\downarrow)$ y para su fragmento libre de desigualdades de datos $\mathrm{XPath}_{=}(\downarrow)^{-}$. Para demostrar completitud construimos, para cada expresión de nodo consistente, un árbol finito con datos en cuya raíz se satisface la formula.

Extendemos $\mathrm{XPath}_{=}$ al universo de grafos con datos, y analizamos la complejidad computacional de decidir si dos nodos en dos grafos con datos son bisimilares. Calculamos cotas ajustadas de complejidad para varios problemas de bisimilaridad y para diferentes universos de modelos.

Introducimos LRV, una lógica para navegar sobre árboles con datos múltiples, y obtenemos procedimientos de decisión para el problema de satisfabilidad de LRV y algunos fragmentos al reducirlo al problema de control-state-reachability de diferentes sistemas con contadores.

Palabras clave: $\mathrm{XPath}_{=}$, data-aware logics, bisimulation, model theory, expressivity, proof theory, computational complexity, automata, satisfiability, branching counter systems.

Model theory, proof theory, and computational aspects of logics for reasoning on data trees

Summary. XPath is the most widely used query language for XML documents; it is an open standard and constitutes a World Wide Web Consortium (W3C) Recommendation. We work with a fragment of this language, suitably reinterpreted as a logic: $\mathrm{XPath}_{=}$, which can be seen as an extension of basic modal logic, but in the context of data trees and, fundamentally, with the capacity to deal with data comparisons between nodes.

We develop the model theory of both $\mathrm{XPath}_{=}(\downarrow)$, which can only navigate the tree downwards, and $\mathrm{XPath}_{=}(\uparrow\downarrow)$, which can also navigate upwards. We obtain definability and separation results for the two types of formulas in $\mathrm{XPath}_{=}$: node expressions and path expressions. We also develop the notion of binary bisimulation for both fragments, and prove a van Benthem-style characterization theorem for paths expressions of $\mathrm{XPath}_{=}(\downarrow)$.

Sound and complete equational axiomatizations are found for $\mathrm{XPath}_{=}(\downarrow)$ and for its data-inequality-free fragment $\mathrm{XPath}_{=}(\downarrow)^{-}$. To prove completeness we construct, for every consistent node expression, a finite data tree where it is satisfied at the root.

$\mathrm{XPath}_{=}$ is extended to the universe of data graphs, and we analyze the computational complexity of deciding if two pointed data graphs are bisimilar. We calculate tight complexity bounds for various bisimilarity problems and different universes of models

We introduce LRV, a logic to navigate over multidata trees, and obtain decision procedures for the satisfiability problem of LRV and some fragments by reducing it to the control-state reachability problem of different counter systems.

Keywords: $\mathrm{XPath}_{=}$, data-aware logics, bisimulation, model theory, expressivity, proof theory, computational complexity, automata, satisfiability, branching counter systems.

Introduction

> Aha, that's a wallpaper word, thought William. When people say *clearly* something, that means there's a huge crack in their argument and they know things aren't clear at all.
>
> *The Truth*
> Terry Pratchett

What is a logic? In a sense, a logic is a way to reason over a certain domain of discourse, a language of symbols and rules that allows us to talk about truth and consequence, about what *can be* and about what *must be* if we start with certain premises. More formally, a logic can be thought of as composed of two interconnected parts: the syntactic and the semantic. At its basic, the *syntactic* part establishes a set of usable symbols and a set of formulas constructed with those symbols, and it might include deduction rules that connect formulas and dictate what follows from what. But, by themselves, these symbols and formulas lack any explicit meaning: this is given by the *semantics* of the logic, the interpretation of its formulas in a certain framework. Given an interpretation for the symbols and formulas of the logic, a formula acquires meaning, becoming a statement about a particular object or model, and as such it may be either true or false. Now, there are many ways to judge the adequacy or usefulness of a logic. One could for example try to gauge its *expressiveness*, the extent up to which we can express various properties (which we can define in informal or meta-logical terms) by only using formulas from our logic. While more expressiveness is, all other things being equal, better, it usually comes at a price; in more expressive logics it might be harder to determine some characteristics of formulas, such as whether two formulas are semantically equivalent, whether a formula is true on a particular model, whether a formula can be satisfied in some model, whether a formula can be deduced from a set of axioms, et cetera. Indeed, other dimensions from which to judge a logic are those related with the *decidability* and with the *algorithmic complexity* of such problems: are those problems solvable by effective means at all? if so, then how hard is it for an algorithm to answer them? On the other hand, making a logic more expressive might have other associated costs, such as the sacrifice of readability or notational succinctness. These are the kind of trade-offs that explain the existing abundance of logics, and justify

then why we might use some logic even when there are more expressive ones available. Such trade-offs will be visible when we analyze our main logics of the thesis: various fragments of $\text{XPath}_=$ and other data logics.

A database query language is a computer language that is used to answer questions about a database, such as whether an entry has a particular property. Query languages can be formally studied as logics, by making adequate mathematical abstractions of the data structures on which they are used and by identifying *queries* with corresponding *formulas*. When this is done, we get an equivalence between the answers to queries on databases and the semantics of their translation into formulas of the logic (on the mathematical models corresponding to those databases). XPath (XML Path Language) is a query language designed to work over XML documents, and Data XPath, here called $\text{XPath}_=$, is the corresponding logic designed to reason over data trees, an abstraction of XML documents consisting of a rooted tree where each node comes with a single label and a single data value. $\text{XPath}_=$ is close in many aspects to basic modal logic (BML): it has navigational modalities (such as $\uparrow$ or $\downarrow$) and it is local (formulas are analyzed on particular nodes of the trees, and, depending on the fragment of the logic, the depth up to which they can 'see' is bounded by the size of the formula). $\text{XPath}_=$ has two sorts of formulas: *node expressions*, which, like BML formulas, are analyzed on nodes, but also *path expressions*, which are analyzed on pairs of nodes. Unlike BML, where nodes do not even have data values, $\text{XPath}_=$ is capable of making *data comparisons* between two nodes, that is, it can check whether two nodes have the same data value or not, but it cannot query for the particular data value of a node. Furthermore, as we will see, this difference in capabilities is fundamental, and BML cannot properly express the concept of data comparison even when its nodes are enriched with data values. While the theory of BML is well-developed, such is not the case for $\text{XPath}_=$. In this thesis we made progress on the following theoretical aspects of $\text{XPath}_=$:

Model-theoretical. We studied problems such as *definability*, which asks if a class of models can be characterized by $\text{XPath}_=$ formulas; and *separation*, which asks if two classes can be separated by formulas. Our work on definability and separation for $\text{XPath}_=$ is inspired in the corresponding results for basic modal logic; in order to adjust the proofs for our framework, we devised appropriate notions of saturation and ultraproduct, and proved various technical results before arriving to the desired theorems. We also studied problems related to bisimulation, a central concept that approximates logical equivalence with a more structural notion. We created adequate notions of bisimulations for path expressions, and derived various key results for this framework. We also observed that results of $\text{XPath}_=$ over data trees can be extended to the wider context of data graphs.

Proof-theoretical. We obtained sound and complete *axiomatizations* for two fragments of $\text{XPath}_=$, that is, axioms such that all provable equivalences are universally true over data trees, and such that all equivalences which are universally true can be proved from those axioms. Doing this, we extended previous work done in the simpler case of XPath

without data comparisons. We proved the completeness of our axiomatizations through normal form theorems, and by making rather intricate constructions that show that all consistent node expressions in normal form are satisfiable.

Computational. We expanded our study from the universe of data trees to the more general case of $\text{XPath}_=$ over *data graphs*, which are a widely used abstraction of graph databases. In this area, we extended results of $\text{XPath}_=$ that were originally stated over data trees, and studied the problems of similarity and bisimilarity between data graphs from a computational perspective. We tightly classified these problems into various complexity classes, depending on the (bi)simulation notions selected and on the restrictions we apply to the class of models.

Additionally, we continued our study of data-aware logics to reason about data trees, expanding into *logics of repeating values* (LRV) over multidata trees. We started by generalizing previous work made over multidata words, proving interreducibility between the satisfiability problem of disjoint LRV over ranked trees and the coverability problem for branching vector addition systems with states (VASS). Then we presented an extension of BVASS called merging VASS (MVASS), and we proved that the satisfiability problem of LRV on ranked data trees is reducible to the control-state reachability problem of MVASS.

I.1 Context and preliminaries

In this section we introduce various concepts that are fundamental for this thesis. We start with a general overview of logics, where we present propositional logic and basic modal logic. While describing these logics we introduce various problems of relevance for this thesis, such as characterization, satisfiability, definability, and separation. We then advance to the main focus of study of this thesis, giving the necessary background and definitions for $\text{XPath}_=$, showing tools and terminology that are ubiquitously used in the rest of this thesis. After the basics of $\text{XPath}_=$ are explained, we state important properties and theorems that form the basis of some of our own results.

I.1.1 Logics

The fundamental syntactic component of a logic is made up of its symbols and the formulas that can be constructed with them. While formulas are inherently meaningless, the *semantics* of a logic indicate how these formulas are to be interpreted in concrete models, where it does makes sense to ask if a certain formula is *true* or *false*. **Model theory** is the area of study that deals with the semantic aspects of a logic, with the connection between structures and formulas. A classic problem of model theory is that of **definability**: determining conditions for a class of models to be definable by a formula or a set of formulas. Closely related is the problem of **separation**: that of determining conditions for two disjoint classes of models, such that there must be a formula (or set of formulas) which defines a third class of models that contains the first one but is disjoint with the

second. An important decision problem related to this area is the **satisfiability problem**: deciding whether a formula of a logic has a model where it is true.

Beyond symbols and formulas, the **syntactics** of a logic can include *deduction rules*, which connect formulas and establish what formulas can be deduced from what formulas. **Proof theory** deals with problems related with the deduction rules and their interaction with the semantics of a logic. A key problem in this area is that of finding **axiomatizations** for a logic: given the formulas and semantics of a logic, finding a deductive system such that it only deduces formulas that are universally true in all models, and such that all universal truths can be deduced in this system.

In the remaining of this subsection we give a very short overview of two commonly used logics: propositional logic and basic modal logic. We present propositional logic as an introduction to basic modal logic, which is itself presented for its many parallels with XPath. Among the theoretical results for these logics we include results for some of the aforementioned problems, and we briefly indicate some interesting computational aspects of these logics.

Propositional logic

Propositional logic is a very simple logic that can deal with propositional variables and a limited repertoire of formulas constructed with them. It can express statements of the form:

'It is not the case that p_1' (1)

'It is the case that either p_1 holds or p_2 and p_3 both hold' (2)

'If it is the case that p_1, then it is the case that p_2' (3)

That is, a formula is formed with propositional variables (in our examples, p_1, p_2, and p_3), Boolean conectives (*not*, *or*, and *and*), and implication (*then*). Formally, we have propositional variables $p_1, p_2, \ldots$ from a countably infinite set P, and logic symbols[1] $\neg$ and $\rightarrow$. A **formula** can then defined constructively as either a propositional variable, the negation of a single formula, or the composition of two formulas via $\rightarrow$. We notate this schema with the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid (\varphi \rightarrow \varphi)$$

where p can be any of the propositional variables.

So far the formulas are just strings made up in a certain way out of our given symbols. We still have to provide these of meaning; we have to define their semantics. While in more complex logics we speak of models, in the simple propositional logic this role is held by **valuations**, which assign either *true* or *false* to each propositional variable, that is: a valuation is a function $V : P \rightarrow \{0, 1\}$. The truth value of a formula given a valuation v,

[1]The symbols $\vee$ and $\wedge$ are absent, as they are not necessary; when the semantics is given, it can be seen that they can be simulated using only $\neg$ and $\rightarrow$.

is then given as would be expected: we say that v *satisfies* p, notated $v \models p$, iff $v(p) = 1$, we say that $v \models \neg\varphi$ iff $v \not\models \varphi$, and $v \models (\varphi \to \psi)$ iff $v \models \psi$ or $v \not\models \varphi$. It can be seen that $\vee$ and $\wedge$, with their expected semantics, can be constructed by only using $\to$ and $\neg$: $\varphi \vee \psi$ is thus a meta-syntactical shorthand for $(\neg\varphi \to \psi)$ and $\varphi \wedge \psi$ stands for $\neg(\varphi \to \neg\psi)$.

The following propositional formulas express the aforementioned statements:

$$\neg p_1 \qquad \text{(Expresses (1))}$$
$$p_1 \vee (p_2 \wedge p_3) \qquad \text{(Expresses (2))}$$
$$p_1 \to p_2 \qquad \text{(Expresses (3))}$$

While propositional logic is quite simple, there are already some interesting problems to study. As an example, we can ask if there is a decision procedure such that, given a propositional formula φ, determines whether or not there exists a valuation v such that $v \models \varphi$. This is the **satisfiability problem** of propositional logic, and it is quite easier than in other logics to see that such a procedure exists: it is enough to check whether the formula is satisfied over valuations with a domain restricted to the variables that appear in the formula. Regarding the **algorithmic complexity**, the measure of how hard the problem is, this satisfiability problem is called the Boolean Satisfiability Problem, and it is proven to be in NP-COMPLETE, i.e. in the category of decision problems whose positive answers can be checked in polynomial time.

Other problem of propositional logic is the **validity problem**: can we decide, for any formula φ, whether or not it holds true for every possible valuation?[2] We actually already know the answer to this problem, as it is the dual of the satisfiability problem: a formula φ is valid if and only if $\neg\varphi$ is not satisfiable.

We still can mention another of the syntactic component of logics: deduction rules. Propositional logic has a set of (somewhat abstruse) axiom schemas:

- $\varphi \to (\psi \to \varphi)$,
- $(\varphi \to (\psi \to \rho)) \to ((\varphi \to \psi) \to (\varphi \to \rho)$,
- $(\neg\psi \to \neg\varphi) \to ((\neg\psi \to \varphi) \to \psi)$,

and the single rule of inference *modus ponens*, which indicates that, for any formulas φ and ψ, we have that ψ is a consequence of the pair $\varphi \to \psi$ and φ. A formula φ is said to be *provable* from a set of formulas Γ, if there is finite derivation, using only the axioms, the formulas in Γ, and the application of *modus ponens*, that ends in φ. These axiom schema (with the inference rule *modus ponens*) constitute a sound and complete **axiomatization** of propositional logic: soundness means that all formulas that are provable from these axioms are true in any valuation, and completeness means that all formulas which are true in all valuations can be proved from these axioms.

[2] Such formulas are called a *tautologies*.

We now move on to basic modal logic, a generalization of propositional logic in which there are potentially many connected *worlds*, each with their own valuation for the propositional variables.

Basic modal logic

The main features of modal logics are their relational structure and the fact that they are mostly 'internal' or 'local' in the way they can view the structures, unlike first-order logic, in which formulas have a bird's-eye view and can 'see' all the universe such as with formulas of the type $\forall x \ldots$ or $\exists x \ldots$.

Basic modal logic can be thought of as a generalization of propositional logic. The framework of semantic discourse for modal logics are frames plus valuations. A **frame** F of BML is simply a (directed) graph: it consists of a set of elements U, called *worlds*, and a binary relation R over the domain U. We often denote $(w, w') \in R$ as $w \to w'$, and we say that such w' is a *neighbor* of w (although R is not necessarily symmetric) or a *child* of w (specially when F is a tree). A **valuation** over a frame with universe U is a function $V : U \times P \to \{0, 1\}$ that assigns, for each world, a truth value for each propositional variable. On each world, the logic can evaluate all the same things as propositional logic, but it also has the 'diamond' navigational operator $\Diamond$, which makes it possible to 'explore' the topological structure of the frame, and gives rise to quite complex queries that combine the propositional and the navigational parts.

Basic modal logic can express properties such as:

'If p_1 does not hold in this world, then p_2 holds in this world' (4)

'This world has a neighbor where p_1 and p_2 hold' (5)

'This world has a neighbor such that it has a neighbor where p_1 holds' (6)

'All neighbors of this world have a neighbor where p_1 does not hold' (7)

Formulas of BML are defined by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid (\varphi \to \varphi) \mid \Diamond\varphi,$$

where p represents a propositional variable. As in the propositional case, $\vee$ and $\wedge$ are not necessary.

Given a frame F and a valuation V, we call **model** (or **Kripke model**) to the pair F, V; basically, a Kripke model can be seen as a labeled graph, where each node has as labels those propositional variables that are true in it. For the semantics of BML, formulas are evaluated over a model and a world of the respective frame. The semantics is as in propositional logic, but we also need to assign meaning to formulas of the type $\Diamond\varphi$, which intuitively will be interpreted as 'there is a neighbor of the current node such that φ holds in it'. Formally, given a model $\mathcal{M} = \{F, V\}$ and a world w in F, we say that $\mathcal{M}, w \models \Diamond\varphi$ iff there is a world w' such that wRw' and $\mathcal{M}, w' \models \varphi$. It is customary to use the 'box' symbol $\Box$ in BML-formulas, where $\Box\varphi$ is a shorthand for $\neg\Diamond\neg\varphi$; this means that the semantics of $\Box\varphi$ over a world w is that φ holds in all neighbours of w (if any).

The aforementioned properties can be written as BML-formulas in the following way:

$$(\neg p_1 \to p_2) \qquad \text{(Expresses (4))}$$
$$\Diamond(p_1 \wedge p_2) \qquad \text{(Expresses (5))}$$
$$\Diamond\Diamond p_1 \qquad \text{(Expresses (6))}$$
$$\Box\Diamond\neg p_1 \qquad \text{(Expresses (7))}$$

See Figure 1 for an example of the semantics of the formulas above on a particular model; we write only the propositional variables that are positive on each node.

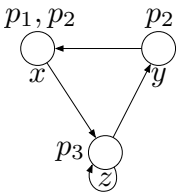


Figure 1: A graphical representation of a model $\mathcal{M} = \{F, V\}$. The frame F is constituted of three elements x, y, z, a relation $R = \{(y, x), (x, z), (z, z), (z, y)\}$. The valuation V assigns value 1 to a propositional variable on a world iff that propositional variable appears next to the node in the figure. Here property (4) is satisfied in worlds x and y, property (5) in y, (6) in z, and (7) in x, y.

An important concept for BML, and for logics in general, is that of logical **equivalence** between nodes. Given two models $\mathcal{M}$ and $\mathcal{M}'$, with worlds w and w' in the respective frames M and M', we say that $\mathcal{M}, w$ is **equivalent** to $\mathcal{M}', w'$ (notated $\mathcal{M}, w \equiv \mathcal{M}', w$) if and only if, for every BML-formula φ, φ holds on w iff it holds on w'. More precisely stated:

$$\mathcal{M}, w \equiv \mathcal{M}', w' \overset{\text{def}}{\Leftrightarrow} \text{ for all (BML) formulas } \varphi,\ \mathcal{M}, w \models \varphi \text{ iff } \mathcal{M}', w' \models \varphi.$$

That is, the logic cannot internally distinguish between those two worlds, although they might be distinguishable by an external observer.

A notion that is closely related to equivalence is that of **bisimilarity**. For the purpose of succinctness, let us call **pointed model** to a pair $\mathcal{M}, w$, where $\mathcal{M}$ is model and w is a world of the frame of $\mathcal{M}$. Given two models $\mathcal{M}$ and $\mathcal{M}'$ with respective frames M, M', we say that $Z \subseteq M \times M'$ is a **bisimulation** if for all $(x, x') \in M \times M'$ such that $(x, x') \in Z$ the following conditions hold:

- (**Harmony**) x and x' satisfy the same propositional variables.
- (**Zig**) If $x \to y$, then there exists $y' \in M'$ such that $x' \to y'$ and $(y, y') \in Z$.
- (**Zag**) If $x' \to y'$, then there exists $y \in M$ such that $x \to y$ and $(y, y') \in Z$.

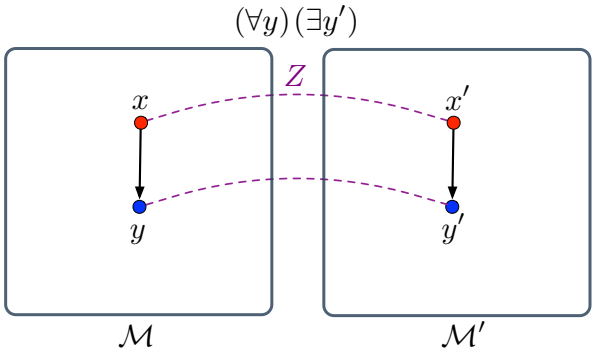


Figure 2: The process of checking the condition **Zig** of BML-bisimulation for $(x, x') \in Z$ entails checking that all y that are neighbors of x have a corresponding y' that is a neighbor of x'.

In Figure 2 we show an example of the process of checking **Zig** for a pair of nodes in a bisimulation.

Now, given two pointed models $\mathcal{M}, x$ and $\mathcal{M}', x'$, with respective frames M, M', we define when x and x' are said to be **bisimilar**, notated $\mathcal{M}, x \leftrightarrows \mathcal{M}', x'$:

$$\mathcal{M}, x \leftrightarrows \mathcal{M}', x' \overset{\text{def}}{\Leftrightarrow} \text{ there exists a bisimulation } Z \text{ such that } (x, x') \in Z \ .$$

For example, if x and x' satisfy the same propositional variables and they have no neighbors, then they are bisimilar via the bisimulation $Z = \{(x, x')\}$. If they have neighbors, a bisimulation Z that contains (x, x') will also contain other pairs, which must also satisfy the clauses **Harmony**, **Zig**, and **Zag**.

In a sense, bisimilarity indicates that basic BML operations done in any of the two models can be mirrored in the other one. Since it can be seen that BML-bisimulations are closed under union, it follows that there always exists a single *maximal* bisimulation between two models; an example of a maximal BML-bisimulation is shown in Figure 3.

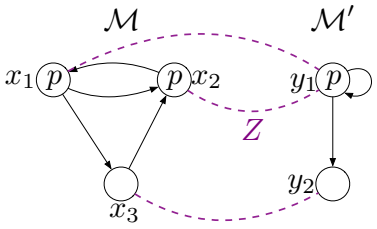


Figure 3: $Z = \{(x_1, y_1), (x_2, y_1), (x_3, y_2)\} \subseteq M \times M'$ is a bisimulation, as it can be verified that **Harmony**, **Zig**, and **Zag** all hold for the three pairs in Z. Also note that this is a (the) maximal bisimulation between $\mathcal{M}$ and $\mathcal{M}'$, simply because adding any other pair would fail **Harmony**.

The one-direction version of bisimilarity is called **similarity**, and it is notated $\mathcal{M}, x \rightrightarrows \mathcal{M}', x'$. The definition is as that of bisimilarity, but **simulation** does not include the **Zag**

condition; here, operations done in $\mathcal{M}$ can be replicated in $\mathcal{M}'$, but not necessarily the other way around. A basic example of two pointed models such that $\mathcal{M}, x \rightrightarrows \mathcal{M}', x'$ but not $\mathcal{M}, x \leftrightarrows \mathcal{M}', x'$ can be seen in Figure 4.

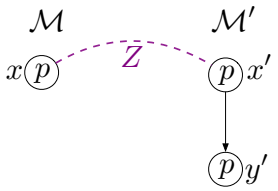


Figure 4: $Z = \{(x, x')\} \subseteq M \times M'$ is a simulation, since **Harmony** holds and **Zig** holds trivially. But there is no possible Z such that $(x, x') \in Z$ and **Zag** holds.

It is also important to remark that having both $\mathcal{M}, x \rightrightarrows \mathcal{M}', x'$ and $\mathcal{M}', x' \rightrightarrows \mathcal{M}, x$ does not imply that $\mathcal{M}, x \leftrightarrows \mathcal{M}', x'$: Figure 5 shows a counterexample.

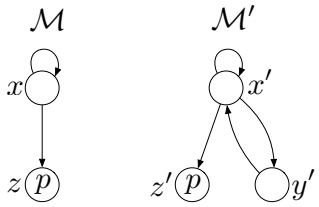


Figure 5: Similarity in both directions does not imply bisimilarity: it is trivial to see that $\mathcal{M}, x \rightrightarrows \mathcal{M}', x'$ holds, and it can be seen that $\mathcal{M}', x' \rightrightarrows \mathcal{M}, x$ via $Z = \{(x', x), (z', z), (y', x))\}$; however, $\mathcal{M}, x \not\leftrightarrows \mathcal{M}', x'$, as any bisimulation Z including (x, x') would need to relate y' to some node w in M, and that would violate **Harmony** in the case of $w = z$ and **Zig** in the case of $w = x$ (since from x we can move to z, which satisfies p unlike x', the only possible neighbor of y', and as such $(z, x') \notin Z$).

Bisimilarity between two pointed models is quite a strong property: all bisimilar pointed models are also equivalent, but there are pointed models that are equivalent but not bisimilar (see Figure 6 for a typical counterexample taken from [15]). However, Hennessy-Milner's theorem (or **Hennessy-Milner's characterization** theorem) establishes that the two concepts coincide over finitely branching frames[3] , that is, that nodes u and u' of finitely branching frames are bisimilar if and only if they cannot be distinguished by BML-formulas (see, e.g., [16]). Now, bisimilarity between two finite (and thus finitely branching) pointed models can be easily determined in PTIME (see [8] for a more general framework). In particular, we can efficiently calculate the maximal (auto)bisimulation between a finite model and itself, and obtain (because of the Hennessy-Milner's theorem) a division of its nodes

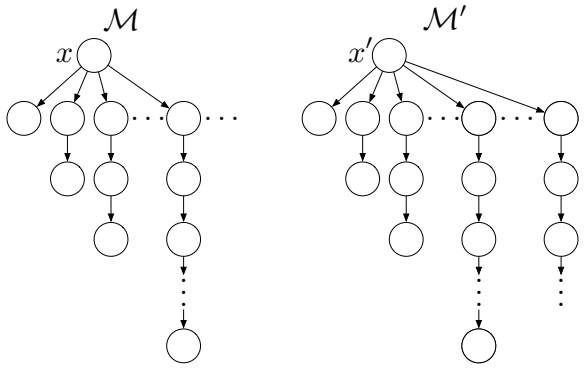


Figure 6: Bisimilarity and equivalence for BML do not coincide over arbitrarily branching frames. The represented pointed models are equivalent, as any BML-formula has a finite depth and thus can be satisfied in the finite branches of $\mathcal{M}$. But these pointed models are not bisimilar: suppose there is a bisimulation Z with $(x, x') \in Z$. Then necessarily, if we take y' as the first node of the infinite branch of M', from **Zag** there is some y with $x \to y$ such that $(y, y') \in Z$. Now, iteratively applying **Zag**, we eventually arrive to the ending node of a branch in M, and **Zag** does not hold between this node and the corresponding node of $\mathcal{M}'$.

into equivalence classes of logical indistinguishability.

Beyond the connection between bisimilarity and equivalence, another important result for basic modal logic and bisimulations is the one that characterizes basic modal logic as a fragment of first-order logic. Before giving this theorem, we present the standard translation of BML into first-order logic. We fix the signature σ with a binary relation $\rightsquigarrow$, and an infinite number of unary predicates P_i (one for each propositional variable). Any Kripke model $\mathcal{M} = \{(U, R), V\}$ can be seen as a first-order σ-structure with universe U, where the symbols of the signature have the following semantics:

$$\begin{aligned} \rightsquigarrow^{\mathcal{M}} &= \{(x, y) \in U^2 \mid xRy\}; \\ P_i^{\mathcal{M}} &= \{x \in U \mid V(x, p_i) = 1\}. \end{aligned}$$

We now present a truth-preserving translation Tr_x mapping BML-formulas into first-order σ-formulas with one free variable x:

$$\begin{aligned} \mathrm{Tr}_x(p_i) &= P_i(x) && (p_i \in P \text{ a propositional variable}) \\ \mathrm{Tr}_x(\neg\varphi) &= \neg\mathrm{Tr}_x(\varphi) \\ \mathrm{Tr}_x(\varphi \to \psi) &= \mathrm{Tr}_x(\varphi) \to \mathrm{Tr}_x(\psi) \\ \mathrm{Tr}_x(\Diamond\varphi) &= \exists y(x \rightsquigarrow y \wedge \mathrm{Tr}_y(\varphi)) \end{aligned}$$

If $\varphi(x)$ is a first-order formula with a free variable x, we use $\mathcal{M} \models \varphi[a]$ to denote that φ is true in $\mathcal{M}$ under the first-order valuation which maps x to an element a of the universe of $\mathcal{M}$. The fact that the above translation is truth-preserving is expressed as follows:

[3] A frame with a binary relation R and universe U is said to be **finitely branching** if for each node $x \in U$ we have that $\{y \in U \mid xRy\}$ is finite.

Proposition 1. *If φ is a BML-formula, then $\mathcal{M}, u \models \varphi$ (in the sense of BML) iff $\mathcal{M} \models \mathrm{Tr}_x(\varphi)[u]$ (in first-order logic).*

Given this translation, we can now state **van Benthem's characterization** theorem:

> *Basic modal logic is the fragment of first-order logic whose truth remains invariant by bisimulations.*

Or, more precisely stated:

Theorem 2 (Characterization [104])**.** *Let $\varphi(x)$ be a first-order formula (over the signature σ) with one free variable x. Then:*

1. *$\varphi(x)$ is $\leftrightarrows$-invariant over Kripke models*
2. *$\varphi(x)$ is logically equivalent over Kripke models to a formula of BML.*

In other words, we have that (translated) formulas of BML are precisely those first-order formulas with one free variable whose truth value always coincides in pairs of pointed models which are bisimilar.

One might ask what is the benefit of using basic modal logic instead of the more expressive first-order logic. As we have mentioned before, there are usually many trade-offs between expressiveness and the complexity of various decision problems. One particular example for our case is that of the **satisfiability** problem: in first-order logic it is undecidable, while in BML it is 'merely' PSPACE-COMPLETE [75].

Another two problems, which we will also explore in the context of XPath over data trees, are those of definability and separation. A class K of pointed models is said to be *definable* by a set of formulas Γ when for every pointed model $\mathcal{M}, w$ it happens that $\mathcal{M}, w \models \Gamma$ iff $\mathcal{M}, w \in K$. The **definability** problem over BML asks, when given a class K of pointed models, if there exists a set Γ of BML-formulas such that K is definable by Γ. The definability theorem for basic modal logic [30] provides necessary and sufficient conditions for the K to be definable by a set of formulas, and other necessary and sufficient conditions for it to be definable by a *single* formula. This theorem can be stated as follows:

> *A class of pointed models K is definable by means of a set of basic modal formulas if and only if K is closed under ultraproducts and bisimulations, and the complement of K is closed under ultrapowers. Also, K is definable by a single basic modal formula if and only if both K and its complement are closed under ultraproducts and bisimulations.*

The closely related **separation** problem asks, given two disjoint classes of pointed models K_1, K_2, whether there exists a set of formulas Γ such that the class K of pointed models defined[4] by Γ contains K_1 and its disjoint with K_2. Similarly as before, the separation theorem [30] can be stated as:

[4]That is, those pointed models $\mathcal{M}, w$ such that $\mathcal{M}, w \models \Gamma$.

> *Let K_1 and K_2 be two disjoint classes of pointed models such that K_1 is closed under bisimulations and ultraproducts and K_2 is closed under bisimulations and ultrapowers. Then there exists a class K that is definable by a set of basic modal formulas, contains K_1, and is disjoint from K_2. Furthermore, if both K_1 and K_2 are closed under bisimulations and ultraproducts, then such K is definable by a single basic modal formula.*

The ultraproduct is an operation that, from a family of models, constructs a new model that in some way is the limit of their constituents. The complete definition is given in Chapter 1, where we adapt ultraproducts and other classical tools to our framework.

It would be desirable to have a sound and complete deductive system that captured the truth of the semantics of BML, that is, one where every universal truth over models can be deduced from the axioms, and where everything that can be deduced from the axioms is an universal truth. Such a system exists: it has all the axioms of propositional logic, it has *modus ponens*, and additionally has one new inference rule and one new axiom schema. The inference rule is called **N** (from *necessitation*), and it states that if φ is a theorem, then so is $\Box\varphi$ (recall that $\Box\varphi$ means $\neg\Diamond\neg\varphi$). The *distribution axiom* **K** is the schema:

$$\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi).$$

Other (sound and complete) **axiomatizations** can be given for different semantical fragments of BML. For example, for the fragment where all frames have a relation that is reflexive we would take our previous axiomatization of the full BML and add the following axiom schema[5]:

$$\Box\varphi \to \varphi$$

I.1.2 XPath$_=$ basics

Basic modal logic, for all its apparent simplicity, is quite expressive and versatile in some aspects. Even with just propositional variables and a single modality (the diamond $\Diamond$), a pointed model can be queried for many interesting topological properties. And, by just adding extra modalities, a wealth of interesting new logics can be constructed, such as Linear Temporal Logic (LTL) [11] or Propositional Dynamic Logic (PDL) [60].

However, BML is simply unequipped to deal with some type of domains and queries. One of these domains, close to that of the labeled graphs of Kripke models, is that of graphs in which each node has a data value taken from some fixed set of possible values (such as $\mathbb{N}$ or words over some alphabet). If this set of possible values is finite, we can codify them into a valuation of the propositional values: with only 2 possible data values, we can use p_0 to query whether a node has 'the same data value' as a neighbor: we use the formula $(p_0 \to \Diamond p_0) \wedge (\neg p_0 \to \Diamond\neg p_0)$. By using combinations of the truth values of many p_i, we can expand this idea to any finite set of values (making very large and exhaustive

[5]This axiom schema is known in the literature as T, or the *Reflexivity Axiom*.

formulas), but we cannot expand the idea to infinite sets of values. Indeed, being able to query for the exact data value in a node is not a good extension if our aim is to be able to *compare* data values.

While we have explained that data comparison cannot be simulated by querying for particular data values in a node, one may ask: why not have both possibilities? There are three main reasons for this. First, we want to consider as equivalent two models that differ only in a (bijective) renaming of the data values, so as to make the logic independent on the data domain. Second, querying for particular data values might require an infinitary logic, a case that is preferable to avoid when thinking of applications into database querying languages. Third, having a data-querying operation would in some sense trivialize the logic, since, for a logic with both data comparison and data querying, the notion of bisimulation and many related theorems would be almost identical to those of BML. Therefore, we want a logic that, without being able to compare values against data constants, is capable of making data comparisons between pairs of nodes.

Of course, comparing data values in nodes of graphs is not a capricious wish, but a type of query that often occurs in databases. To introduce our main domain of discourse, we observe that the abstraction of an XML (Extensible Markup Language) document is a **data tree**, a tree whose nodes contain a single **label** (or tag) taken from a finite set $\mathbb{A}$ and a single **data value** taken from an infinite set $\mathbb{V}$ (equivalently from our theoretical perspective, as we only care about data *comparison*, we can say that there is an equivalence relation $=$ defined over the nodes). For an example[6] of a data tree derived from an XML representing a bibliographical database, see Figure 7.

Introducing our central focus of study, XPath is the most widely used query language for XML documents; it is an open standard and constitutes a World Wide Web Consortium (W3C) Recommendation [24]. When suitably simplified and reinterpreted as a logic, XPath can be seen as an extension of BML, but in the context of data trees, with a finite number of propositional variables, and also, fundamentally, with the capacity to deal with data comparisons between nodes. And while, as we will see, it has many points in common with basic modal logic, it also has many complexities that are absent in that simpler framework.

XPath can express properties such as:

'This node has label a' (8)

'This node has two children with different data values' (9)

'The ending node of this path has label a and is the grandchild of the starting node' (10)

'The starting node of this path has the same data value as one of its children with label c, its ending node has label a and is a grandchild of the starting node, and the sole child of the starting node in this path has label b'. (11)

As indicated by the example properties, XPath is a logic with two sorts of formulas, or expressions: **node expressions**, which represent properties of nodes and whose truth

[6]This figure is taken from Diego Figueira's slides of ICDT14. Used with permission.

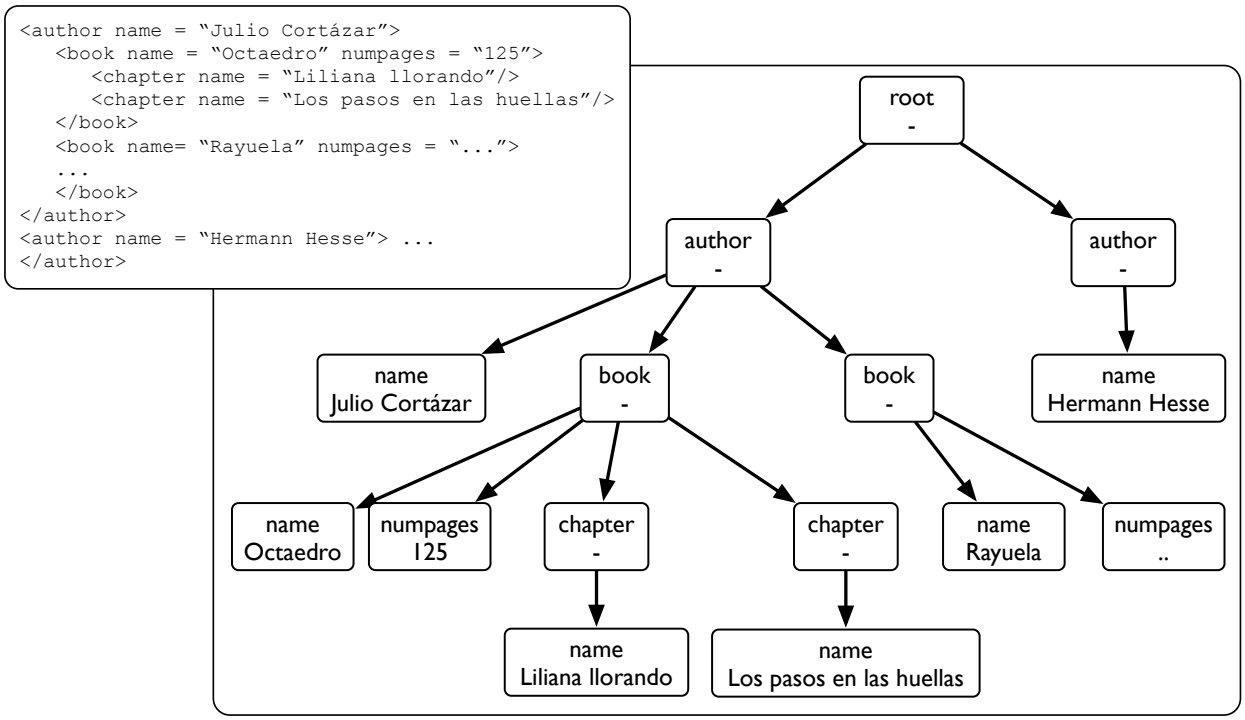


Figure 7: In this representation of an XML document as a data tree, the set of labels is $\mathbb{A} = \{\text{root, author, name, book, numpages, chapter}\}$, and the data domain is $\mathbb{V} = \{-\} \cup \mathbb{N} \cup \{\text{strings of characters}\}$.

values are analyzed in nodes (as in BML), and **path expressions**, which represent properties of paths and are analyzed in pairs of nodes. XPath contains the $\downarrow$ operator, which, similarly to the $\Diamond$ operator of basic modal logic, relates a node in a tree with a child, but XPath can also navigate the tree with other so-called navigational axes such as $\uparrow$ (which relates a node with is parent), $\downarrow_*$ (descendant), $\uparrow^*$ (ancestor), $\leftarrow$ (left sibling), $\rightarrow$ (right sibling), et cetera. Core-XPath [57] is the fragment of XPath 1.0 containing only the navigational behavior of XPath; it is able to express properties (8) and (10) of our previous examples, but not (9) or (11). Core-Data-XPath [17], on the other hand, can express them all. Indeed, $\text{XPath}_=$ is the extension of Core-XPath with (in)equality tests between attributes of elements in an XML document.

In this thesis we will focus on simplified fragments of $\text{XPath}_=$ corresponding to the navigational part of XPath 1.0 with data equality and inequality. Furthermore, altough XPath supports many navigational axes, we will mostly work only with the downard subfragment $\text{XPath}_=(\downarrow)$ (which only contains the navigational axis $\downarrow$), and with the vertical subfragment $\text{XPath}_=(\uparrow\downarrow)$ (which contains both $\downarrow$ and $\uparrow$).

We now proceed to more formal definitions about the models, the language, and the semantics of $\text{XPath}_=$.

Data trees

A **tree** is a connected graph with no cycles; even though only finite graphs make sense in the context of databases, we allow them to be infinite for our study of expressive power of logics. When we speak of trees, we will actually refer to *rooted trees*, that is, trees where one node or vertex has been designated as the root. We say that a tree is **ordered** when the children of all nodes have a particular ordering; we will not make this definition more precise, since we will work with *unordered* trees. We say that a tree is **k-ranked** if every node has at most k children. More often we refer to rank when talking about classes of trees, and we say that a class of trees is **unranked** if there is no fixed bound on the number of children the nodes can have. A tree is said to be **finitely branching** if every node has finitely many children. Observe that being finitely branching is not the same as being ranked, save in the finite case: imagine an infinite tree where the root has one child, which has two children, which themselves have three children each, and so on.

Let $\mathit{Trees}(A)$ be the set of unordered and unranked (finite or infinite) trees where each node is provided with a letter from an arbitrary alphabet $\mathbb{A}$. We say that $\mathcal{T}$ is a **data tree** if it is a tree from $\mathit{Trees}(\mathbb{A} \times \mathbb{V})$, where $\mathbb{A}$ is a finite set of **labels** and $\mathbb{V}$ is an infinite set of **data values** (equivalently for our perspective of data comparison, we can think of data trees as trees from $\mathit{Trees}(\mathbb{A})$ which are provided with an equivalence relation for its nodes). For instance, the tree $\mathcal{T}$ of Figure 8 belongs to $\mathit{Trees}(\{a, b\} \times \mathbb{N})$. For any given data tree $\mathcal{T}$, we typically denote by T its set of nodes. We use letters x, y, z, u, v, w as variables for nodes. Given a node $x \in T$ of $\mathcal{T}$, we write $\mathit{label}(x) \in \mathbb{A}$ to denote the node's label, and $\mathit{data}(x) \in \mathbb{V}$ to denote the node's data value.

Given two nodes $x, y \in T$ we write $x \to y$ if y is a child of x, and $x \xrightarrow{n} y$ if y is a descendant of x at distance n. In particular, $\xrightarrow{1}$ is the same as $\to$, and $\xrightarrow{0}$ is the identity relation. We denote by $x \xrightarrow{\leq m} y$ the fact that $x \xrightarrow{n} y$ for some $n \leq m$. $(\xrightarrow{n} y)$ denotes the sole ancestor of y at distance n (assuming it has one).

If x, y are nodes in a data tree $\mathcal{T}$, we say that $\mathcal{T}, x$ is a **pointed data tree**, and that $\mathcal{T}, x, y$ is a **two-pointed data tree**.

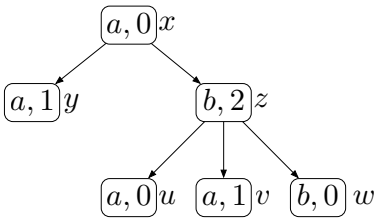


Figure 8: A graphical representation of a data tree $\mathcal{T}$ with a and b as labels and integers as data values. The graphical positions of u, v, and w are interchangeable, as are those of y and z; all these graphical permutations would yield the same underlying (*unordered*) data tree.

Recall that, since we will not work with fragments of XPath capable of determining the particular data values of nodes, our formalism of data trees is equivalent to that where

instead of data values we have an equivalence relation $=$ over the nodes of the tree. We might switch to this perspective when convenient, but it is generally easier to illustrate examples of data trees using concrete data values instead of abstract equivalence classes.

Our main fragments of XPath$_=$

As we have previously said, we work with a simplification of XPath stripped of its syntactic sugar, corresponding to the navigational part of XPath 1.0 with data equality and inequality. XPath$_=$ is a two-sorted language, with **node expressions** (that we typically notate φ, ψ, η) expressing properties of nodes, and with **path expressions** (that we typically notate α, β, γ) expressing properties of paths. **Vertical XPath** over the set of labels $\mathbb{A}$, notated simply XPath$_=(\uparrow\downarrow)$, is defined by mutual recursion as follows:

$$\begin{array}{lll} \alpha, \beta & ::= & o \mid [\varphi] \mid \alpha\beta \mid \alpha \cup \beta \qquad\qquad o \in \{\varepsilon, \uparrow, \downarrow\} \\ \varphi, \psi & ::= & a \mid \neg\varphi \mid \varphi \wedge \psi \mid \langle\alpha\rangle \mid \langle\alpha = \beta\rangle \mid \langle\alpha \neq \beta\rangle \qquad a \in \mathbb{A} \end{array}$$

As usual, we allow formulas of the type $\varphi \vee \psi$, which we take to mean $\neg(\neg\varphi \wedge \neg\psi)$. After giving the semantics, we will also see that $\langle\alpha\rangle$ is unessential[7] and that $\cup$ is unessential for node expressions.

We call **downward XPath$_=$**, notated XPath$_=(\downarrow)$, to the syntactic fragment which only uses the navigation axis $\downarrow$, but not $\uparrow$. An **XPath$_=(\uparrow\downarrow)$-formula** [resp. **XPath$_=(\downarrow)$-formula**] is either a node expression or a path expression of XPath$_=(\uparrow\downarrow)$ [resp. XPath$_=(\downarrow)$].

Next we define the semantics of XPath$_=(\uparrow\downarrow)$ in a given data tree $\mathcal{T}$, defining the set of nodes that satisfy node expressions and the set of pairs of nodes that satisfy path expressions:

$$\begin{aligned} [\![\downarrow]\!]^{\mathcal{T}} &= \{(x,y) \mid x{\rightarrow}y\} \\ [\![\uparrow]\!]^{\mathcal{T}} &= \{(x,y) \mid y{\rightarrow}x\} \\ [\![\varepsilon]\!]^{\mathcal{T}} &= \{(x,x) \mid x \in T\} \\ [\![\alpha\beta]\!]^{\mathcal{T}} &= \{(x,z) \mid \exists y \in T : (x,y) \in [\![\alpha]\!]^{\mathcal{T}}, (y,z) \in [\![\beta]\!]^{\mathcal{T}}\} \\ [\![\alpha \cup \beta]\!]^{\mathcal{T}} &= [\![\alpha]\!]^{\mathcal{T}} \cup [\![\beta]\!]^{\mathcal{T}} \\ [\![[\varphi]]\!]^{\mathcal{T}} &= \{(x,x) \mid x \in [\![\varphi]\!]^{\mathcal{T}}\} \\ [\![a]\!]^{\mathcal{T}} &= \{x \in T \mid \mathit{label}(x) = a\} \\ [\![\neg\varphi]\!]^{\mathcal{T}} &= T \setminus [\![\varphi]\!]^{\mathcal{T}} \\ [\![\varphi \wedge \psi]\!]^{\mathcal{T}} &= [\![\varphi]\!]^{\mathcal{T}} \cap [\![\psi]\!]^{\mathcal{T}} \\ [\![\langle\alpha\rangle]\!]^{\mathcal{T}} &= \{x \in T \mid \exists y \in T : (x,y) \in [\![\alpha]\!]^{\mathcal{T}}\} \\ [\![\langle\alpha = \beta\rangle]\!]^{\mathcal{T}} &= \{x \in T \mid \exists y,z \in T : (x,y) \in [\![\alpha]\!]^{\mathcal{T}}, (x,z) \in [\![\beta]\!]^{\mathcal{T}}, \mathit{data}(y) = \mathit{data}(z)\} \\ [\![\langle\alpha \neq \beta\rangle]\!]^{\mathcal{T}} &= \{x \in T \mid \exists y,z \in T : (x,y) \in [\![\alpha]\!]^{\mathcal{T}}, (x,z) \in [\![\beta]\!]^{\mathcal{T}}, \mathit{data}(y) \neq \mathit{data}(z)\} \end{aligned}$$

[7] Even though it is unessential, we keep $\langle\alpha\rangle$ in the logic in order to make XPath$_=$ a syntactical extension of the data-oblivious Core-XPath, which has $\langle\alpha\rangle$ but does not have $\langle\alpha = \beta\rangle$ nor $\langle\alpha \neq \beta\rangle$.

Observe that if α is an XPath$_=(\downarrow)$-path expression and $(x, y) \in [\![\alpha]\!]^{\mathcal{T}}$, then either $x = y$ or y is a strict descendant of x in the tree $\mathcal{T}$. Note that $\langle \alpha \neq \beta \rangle$ is *not* the same as $\neg\langle \alpha = \beta \rangle$. Indeed, $\langle \alpha \neq \beta \rangle$ states that there exist paths α and β starting from the current node such that they end in nodes with different data values, while $\neg\langle \alpha = \beta \rangle$ functions as a kind of 'box' from BML, saying that *all* such paths (if they exist at all) must end in differing data values. Also observe that $\langle \alpha \rangle$ has the same semantics as $\langle \alpha = \alpha \rangle$. We usually write $\downarrow^n$ to mean $\underbrace{\downarrow \ldots \downarrow}_{n}$, and $\uparrow^n$ to mean $\underbrace{\uparrow \ldots \uparrow}_{n}$.

For a node expression φ, we write $\mathcal{T}, x \models \varphi$ to denote $x \in [\![\varphi]\!]^{\mathcal{T}}$, and in that case we say that $\mathcal{T}, x$ **satisfies** φ or that φ is true at $\mathcal{T}, x$. In the same way, for a path expression α, we write $\mathcal{T}, x, y \models \alpha$ to denote $(x, y) \in [\![\alpha]\!]^{\mathcal{T}}$, and we say that $\mathcal{T}, x, y$ **satisfies** α or that α is true at $\mathcal{T}, x, y$. We say that the node expressions φ, ψ of XPath$_=$ are **equivalent** (notated $\varphi \equiv \psi$) iff $[\![\varphi]\!]^{\mathcal{T}} = [\![\psi]\!]^{\mathcal{T}}$ for all data trees $\mathcal{T}$. Similarly, path expressions α, β of XPath$_=$ are **equivalent** (notated $\alpha \equiv \beta$) iff $[\![\alpha]\!]^{\mathcal{T}} = [\![\beta]\!]^{\mathcal{T}}$ for all data trees $\mathcal{T}$.

We define $\mathrm{Th}_{\uparrow\downarrow}(\mathcal{T}, x)$ [resp. $\mathrm{Th}_{\downarrow}(\mathcal{T}, x)$] as the set of all XPath$_=(\uparrow\downarrow)$-node expressions [resp. XPath$_=(\downarrow)$-node expressions] true at $\mathcal{T}, x$. Similarly, we define $\mathrm{Th}_{\uparrow\downarrow}(\mathcal{T}, x, y)$ [resp. $\mathrm{Th}_{\downarrow}(\mathcal{T}, x, y)$] as the set of all XPath$_=(\uparrow\downarrow)$-path expressions [resp. XPath$_=(\downarrow)$-path expressions] true at $\mathcal{T}, x, y$.

In terms of expressive power of node expressions, it is easy to see that $\cup$ is unessential (see [44, §2.2]): every XPath$_=$-node expression φ has an equivalent φ' with no $\cup$ in its path expressions. It is enough to use the following equivalences to eliminate occurrences of $\cup$:

$$\begin{aligned} \langle \alpha \star \beta \rangle &\equiv \langle \beta \star \alpha \rangle \\ \langle \beta(\alpha \cup \alpha')\beta' \rangle &\equiv \langle \beta\alpha\beta' \rangle \vee \langle \beta\alpha'\beta' \rangle \\ \langle \gamma \star \beta(\alpha \cup \alpha')\beta' \rangle &\equiv \langle \gamma \star \beta\alpha\beta' \rangle \vee \langle \gamma \star \beta\alpha'\beta' \rangle \end{aligned}$$

where $\star \in \{=, \neq\}$.

For the case of path expressions, there is no such possible elimination of $\cup$. Indeed the path expression $\downarrow[a] \cup \downarrow\downarrow[a]$ cannot be restated without using $\cup$. The reason of this is that there are no intersections or complementations of path expressions (in §1.4 and §1.5 we study this issue).

Now, let us express the example properties (8), (9), (10), and (11) using the language of XPath$_=(\downarrow)$:

$$\begin{aligned} &\varphi = a && \text{(Expresses (8))} \\ &\psi = \langle \downarrow \neq \downarrow \rangle && \text{(Expresses (9))} \\ &\alpha = \downarrow\downarrow[a] = \downarrow^2[a] && \text{(Expresses (10))} \\ &\beta = [\langle \varepsilon = \downarrow[c] \rangle]\downarrow[b]\downarrow[a] && \text{(Expresses (11))} \end{aligned}$$

It is interesting enough to remark that there are some properties that, non-obviously, can be expressed in XPath$_=(\uparrow\downarrow)$ but not in XPath$_=(\downarrow)$. An example is the path property:

'this descending path covers 3 nodes, all of which have different data values'. (12)

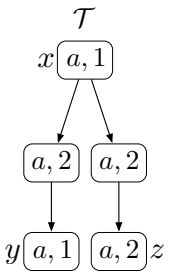


Figure 9: $\mathcal{T}, x, y \models [\langle \varepsilon \neq \downarrow \rangle \wedge \langle \varepsilon \neq \downarrow\downarrow \rangle]\downarrow[\langle \varepsilon \neq \downarrow \rangle]\downarrow$, but x and y have the same data value; the witness for $\langle \varepsilon \neq \downarrow\downarrow \rangle$ is not y, but z.

In other words, this property is satisfied by pairs of nodes such that the second node is a grandchild of the first, and such that it, its parent, and its grandparent all have different data values. If trying to express this in XPath$_=(\downarrow)$, a first attempt might be to take:

$$\alpha = [\langle \varepsilon \neq \downarrow \rangle \wedge \langle \varepsilon \neq \downarrow\downarrow \rangle]\downarrow[\langle \varepsilon \neq \downarrow \rangle]\downarrow$$

However, there is no guarantee that the *witness* for the subformula $\langle \varepsilon \neq \downarrow\downarrow \rangle$ is the ending node of the path: see Figure 9 for a counterexample. On the other hand, we can verify that the following XPath$_=(\uparrow\downarrow)$-path expression has the desired semantics:

$$\beta = \downarrow[\langle \varepsilon \neq \uparrow \rangle]\downarrow[\langle \varepsilon \neq \uparrow \rangle \wedge \langle \varepsilon \neq \uparrow\uparrow \rangle]$$

It is relevant to emphasize that this path expression has the desired semantics because we are restricted to data trees, where each node has at most one parent; on the wider context of data graphs, β would not express our desired property.

While we saw that our given α in particular did not work, we have not actually proved that there is no XPath$_=(\downarrow)$-path expression that does. We will prove this fact later, after we show a van Benthem-like *characterization* theorem for XPath$_=(\downarrow)$. In order to do so, however, we need to first introduce a central concept: that of XPath$_=$-bisimulation.

XPath$_=(\downarrow)$-bisimulation

As in the case of basic modal logic, bisimulation is a fundamental notion that gives a strong definition of indistinguishability by operations of XPath$_=$. In this case, however, the definition is more complex in order to deal with the possibility of forking from a single node into two paths of arbitrary length that compare data only at the end. The following definitions, as well as those we will later see for ℓ, XPath$_=(\uparrow\downarrow)$, and (r, s, k) bisimulation, were originally developed in [44] and [45].

Given two data trees $\mathcal{T}$ and $\mathcal{T}'$, with respective sets of nodes T, T', we say that the relation $Z \subseteq T \times T'$ is an **XPath$_=(\downarrow)$-bisimulation** if for all $(x, x') \in T \times T'$ such that $(x, x') \in Z$, the following conditions hold:

- (**Harmony**) $label(x) = label(x')$

- (**Zig**) If we have paths $x \to v_1 \to \ldots \to v_m$ and $x \to w_1 \to \ldots \to w_n$ in T, then there exist $v'_1, \ldots, v'_m$ and $w'_1, \ldots, w'_n$ in T' such that:
  - $x' \to v'_1 \to \ldots \to v'_m$,
  - $(v_i, v'_i) \in Z$ for all $1 \leq i \leq m$,
  - $x' \to w'_1 \to \ldots \to w'_n$,
  - $(w_i, w'_i) \in Z$ for all $1 \leq i \leq n$, and
  - $data(v_m) = data(w_n)$ iff $data(v'_m) = data(w'_n)$.
- (**Zag**) (The symmetric condition to **Zig**, starting from two paths in T'). If we have paths $x' \to v'_1 \to \ldots \to v'_m$ and $x' \to w'_1 \to \ldots \to w'_n$ in T', then there exist $v_1, \ldots, v_m$ and $w_1, \ldots, w_n$ in T such that:
  - $x \to v_1 \to \ldots \to v_m$,
  - $(v_i, v'_i) \in Z$ for all $1 \leq i \leq m$,
  - $x \to w_1 \to \ldots \to w_n$,
  - $(w_i, w'_i) \in Z$ for all $1 \leq i \leq n$, and
  - $data(v_m) = data(w_n)$ iff $data(v'_m) = data(w'_n)$.

In Figure 10 we show an example of the process of checking **Zig** for two nodes. Note that, since each node in a tree has exactly one parent (except the root), to determine a descending path it suffices to give its starting and ending nodes.

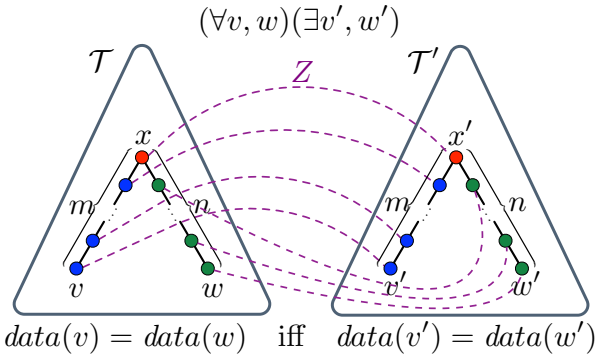


Figure 10: The process of checking the condition **Zig** of XPath$_=(\downarrow)$-bisimulation for $(x, x') \in Z$ entails checking that all descending paths starting from $x \in T$ can be mirrored pairwise in $\mathcal{T}'$.

We now define when two nodes $x \in T$ and $x' \in T'$ are said to be **XPath$_=(\downarrow)$-bisimilar**, notated $\mathcal{T}, x \leftrightarrows^{\downarrow} \mathcal{T}', x'$:

$$\mathcal{T}, x \leftrightarrows^{\downarrow} \mathcal{T}', x' \overset{\text{def}}{\Leftrightarrow} \text{ there is an XPath}_=(\downarrow)\text{-bisimulation } Z \text{ with } (x, x') \in Z.$$

Note that in particular, for any pointed data trees $\mathcal{T}, x$ and $\mathcal{T}', x'$, if x and x' are leaves with the same label, then $\mathcal{T}, x \leftrightarrows^{\downarrow} \mathcal{T}', x'$. Also note that, if (r, r') is in a $\mathrm{XPath}_{=}(\downarrow)$-bisimulation Z for r and r' the roots of $\mathcal{T}$ and $\mathcal{T}'$, then every element in T must be related to at least one element in T', and vice versa, that is: $\pi_1(Z) = T$ and $\pi_2(Z) = T'$.

The one-direction version of bisimilarity, **similarity**, is defined as in BML by not asking for the **Zag** condition, and it is analogously notated as $\rightrightarrows^{\downarrow}$.

An important observation is that, as in the case of BML, $\mathrm{XPath}_{=}(\downarrow)$-bisimulations and simulations (and the other types of (bi)simulations we study) are closed under union. That is, if $Z_1 \subseteq T \times T'$ and $Z_2 \subseteq T \times T'$ are two $\mathrm{XPath}_{=}(\downarrow)$-(bi)simulations, then $Z_1 \cup Z_2$ also is an $\mathrm{XPath}_{=}(\downarrow)$-(bi)simulation. This immediately implies that, if a (bi)simulation exists, then there also exists a *maximal* (bi)simulation.

It can be shown, using similar counterexamples as those seen in §I.1.1 for BML[8], that we still have that a simulation is not necessarily a bisimulation, and that a similarity in both ways between pointed data trees does not imply bisimilarity between them.

$\mathrm{XPath}_{=}(\uparrow\downarrow)$-bisimulation

Given that path expressions in $\mathrm{XPath}_{=}(\uparrow\downarrow)$ can alternate $\downarrow$ and $\uparrow$ indefinitely, it could intuitively appear that the notion of $\mathrm{XPath}_{=}(\uparrow\downarrow)$-bisimulation should be quite complex. This it not the case; since we are working over data trees, the determinism of the $\uparrow$ and thus the existence of quite compact normal forms (of which we speak later) allows us to have a relatively simple definition [45].

Given two data trees $\mathcal{T}$ and $\mathcal{T}'$, with respective sets of nodes T, T', we say that the relation $Z \subseteq T \times T'$ is an **$\mathrm{XPath}_{=}(\uparrow\downarrow)$-bisimulation** if for all $(x, x') \in T \times T'$ such that $(x, x') \in Z$, the following conditions hold:

- (**Harmony**) $label(x) = label(x')$
- (**Zig**) If $v \xrightarrow{m} x$ and $v \xrightarrow{n} w$ then there are $v', w' \in T'$ such that:
  - $v' \xrightarrow{m} x'$,
  - $v' \xrightarrow{n} w'$,
  - wZw', and
  - $data(w) = data(x)$ iff $data(w') = data(x')$.
- (**Zag**) (The symmetric condition to **Zig**, starting in T'). If $v' \xrightarrow{m} x'$ and $v' \xrightarrow{n} w'$ then there are $v, w \in T$ such that:
  - $v \xrightarrow{m} x$,
  - $v \xrightarrow{n} w$,
  - wZw', and

[8] The data values can be made irrelevant to the notions of $\mathrm{XPath}_{=}$-bisimulation by giving the same data value to all nodes.

- $data(w) = data(x)$ iff $data(w') = data(x')$.

In Figure 11 we show an example of the process of checking **Zig** for two nodes. Since from the root of a tree we can reach any node, we remark that any non-empty XPath$_=$(↑↓)-bisimulation Z between two data trees $\mathcal{T}$ and $\mathcal{T}'$ connects every node of T with at least one node of T', and every node of T' with at least one node of T.

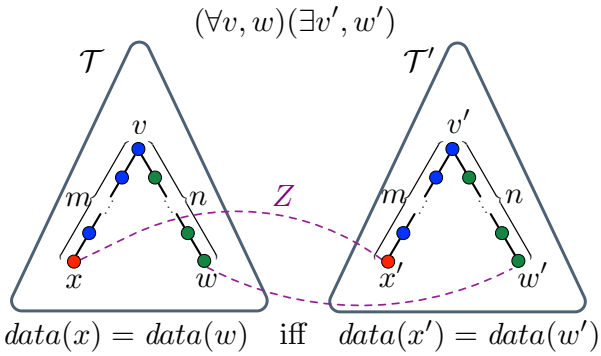


Figure 11: The process of checking the condition **Zig** of XPath$_=$(↑↓)-bisimulation for $(x, x') \in Z$.

We define when two nodes $x \in T$ and $x' \in T'$ are said to be **XPath$_=$(↑↓)-bisimilar**, notated $\mathcal{T}, x \leftrightarrows^{\uparrow\downarrow} \mathcal{T}', x'$:

$$\mathcal{T}, x \leftrightarrows^{\uparrow\downarrow} \mathcal{T}', x' \overset{\text{def}}{\Leftrightarrow} \text{ there is an XPath}_=(\uparrow\downarrow)\text{-bisimulation } Z \text{ such that } (x, x') \in Z.$$

Bounded notions of bisimulation

The previously defined notions of XPath$_=$(↓) and XPath$_=$(↑↓)-bisimilarity might be too strong for some purposes, as they may distinguish two nodes that only fail the bisimilarity test with a pair of paths of high length. In §1.2 we write the definitions for two 'bounded' notions of bisimulation from [45], ℓ-bisimulation and (r, s, k)-bisimulation, which only need to explore the tree in some restricted ways.

Logical equivalence

As in BML, we say that two pointed data trees are **equivalent** respect to some XPath fragment when the truth value of any node expression of that fragment coincides over the two nodes. The formal definition of equivalence for the fragments XPath$_=$(↓) and XPath$_=$(↑↓) (notated $\equiv^{\downarrow}$ and $\equiv^{\uparrow\downarrow}$ respectively) is as follows:

$$\mathcal{T}, x \equiv^{\downarrow} \mathcal{T}', x' \overset{\text{def}}{\Leftrightarrow} \text{ for all XPath}_=(\downarrow)\text{-node expressions } \varphi, \mathcal{T}, x \models \varphi \text{ iff } \mathcal{T}', x' \models \varphi.$$
$$\mathcal{T}, x \equiv^{\uparrow\downarrow} \mathcal{T}', x' \overset{\text{def}}{\Leftrightarrow} \text{ for all XPath}_=(\uparrow\downarrow)\text{-node expressions } \varphi, \mathcal{T}, x \models \varphi \text{ iff } \mathcal{T}', x' \models \varphi.$$

A node expression is said to be **positive** if it contains no negation. We define

$$\mathcal{T}, x \Rrightarrow^{\downarrow} \mathcal{T}', x' \overset{\text{def}}{\Leftrightarrow} \mathcal{T}, x \models \varphi \Rightarrow \mathcal{T}', x' \models \varphi \text{ for every positive node expression } \varphi.$$

In Section I.1.3 we will state the important results that relate the various notions of bisimulation with the corresponding types of equivalence.

Node-labeled vs edge-labeled data graphs

In some kinds of databases, such as in social networks, it is more natural to have labels in the edges connecting nodes rather than in the nodes themselves, specially when wanting to emphasize relations between objects. This is particularly true in databases whose underlying topology is not a tree but a graph, which are common in knowledge representation.

$\text{XPath}_{=}(\downarrow)$ can be adapted to this framework of edge-labeled data graphs by some small changes in the definition of the node and path expressions:

$$\begin{aligned} \alpha, \beta &::= \varepsilon \mid \downarrow_a \mid \alpha\beta \mid \alpha \cup \beta \mid [\varphi] && (a \in \mathbb{A}) \\ \varphi, \psi &::= \neg\varphi \mid \varphi \wedge \psi \mid \langle\alpha\rangle \mid \langle\alpha = \beta\rangle \mid \langle\alpha \neq \beta\rangle. \end{aligned}$$

We call this logic $\text{XPath}_{=}(\downarrow_{\mathbf{a}})$, to highlight its edge-labeled nature. Given a data graph $\mathcal{G}$ with nodes G and labeled edges $E \subseteq G \times \mathbb{A} \times G$, the semantics for $\text{XPath}_{=}(\downarrow_{\mathbf{a}})$-formulas over $\mathcal{G}$ is as follows:

$$\begin{aligned} [\![\downarrow_a]\!]^{\mathcal{G}} &= \{(x,y) \mid (x,a,y) \in E\} \\ [\![\varepsilon]\!]^{\mathcal{G}} &= \{(x,x) \mid x \in G\} \\ [\![\alpha\beta]\!]^{\mathcal{G}} &= \{(x,z) \mid \exists y \in G : (x,y) \in [\![\alpha]\!]^{\mathcal{G}}, (y,z) \in [\![\beta]\!]^{\mathcal{G}}\} \\ [\![\alpha \cup \beta]\!]^{\mathcal{G}} &= [\![\alpha]\!]^{\mathcal{G}} \cup [\![\beta]\!]^{\mathcal{G}} \\ [\![[\varphi]]\!]^{\mathcal{G}} &= \{(x,x) \mid x \in [\![\varphi]\!]^{\mathcal{G}}\} \\ [\![\neg\varphi]\!]^{\mathcal{G}} &= G \setminus [\![\varphi]\!]^{\mathcal{G}} \\ [\![\varphi \wedge \psi]\!]^{\mathcal{G}} &= [\![\varphi]\!]^{\mathcal{G}} \cap [\![\psi]\!]^{\mathcal{G}} \\ [\![\langle\alpha\rangle]\!]^{\mathcal{G}} &= \{x \in G \mid \exists y \in G : (x,y) \in [\![\alpha]\!]^{\mathcal{G}}\} \\ [\![\langle\alpha = \beta\rangle]\!]^{\mathcal{G}} &= \{x \in G \mid \exists y,z \in G : (x,y) \in [\![\alpha]\!]^{\mathcal{G}}, (x,z) \in [\![\beta]\!]^{\mathcal{G}}, \mathit{data}(y) = \mathit{data}(z)\} \\ [\![\langle\alpha \neq \beta\rangle]\!]^{\mathcal{G}} &= \{x \in G \mid \exists y,z \in G : (x,y) \in [\![\alpha]\!]^{\mathcal{G}}, (x,z) \in [\![\beta]\!]^{\mathcal{G}}, \mathit{data}(y) \neq \mathit{data}(z)\} \end{aligned}$$

Bisimulation in the context of edge-labeled data graphs is very close to the one we gave for node-labeled data trees, and its definition is given in Chapter 3, where the framework of edge-labeled graphs is used. Importantly, having data graphs that are edge-labeled instead of node-labeled does not significantly alter the key notions and results of $\text{XPath}_{=}$: each node-labeled data tree $\mathcal{T}$ can be represented as an edge-labeled data graph $\widetilde{\mathcal{T}}$ in a straightforward way, and vice versa, with a translation that is invariant under the respective notions of bisimulation.

I.1.3 Known model theory of XPath$_=$

We now proceed to state some important properties and theorems of XPath$_=$ that are mentioned or used later in this thesis. Unless otherwise stated, all these results where originally developed in the foundational paper [45].

Bisimulation and equivalence

We now state theorems from [45], which are analogous to Hennessy-Milner's characterization theorem, which show the connection between bisimulation and equivalence for the downward and for the vertical fragments of XPath$_=$. As in the case of BML, we confirm that bisimilarity is a strictly stronger notion than logical equivalence, but that both coincide over finitely branching models.

We write more complete versions of these theorems in §1.2, after the definitions of some bounded notions of bisimulation and their corresponding notions of equivalence.

> *Given two pointed data trees $\mathcal{T}, x$ and $\mathcal{T}', x'$, the [bi]similarity $\mathcal{T}, x \rightrightarrows^{\downarrow} \mathcal{T}', x'$ [resp. $\mathcal{T}, x \leftrightarrows^{\downarrow} \mathcal{T}', x'$] implies that $\mathcal{T}, x \Rrightarrow^{\downarrow} \mathcal{T}', x'$ [$\mathcal{T}, x \equiv^{\downarrow} \mathcal{T}', x'$]. The converse is not true in general, but it holds when $\mathcal{T}$ and $\mathcal{T}'$ are finitely branching. Analogously, $\mathcal{T}, x \leftrightarrows^{\uparrow\downarrow} \mathcal{T}', x'$ implies that $\mathcal{T}, x \equiv^{\uparrow\downarrow} \mathcal{T}', x'$. The converse is not true in general, but it holds when $\mathcal{T}$ and $\mathcal{T}'$ are finitely branching.*

The practical importance of theorems of this type is that they affirm that, over finite data structures, logical indistinguishability between nodes can be exactly determined via bisimulations. Finding equivalence classes of entries/nodes in a database that respond equally to all possible queries can be used to improve the performance of an actual querying language: if we want to retrieve all entries that satisfy a certain query, it is enough to check the query (and, recursively, its subqueries) on one node of each equivalence class. In Chapter 3 we perform an exploration of this topic, and calculate tight complexity classes of various bisimulation problems.

Connection to first-order logic

We fix the signature σ with binary relations $\rightsquigarrow$ and $\sim$, and a unary predicate P_a for each $a \in \mathbb{A}$. Any data tree $\mathcal{T}$ can be seen as a first-order σ-structure with universe T, where

$$\begin{aligned} \rightsquigarrow^{\mathcal{T}} &= \{(x,y) \in T^2 \mid x \to y \text{ in } \mathcal{T}\}; \\ \sim^{\mathcal{T}} &= \{(x,y) \in T^2 \mid \mathit{data}(x) = \mathit{data}(y)\}; \\ P_a^{\mathcal{T}} &= \{x \in T \mid \mathit{label}(x) = a\}. \end{aligned}$$

In [44] is shown a truth-preserving translation Tr_x mapping XPath$_=$(↑↓)-node expressions into first-order σ-formulas with one free variable x. Our following translation, given in [5], is slightly more clear than the one described in [44], and they differ importantly in

that ours can consider the translation of path expressions (resulting in first-order formulas with two variables):

$$
\begin{aligned}
\mathrm{Tr}_x(a) &= P_a(x) && (a \in \mathbb{A})\\
\mathrm{Tr}_x(\varphi \dagger \psi) &= \mathrm{Tr}_x(\varphi) \dagger \mathrm{Tr}_x(\psi) && (\dagger \in \{\wedge, \vee\})\\
\mathrm{Tr}_x(\neg\varphi) &= \neg\mathrm{Tr}_x(\varphi)\\
\mathrm{Tr}_x(\langle\alpha\rangle) &= (\exists y)\mathrm{Tr}_{x,y}(\alpha) && (y \text{ a fresh variable})\\
\mathrm{Tr}_x(\langle\alpha = \beta\rangle) &= (\exists y)(\exists z)\big(y \sim z \wedge \mathrm{Tr}_{x,y}(\alpha) \wedge \mathrm{Tr}_{x,z}(\beta)\big) && (y, z \text{ fresh variables})\\
\mathrm{Tr}_x(\langle\alpha \neq \beta\rangle) &= (\exists y)(\exists z)\big(y \not\sim z \wedge \mathrm{Tr}_{x,y}(\alpha) \wedge \mathrm{Tr}_{x,z}(\beta)\big) && (y, z \text{ fresh variables})\\
\mathrm{Tr}_{x,y}(\varepsilon) &= (x = y)\\
\mathrm{Tr}_{x,y}(\downarrow) &= (x \rightsquigarrow y)\\
\mathrm{Tr}_{x,y}(\uparrow) &= (y \rightsquigarrow x)\\
\mathrm{Tr}_{x,y}(\alpha\beta) &= (\exists z)\big(\mathrm{Tr}_{x,z}(\alpha) \wedge \mathrm{Tr}_{z,y}(\beta)\big) && (z \text{ a fresh variable})\\
\mathrm{Tr}_{x,y}(\alpha \cup \beta) &= \mathrm{Tr}_{x,y}(\alpha) \vee \mathrm{Tr}_{x,y}(\beta)\\
\mathrm{Tr}_{x,y}([\varphi]) &= \mathrm{Tr}_x(\varphi) \wedge (x = y).
\end{aligned}
$$

It is easy to see that the above translation is truth-preserving:

Proposition 3. *If φ is a node expression of XPath$_=(\uparrow\downarrow)$ then $\mathcal{T}, u \models \varphi$ iff $\mathcal{T} \models \mathrm{Tr}_x(\varphi)[u]$. If α is a path expression of XPath$_=(\uparrow\downarrow)$ then $\mathcal{T}, u, v \models \alpha$ iff $\mathcal{T} \models \mathrm{Tr}_{x,y}(\alpha)[u, v]$.*

Using the previously given translation, we can now give the characterization theorem for XPath$_=(\downarrow)$-node expressions, the result corresponding to the characterization theorem of BML that we previously showed. In Chapter 1, we will give an analogous result but for XPath$_=(\downarrow)$-*path* expressions.

We can say that:

> *Over data trees, XPath$_=(\downarrow)$ is the fragment of first-order logic (with signature σ) whose truth remains invariant by bisimulations.*

This can be written more precisely as:

Theorem 4 (Characterization)**.** *[45] Let $\varphi(x)$ be a first-order formula (over the signature σ) with one free variable x. Then the following properties are equivalent:*

1. *$\varphi(x)$ is $\underline{\leftrightarrow}^{\downarrow}$-invariant over data trees*
2. *$\varphi(x)$ is logically equivalent over data trees to a node expression of XPath$_=(\downarrow)$.*

Furthermore, in [45] it is shown that there is a relation between the *quantifier rank* of φ and the ℓ such that the equivalent node expression lies in ℓ-XPath$_=(\downarrow)$.

Note that with this result we can demonstrate that the property (12), of being a descending path of 3 nodes all having different data values is not expressible with a single XPath$_=(\downarrow)$-path expression. Indeed, were it to be expressible with a single path expression

α, we would get the node expression $\langle\alpha\rangle$ with the interpretation 'there exists a descending path of depth two, starting from this node, such that all its nodes have different data values'. Now it suffices to show that this first-order-expressible property[9] is not expressible with a single XPath$_=(\downarrow)$-node expression, which by Theorem 4 can be done by giving two XPath$_=(\downarrow)$-bisimilar pointed data trees where its truth value differs. The pointed data trees of Figure 12 serve this purpose. In §1.3.3 we will prove, using a definability theorem, a stronger version of this inexpressibility result: that this node property cannot be expressed even with *sets* of node-expressions.

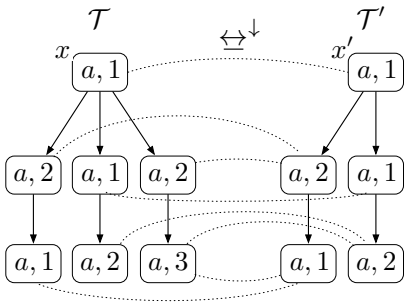


Figure 12: The node version of property (12) is not XPath$_=(\downarrow)$-bisimulation-invariant over data trees: here we can check that $\mathcal{T}, x \leftrightarrows^{\downarrow} \mathcal{T}', x'$, but the property holds on $\mathcal{T}, x$ and not in $\mathcal{T}', x'$.

The statement corresponding to Theorem 4 but adapted to XPath$_=(\uparrow\downarrow)$ is false: a counter-example would be to take the property 'there exists in this tree some node labeled a'. This first-order-expressible property is invariant by bisimulation: given a pointed data tree where it holds, the node that witnesses a must be reachable by some path of the form $\uparrow^m\downarrow^n$; given another pointed data tree that is bisimilar to the first one, it must have (from **Zig**) a corresponding path ending in a node that must necessarily (using **Harmony**) have label a. However, there is no way to express the stated property with a single node expression in XPath$_=(\uparrow\downarrow)$, as such formulas can at most 'see' around a radius that is bounded by their length.

I.2 Focus of study and contributions

In this section we present a quick overview of the main subjects we study in this thesis. We briefly mention previous works that are relevant to our topics of interest, and then proceed to introduce an outline of the problems we answer in this thesis and the approaches we take to tackle them.

[9] $\varphi(x) ::= \exists y(\exists z\, x \rightsquigarrow y \wedge y \rightsquigarrow z \wedge \neg(x \sim y) \wedge \neg(x \sim z) \wedge \neg(y \sim z))$.

I.2.1 Bisimulations

As we have seen, bisimulation is a central concept in BML and XPath$_=$, closely related to that of logical equivalence. In general, bisimulation is a fundamental notion that establishes when two nodes (states) in graph-represented data (transition system) cannot be distinguished by an external observer. It was independently discovered in the areas of computer science and philosophical logic during the 1970s —see [96] for a thorough historical revision of the notion of bisimulation. In both contexts, bisimulation (and its "one-direction" version, *simulation*) appeared as a refinement of the notion of morphism, i.e, "structure-preserving" mappings. In the case of computer science, bisimulation was developed in the context of concurrency theory as a way to study the behavior of programs [85, 92]. In philosophical logic, it was introduced by van Benthem in order to characterize the expressive power of the basic modal logic in terms of a fragment of first-order logic [104].

Nowadays, (bi)simulation is applied in many different fields of computer science. For instance, it is used in concurrency to study behavioral equality for processes [86]; in model checking to tackle the state-explosion problem [25]; in databases as a method for indexing and compressing semi-structured data [87, 41]; in stochastic planning to solve Markov decision processes efficiently [56]; in description logics to understand the expressiveness of some languages [74]; and in natural language generation to define semantic counterparts to the notion of referring expression [8]. Also, the closely related notion of arc consistency is used in constraint satisfaction as an approximation of satisfiability [32, 33] and as a method for finding tractable cases of the satisfiability problem [68, 28].

As we mentioned, XPath is a two-sorted logic, with node expressions and path expressions. Most of the recent model theory on XPath$_=$ focuses on node expressions, but here we extend many results to its path expressions. As a step in doing this, we devise an appropriate notion of bisimulation for path expressions, which we call *binary bisimulation*. We show that this notion, which extends that of unary bisimulation developed in [44], coincides with that of equivalence over finitely branching data trees, and prove a van Benthem-like characterization theorem for path expressions. This binary framework is further away from BML than the unary one of node expressions, and thus the modifications required for the definitions and proofs turned out to be quite complex. Our research on these topics can be found in [2] and [5], and here we present it in Chapter 1, specifically in Section 1.4.

In BML, it is easy to see that deciding whether a relation Z is a bisimulation can be done in polynomial time over the number of nodes in the models, as we have a clear bound on the search space needed to verify **Zig** and **Zag** for each pair in Z. For XPath$_=$ over data trees, the same idea applies: while, for a given pair of nodes in Z, **Zig** or **Zag** putatively need to check paths of arbitrary depth, the size of the data trees gives us a bound to this search. As a consequence, by starting with a relation $Z = T \times T'$ and iteratively removing pairs that fail any bisimulation test, XPath$_=$ bisimilarity between pointed data trees can be determined in PTime, as was the case for BML. It is no longer so easy to answer the bisimulation problem in the case of XPath$_=$ *over data graphs*, as in this case we lack an immediately obvious bound to the depth of the paths we have to check in the **Zig** and

Zag operations. The findings on this topic presented in Chapter 3 are from [1], where we analyze the complexity classes of different bisimilarity problems (unbounded bisimulation, bisimulation bounded by constants, and bisimulation bounded by functions) over different classes of models (data trees, directed acyclic graphs, and graphs).

I.2.2 Expressive power

The classical result of definability for first-order logic was adapted to the context of many modal logics, where the notion of *isomorphism* is replaced by the weaker concept of *bisimulation* (the one which turns to be adequate for the chosen modal logic). Thus, definability theorems were established for the basic modal logic [30], for temporal logics with *since* and *until* operators [72], for negation-free modal languages [73], etc. A global counterpart was studied in [31], and a general framework stating sufficient conditions for an arbitrary (modal) logic $\mathcal{L}$ to verify it was given in [6]. One of those requirements is that the models of $\mathcal{L}$ are closed under ultraproducts, which is true for the aforementioned logics, but not for $\mathrm{XPath}_{=}$. Indeed, models of $\mathrm{XPath}_{=}$ are data trees, which may not remain connected under ultraproducts. Hence one cannot expect to use that framework in this case. The Separation theorem for the basic modal logic was shown by de Rijke in [30], and it was studied for other specific modal logics such as the temporal logic [72]. For more general modal logics, Separation was studied in [6], but again, $\mathrm{XPath}_{=}$ does not fit in here.

As mentioned before, to prove definability and separation results for node and path expressions of $\mathrm{XPath}_{=}$, we need to change or refine many of the tools used in the corresponding results of basic modal logic. For instance, in our case we have to adjust the notion of ultraproduct in order to always remain in the domain of data trees, and so we work with a variant of it called *quasi-ultraproduct*. We also need to devise suitable notions of saturation. Using these specially developed tools, we arrive to definability and separation theorems for node and path expressions over both the downward fragmentd $\mathrm{XPath}_{=}(\downarrow)$ and the vertical fragment $\mathrm{XPath}_{=}(\uparrow\downarrow)$. These tools and results are presented in Chapter 1 of this thesis, and can also be found in [2] and, more completely, in [5].

I.2.3 Axiomatizations

Given a logic, a sound and complete axiomatization of a class of models consists of a set of axioms whose respective theorems are truths in all models of the class, and such that all universal truths in the class are theorems. There exist axiomatizations for purely navigational fragments of XPath with different axes [102], axiomatizations of other fragments of Core-XPath have been investigated in [13], and extensions with XPath 2.0 features have been addressed in [103]. However, the only other research into the proof theory of $\mathrm{XPath}_{=}$ outside of this thesis has been for a simple fragment [10].

In the framework of equational logic, we give a sound and complete axiomatization for $\mathrm{XPath}_{=}(\downarrow)$ and (first) for an easier subfragment that is unable to express data tests of the form $\langle \alpha \neq \beta \rangle$. All axioms are of the form $\varphi \equiv \psi$ for node expressions φ, ψ or of the

form $\alpha \equiv \beta$ for path expressions α, β, and inference rules are the standard of equational logic. As usual, proving completeness is the hard part. This proof relies in a normal form theorem for expressions in $\mathrm{XPath}_{=}(\downarrow)$, and in the construction of a canonical model for any consistent formula in normal form. These results from [3] can be found in Chapter 2 of this thesis.

I.2.4 Relation between data logics and counter systems

In the realms of semistructured data, finite data trees have been considered as simple abstractions of XML documents, timed automata, program verification, and generally of systems manipulating data values. Therefore, developing logics over data trees whose satisfiability problem is decidable is of great importance when reasoning on data-driven systems. A wealth of specification formalisms on these structures (either for data trees or its 'word' version, data words) have been introduced, stemming from automata [88, 99], first-order logic [17, 65, 48, 19], $\mathrm{XPath}_{=}$ [66, 50, 44, 43, 47], or temporal logics [38, 79, 70, 46, 36, 67]. In their full generality, most formalisms lead to undecidable reasoning problems. As we mentioned in the prologue of this chapter, there are entangled interactions between expressiveness and decidability plus algorithmic complexity, and a well-known research trend consists of finding a good trade-off between them.

In certain aspects, our work is a generalization to ranked (multi)data trees of the work done in [35] for data words. We present a simple data-aware Logic of Repeating Values LRV, similar in many aspects to $\mathrm{XPath}_{=}$, and show how the satisfiability problem of the subfragment $\mathrm{LRV}^{\mathrm{D}}$ over k-ranked data trees can be reduced to the control-state reachability problem for VASS_k. Afterwards we introduce an extension of Branching VASS, called Merging VASS, and show that the satisfiability of the full LRV over k-ranked data trees can be reduced to control-state reachability of MVASS_k. These topics and results are explored in [4], and here we present them in Chapter 4.

I.3 Organization

The main body of this thesis is divided into two parts, each one containing two chapters.

Part A, composed of Chapter 1 and Chapter 2, deals with model-theoretical and proof-theoretical aspects of $\mathrm{XPath}_{=}$. In Chapter 1 we give definability and separation results for $\mathrm{XPath}_{=}(\downarrow)$ and $\mathrm{XPath}_{=}(\uparrow\downarrow)$-node expressions, extend previous model-theoretical results such as van Benthem-style characterization to the context of binary bisimulations for two-pointed data trees, and give definability and separation theorems for $\mathrm{XPath}_{=}(\downarrow)$ and $\mathrm{XPath}_{=}(\uparrow\downarrow)$-path expressions. In Chapter 2 we first give a sound and complete axiomatization for $\mathrm{XPath}_{=}(\downarrow)^{-}$, the subfragment of $\mathrm{XPath}_{=}(\downarrow)$ which lacks inequality data tests of the type $\langle \alpha \neq \beta \rangle$; afterwards we give a sound and complete axiomatization for the full fragment $\mathrm{XPath}_{=}(\downarrow)$.

Part B, composed of Chapter 3 and Chapter 4, deals with more computational aspects of $\mathrm{XPath}_{=}$ and the related logic LRV. In Chapter 3 we extend definitions and results of

$\text{XPath}_=$ over data trees to the wider framework of data graphs, and calculate tight bounds on the complexity classes of the bisimilarity problems for various fragments of $\text{XPath}_=$ and for various restrictions on the models. In Chapter 4 we obtain an algorithmic complexity bound for the satisfiability problem of the fragment LRV^{D} over k-ranked data trees by reducing it to the control-state reachability problem of VASS_k, and, after introducing an extension of Branching VASS called Merging VASS, we show that the satisfiability problem of LRV can be reduced to the control-state reachability problem of MVASS_k.

We end the thesis with the conclusions, where we summarize our results and mention possible avenues of further research.

Part A

Model theory and proof theory

Chapter 1

Definability and binary bisimulation

> Still, the creatures covered the territory in careful increments, moving back and forth along parallel, invisible paths.
>
> *Blindsight*
> Peter Watts

1.1 Introduction

In this chapter we focus on studying the model theory and the expressive power of the logics $\mathrm{XPath}_{=}(\downarrow)$ and $\mathrm{XPath}_{=}(\uparrow\downarrow)$, both for node expressions and path expressions. Our main aim is to give *definability theorems* for these logics: on one hand, necessary and sufficient conditions under which we can assure that classes of pointed data trees can be defined by the use of a single node expression or a set of them; on the other hand, necessary and sufficient conditions under which classes of two-pointed data trees are definable by a single path expression or a set of them. As a consequence of these results, we obtain *separation theorems*, which indicate necessary and sufficient conditions for two classes of pointed (or two-pointed) data trees to be separable by a third class that is definable by a single node expression (respectively, path expression) or set of them.

Though our research on $\mathrm{XPath}_{=}$ takes as a motivation the current relevance of XML documents (which of course are finite) and the logics for reasoning over them, we do not restrict ourselves to the finite case. Indeed, an infinite set of node or path expressions may force all of its models to be infinite. Hence, since we aim at working with arbitrary sets of node or path expressions, we must consider arbitrary (i.e. finite or infinite) data trees.

In the context of BML, definability theorems use two basic tools: ultraproducts and bisimulations. As a first step of our adaptation of these theorems into $\mathrm{XPath}_{=}$, we need to modify the concept of ultraproduct so that its application remains in the universe of data trees. And while the notion of bisimulation has already been developed and studied for the

case of pointed data trees, we need to develop appropriate notions of *binary bisimulation* for the case of two-pointed data trees and path expressions of $\mathrm{XPath}_{=}(\downarrow)$ and $\mathrm{XPath}_{=}(\uparrow\downarrow)$, capturing the notion of logical indistinguishability in the respective fragments and over finitely branching data trees. Our definitions of binary bisimulation are more complex than those of unary bisimulation, and actually the notion of binary bisimilarity subsumes that of unary bisimilarity.

For this binary framework, we also show a van Benthem-style characterization theorem paralleling that for unary bisimulation, and show that a first-order formula with two free variables is expressible in $\mathrm{XPath}_{=}(\downarrow)$ if and only if it is binary-bisimulation-invariant and represents a 'forward property'. As in the unary case, the characterization fails for $\mathrm{XPath}_{=}(\uparrow\downarrow)$.

1.1.1 Related work

The notion of bisimulation was introduced independently by van Benthem [104] in the context of modal correspondence theory, by Milner [84] and Park [92] in concurrency theory, and by Forti and Honsell [53] in non-wellfounded set theory (see [97] for a historical outlook). With respect to notions of binary bisimulations, we can mention the work [52], where some notions of bisimulations are given for some fragments of Tarski's calculus of binary relations, with the aim of understanding the expressive power of the calculus of relations as a database query language for binary relation structures.

The classical result of definability for first-order logic has been adapted to the context of many modal logics, where the notion of *isomorphism* is replaced by the weaker concept of *bisimulation* (the one which turns to be adequate for the chosen modal logic). Thus, definability theorems were established for the basic modal logic [30], for temporal logics with *since* and *until* operators [72], for negation-free modal languages [73], etc. A global counterpart was studied in [31], and a general framework stating sufficient conditions for an arbitrary (modal) logic $\mathcal{L}$ to verify it was given in [6]. One of those requirements is that the models of $\mathcal{L}$ are closed under ultraproducts, which is true for the aforementioned logics, but not for $\mathrm{XPath}_{=}$. Indeed, models of $\mathrm{XPath}_{=}$ are data trees, which may not remain connected under ultraproducts. Hence one cannot expect to use that framework in this case. The Separation theorem for the basic modal logic was shown by de Rijke in [30], and it was studied for other specific modal logics such as the temporal logic [72]. For more general modal logics, Separation was studied in [6], but again, $\mathrm{XPath}_{=}$ does not fit in that framework.

In [104] basic modal logic is characterized as the bisimulation invariant fragment of first-order logic. Van Benthem's original result over arbitrary structures was proved to hold for finite structures by Rosen [95]. The proof was then simplified and unified by Otto [89, 90], and later expanded by Dawar and Otto [29] to other classes of structures. We follow the ideas of [89] to show the characterization result for binary bisimulations in the downward fragment.

There are many works in the literature studying the expressive power of Core-XPath

(see e.g. [59, 82, 101]). All these consider the navigational (i.e. data-oblivious) fragment of XPath. A first step towards the study of the expressive power of XPath when equipped with (in)equality test over data trees is the paper [44], and its full version [45]. The developments in this chapter are a natural continuation of that work.

In [45], the expressive power of $\mathrm{XPath}_{=}$ was studied, from a logical and modal model theoretical point of view. Suitable notions of bisimulations were given both for $\mathrm{XPath}_{=}(\downarrow)$ and $\mathrm{XPath}_{=}(\uparrow\downarrow)$. There it is shown that if x and x' are bisimilar then they satisfy exactly the same node expressions, and that the converse is also true for trees whose every node has only finitely many children. Hence, bisimulation coincides with logical equivalence, i.e., with *indistinguishability by means of node expressions*. A van Benthem-like characterization theorem is also given for the downward fragment of $\mathrm{XPath}_{=}$, which states that it coincides with the bisimulation-invariant fragment of first-order logic with one free variable (over the adequate signature). For the case of the vertical fragment of $\mathrm{XPath}_{=}$ this characterization fails.

This chapter contains a natural continuation of [45], as we develop new tools and delve in some aspects of the model theory of the downward and vertical fragments of $\mathrm{XPath}_{=}$, studying their expressive power regarding both node expressions and path expressions. Our main result regarding node expressions is the analog of the classic BML definability theorem mentioned in the introduction of this thesis:

> *A class of pointed models K is definable by means of a set of basic modal formulas if and only if K is closed under ultraproducts and bisimulations, and the complement of K is closed under ultrapowers. Also, K is definable by a single basic modal formula if and only if both K and its complement are closed under ultraproducts and bisimulations.*

As a corollary, we obtain the analog of the separation theorem, which says:

> *Let K_1 and K_2 be two disjoint classes of pointed models such that K_1 is closed under bisimulations and ultraproducts and K_2 is closed under bisimulations and ultrapowers. Then there exists a class K that is definable by a set of basic modal formulas, contains K_1, and is disjoint from K_2. Furthermore, if both K_1 and K_2 are closed under bisimulations and ultraproducts, then such K is definable by a single basic modal formulas.*

1.1.2 Contributions

In the first part of this chapter we study the expressive power of node expressions for the downward and vertical fragments of $\mathrm{XPath}_{=}$. We show definability theorems (which provide conditions under which a class of pointed data trees can be defined by a node expression or by a set of node expressions) and separation theorems (which provide conditions under which two disjoint classes of pointed data trees can be separated by a class definable by a node expression or a set of node expressions).

Our definability and separation theorems for XPath$_=$ themselves are shown using rather known techniques. Our main contribution lies in devising and calibrating the corresponding notions to be used in the XPath$_=$ setting, and in studying the subtle interaction between them:

- *Bisimulation*: already introduced in [44], it is the counterpart of *isomorphisms* in the classical theorem for first-order logic. It is known that if two (possibly infinite) data trees are bisimilar then they are logically equivalent (that is, they are not distinguishable by an XPath$_=$-node expression) but that the converse is not true in general. If the trees are finitely branching, then bisimilarity and logical indistinguishability do coincide.

- *Saturation*: we define and study the new notion of *XPath$_=$-saturation*. We show that for XPath$_=$-saturated data trees, being bisimilar is the same as being logically equivalent. It is also shown that a 2-saturated data tree (regarded as a first-order structure) is already XPath$_=$-saturated.

- *Ultraproducts*: contrary to other adaptations of the classical first-order definability theorem to modal logics, in our case we have to adjust also the notion of *ultraproduct*, and so we work with a variant of it called *quasi-ultraproduct*. The reason is that we must not abandon the universe of data trees, as these are the only allowed models of XPath$_=$.

In the second part of this chapter we start a model-theoretical study of path expressions of XPath$_=$. We introduce a new kind of *binary* bisimulation for both the downward and the vertical fragment, which captures, over finitely branching trees, when two pairs of nodes (instead of single nodes, as in [44]) are indistinguishable by means of path expressions (instead of by node expressions). Our binary bisimulations subsume, in fact, the already known unary bisimulation, since over finitely branching trees, (x, x) is binary-bisimilar to (x', x') if and only if x is unary-bisimilar to x'. The definitions of binary bisimulations require more rules than the unary ones, but they all have the flavor of back-and-forth conditions.

We also show a characterization theorem, which states that a first-order formula with two free variables is expressible in XPath$_=(\downarrow)$ if and only if it is binary-bisimulation-invariant and represents a 'forward property'. Using the new tool of binary bisimulations, together with suitable modifications of saturation, we show definability and separation theorems, this time in the context of path expressions as the language of description, and with respect to classes of two-pointed data trees (with some restrictions that allow the expressibility of complementation and intersection of path expressions).

1.1.3 Organization

This chapter is organized as follows. In §1.2 we give some preliminaries that were ommited from the introduction of the thesis but are of great importance for this chapter, such

as notions of bounded bisimulation, normal forms, and ultraproducts. In §1.3.1 we give suitable notions of saturation for the downward and vertical fragments of $\mathrm{XPath}_{=}$, and show that for saturated trees bisimilarity coincides with logical equivalence. In §1.3.2 we expand the connection between $\mathrm{XPath}_{=}$ and first-order logic, and we introduce the idea of quasi-ultraproducts for the downward and vertical fragments. In §1.3.3 and §1.3.4 we state the theorems on definability and separation, respectively.

In §1.4 we start our study of path expressions, which is divided into the downward fragment (§1.4.1) and the vertical fragment (§1.4.2). For the downward fragment, we begin with some needed facts and we then define the notions of logical equivalence to be used. The definitions of binary bisimulations for the downward fragment are also given here, where it is also shown their coincidence to the logical equivalence for path expressions. Afterwards we give a characterization theorem for binary bisimulations in $\mathrm{XPath}_{=}(\downarrow)$. For the vertical fragment, we first show some needed facts and then introduce the definition of binary bisimulation, where, again, it is shown that it matches logical equivalence. In §1.5 we introduce the needed changes to the notions of saturation and quasi-ultraproducts for the case of two-pointed data trees, and we state for this scenario of path expressions the theorems of definability and separation, over some restricted classes of two-pointed data trees where we can express the concepts of complementation and intersection of path expressions.

1.2 Preliminaries

We begin this section presenting some results from [45]: notions of bounded bisimulation for $\mathrm{XPath}_{=}(\downarrow)$ and $\mathrm{XPath}_{=}(\uparrow\downarrow)$, and the theorems that relate the notions of bisimulation on these fragments with the corresponding notions of logical indistinguishability. Then we state some normal form theorems for $\mathrm{XPath}_{=}(\downarrow)$ and $\mathrm{XPath}_{=}(\uparrow\downarrow)$, mostly taken from [45], which allow us to deal with restricted syntactical fragments that are just as expressive as the full fragments, but that greatly simplify many of our proofs. Finally, we show the basic definitions and results for ultraproducts, which are a fundamental tool for our definability and separation theorems.

1.2.1 Bounded notions of bisimulation

ℓ-bisimulation

The notion of $\mathrm{XPath}_{=}(\downarrow)$-bisimilarity might be too strong for some purposes, as it may distinguish two nodes that only fail the bisimilarity test with a pair of paths of high length. In real-world applications, queries are rarely very deep, and it might be useful to consider two nodes as 'alike' when they merely are bisimilar just 'up to a certain depth' (see [58] for some exploration of this topic). ℓ-bisimulation is a possible answer for these types of utilization, being a bisimulation notion that can only 'see' up to a certain fixed depth from the starting nodes.

Given two data trees $\mathcal{T}$ and $\mathcal{T}'$, with respective sets of nodes T, T', we say that a family of relations $(Z_j)_{0\leq j\leq \ell}$ in $T \times T'$ constitutes an **ℓ-bisimulation** if for each $j \leq \ell$ and $(x, x') \in Z_j$ the following conditions hold:

- (**Harmony**) $label(x) = label(x')$
- (**Zig**) If $x \xrightarrow{m} v$ and $x \xrightarrow{n} w$ with $m, n \leq j$ then there are $v', w' \in T'$ such that $x' \xrightarrow{m} v'$, $x' \xrightarrow{n} w'$ and
  - $(\xrightarrow{i} v)\, Z_{j-m+i}\, (\xrightarrow{i} v')$ for all $0 \leq i < m$,
  - $(\xrightarrow{i} w)\, Z_{j-n+i}\, (\xrightarrow{i} w')$ for all $0 \leq i < n$, and
  - $data(v) = data(w)$ iff $data(v') = data(w')$.
- (**Zag**) (The symmetric condition to **Zig**, starting in T').

In Figure 13 we show an example of the process of checking **Zig** for two nodes.

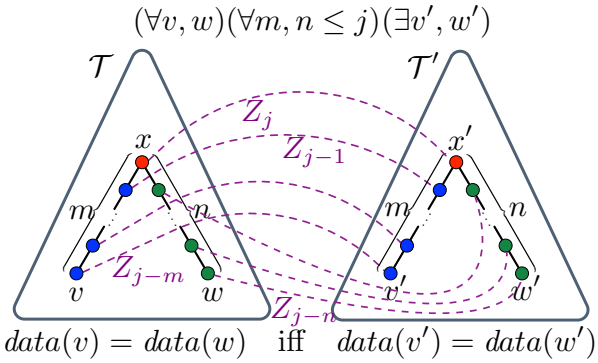


Figure 13: The process of checking the condition **Zig** of ℓ-bisimulation for $j \leq \ell$ and $(x, x') \in Z_j$.

We define when two nodes $x \in T$ and $x' \in T'$ are said to be **ℓ-bisimilar**, notated $\mathcal{T}, x \leftrightarrows^{\downarrow}_{\ell} \mathcal{T}', x'$:

$$\mathcal{T}, x \leftrightarrows^{\downarrow}_{\ell} \mathcal{T}', x' \overset{\text{def}}{\Leftrightarrow} \text{ there is an } \ell\text{-bisimulation } (Z_j)_{0\leq j\leq \ell} \text{ such that } (x, x') \in Z_\ell.$$

Observe that if Z is an XPath$_=(\downarrow)$-bisimulation, then $(Z_j)_{0\leq j\leq \ell}$, with $Z_j = Z$ for all j, is an ℓ-bisimulation for any ℓ. Thus, $\mathcal{T}, x \leftrightarrows^{\downarrow} \mathcal{T}', x'$ implies that $\mathcal{T}, x \leftrightarrows^{\downarrow}_{\ell} \mathcal{T}', x'$ for all ℓ.

(r,s,k)-bisimulation

As XPath$_=(\downarrow)$ has a 'bounded' version of bisimulation with ℓ-bisimulation, so does the logic XPath$_=(\uparrow\downarrow)$. In this case, however, the definition is quite more involved.

Given two data trees $\mathcal{T}$ and $\mathcal{T}'$, with respective sets of nodes T, T', we say that a family of relations $(Z^{\hat{k}}_{\hat{r},\hat{s}})_{\hat{r}+\hat{s}\leq r+s, \hat{k}\leq k}$ in $T \times T'$ constitutes an **(r, s, k)-bisimulation** if for all $\hat{r} + \hat{s} \leq r + s, \hat{k} \leq k$, when $(x, x') \in Z^{\hat{k}}_{\hat{r},\hat{s}}$ the following conditions hold:

- (**Harmony**) $label(x) = label(x')$
- (**Zig**) If $v \xrightarrow{m} x$ and $v \xrightarrow{n} w$ with $m \leq \hat{s}$ and $n \leq \hat{r} + m$ then there are $v', w' \in T'$ such that $v' \xrightarrow{m} x'$, $v' \xrightarrow{n} w'$, and the following hold:
  - if $\hat{k} > 0$, $(w, w') \in Z^{\hat{k}-1}_{\hat{r}',\hat{s}'}$ for $\hat{r}' = \hat{r} + m - n$, $\hat{s}' = \hat{s} - m + n$.
  - $data(w) = data(x)$ iff $data(w') = data(x')$,
- (**Zag**) (The symmetric condition to **Zig**, starting in T').

In Figure 14 we show an example of the process of checking **Zig** for two nodes.

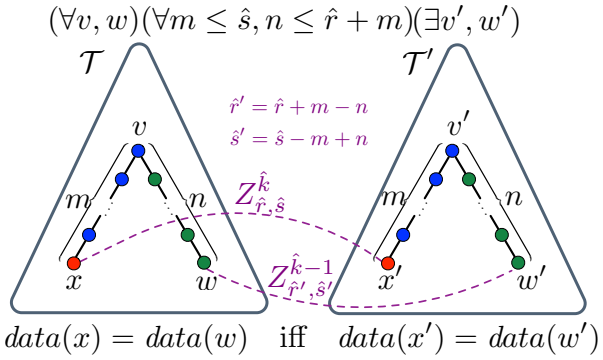


Figure 14: The process of checking the condition **Zig** of (r, s, k)-bisimulation for $\hat{r} + \hat{s} \leq r + s$, $\hat{k} \leq k$, and $(x, x') \in Z^{\hat{k}}_{\hat{r},\hat{s}}$.

We define when two nodes $x \in T$ and $x' \in T'$ are said to be **(r, s, k)-bisimilar**, notated $\mathcal{T}, x \leftrightarrows^{\uparrow\downarrow}_{r,s,k} \mathcal{T}', x'$:

$$\mathcal{T}, x \leftrightarrows^{\uparrow\downarrow}_{r,s,k} \mathcal{T}', x' \overset{\text{def}}{\Leftrightarrow} \text{ there is an } (r,s,k)\text{-bisimulation } (Z^{\hat{k}}_{\hat{r},\hat{s}})_{\hat{r}+\hat{s}\leq r+s,\hat{k}\leq k} \text{ s.t. } (x, x') \in Z^{k}_{r,s}$$

1.2.2 Bounded notions of equivalence

In order to define the appropriate logical fragment for $\leftrightarrows^{\downarrow}_{\ell}$ and $\leftrightarrows^{\uparrow\downarrow}_{r,s,k}$, we first have to provide some definitions.

Given a XPath$_=(\downarrow)$-node expression φ, we write $\mathsf{dd}(\varphi)$ to denote the **downward depth** of φ, which measures 'how deep' the formula can see, and it is defined as follows:

$$\begin{array}{rclcrcl}
\mathsf{dd}(a) & = & 0 & & \mathsf{dd}(\lambda) & = & 0 \\
\mathsf{dd}(\varphi \wedge \psi) & = & \max\{\mathsf{dd}(\varphi), \mathsf{dd}(\psi)\} & & \mathsf{dd}(\varepsilon\alpha) & = & \mathsf{dd}(\alpha) \\
\mathsf{dd}(\neg\varphi) & = & \mathsf{dd}(\varphi) & & \mathsf{dd}([\varphi]\alpha) & = & \max\{\mathsf{dd}(\varphi), \mathsf{dd}(\alpha)\} \\
\mathsf{dd}(\langle\alpha\rangle) & = & \mathsf{dd}(\alpha) & & \mathsf{dd}(\downarrow\alpha) & = & 1 + \mathsf{dd}(\alpha) \\
\mathsf{dd}(\langle\alpha \odot \beta\rangle) & = & \max\{\mathsf{dd}(\alpha), \mathsf{dd}(\beta)\} & & \mathsf{dd}((\alpha \cup \beta)\gamma) & = & \max\{\mathsf{dd}(\alpha\gamma), \mathsf{dd}(\beta\gamma)\}
\end{array}$$

where $a \in \mathbb{A}$, $\odot \in \{=, \neq\}$, and α is any path expression or the empty string λ. We remark that we have a definition for dd including the union $\cup$ of path expressions, as we said in §I.1.2 that the union is only expressively unessential for node expressions.

Example 5. Let $\varphi = a \wedge \langle\downarrow = \downarrow\downarrow\rangle$, $\alpha = \downarrow[\varphi]$, and $\beta = \downarrow[\varphi]\downarrow$. Then $\mathsf{dd}(\varphi) = 2$, and $\mathsf{dd}(\alpha) = \mathsf{dd}(\beta) = 3$.

We name as **ℓ-XPath$_=(\downarrow)$** the fragment of XPath$_=(\downarrow)$ consisting of all node expressions φ such that $\mathsf{dd}(\varphi) \leq \ell$. We say that $\mathcal{T}, x$ and $\mathcal{T}', x'$ are **ℓ-equivalent for XPath$_=(\downarrow)$** (notated $\mathcal{T}, x \equiv_\ell^\downarrow \mathcal{T}', x'$) iff the truth value of any node expression $\varphi \in \ell$-XPath$_=(\downarrow)$ coincides over both pointed data trees. That is:

$$\mathcal{T}, x \equiv_\ell^\downarrow \mathcal{T}', x' \overset{\text{def}}{\Leftrightarrow} \text{ for all } \ell\text{-XPath}_=(\downarrow)\text{-node expressions } \varphi,\ \mathcal{T}, x \models \varphi \text{ iff } \mathcal{T}', x' \models \varphi.$$

For the vertical fragment XPath$_=(\uparrow\downarrow)$, we need to define both the maximum downward distance r and the maximum upward distance s that the formula can reach. We call the pair (r, s) the **vertical depth** of a formula, notated $\mathsf{vd}(\varphi)$. The **nesting depth** of a formula φ, notated $\mathsf{nd}(\varphi)$, is the maximum number of nested [ ] appearing in φ. More precisely:

$$\begin{array}{rcl@{\qquad}rcl}
\mathsf{vd}(a) & = & (0,0) & \mathsf{vd}(\lambda) & = & (0,0) \\
\mathsf{vd}(\varphi \wedge \psi) & = & \max\{\mathsf{vd}(\varphi), \mathsf{vd}(\psi)\} & \mathsf{vd}(\varepsilon\alpha) & = & \mathsf{vd}(\alpha) \\
\mathsf{vd}(\neg\varphi) & = & \mathsf{vd}(\varphi) & \mathsf{vd}([\varphi]\alpha) & = & \max\{\mathsf{vd}(\varphi), \mathsf{vd}(\alpha)\} \\
\mathsf{vd}(\langle\alpha\rangle) & = & \mathsf{vd}(\alpha) & \mathsf{vd}(\downarrow\alpha) & = & \max\{(0,0), \mathsf{vd}(\alpha) + (1,-1)\} \\
\mathsf{vd}(\langle\alpha \odot \beta\rangle) & = & \max\{\mathsf{vd}(\alpha), \mathsf{vd}(\beta)\} & \mathsf{vd}(\uparrow\alpha) & = & \max\{(0,0), \mathsf{vd}(\alpha) + (-1,1)\} \\
\mathsf{vd}((\alpha \cup \beta)\gamma) & = & \max\{\mathsf{vd}(\alpha\gamma), \mathsf{vd}(\beta\gamma)\} & & &
\end{array}$$

$$\begin{array}{rcl@{\qquad}rcl}
\mathsf{nd}(a) & = & 0 & \mathsf{nd}(\alpha\beta) & = & \max\{\mathsf{nd}(\alpha), \mathsf{nd}(\beta)\} \\
\mathsf{nd}(\varphi \wedge \psi) & = & \max\{\mathsf{nd}(\varphi), \mathsf{nd}(\psi)\} & \mathsf{nd}(\varepsilon) & = & 0 \\
\mathsf{nd}(\neg\varphi) & = & \mathsf{nd}(\varphi) & \mathsf{nd}([\varphi]) & = & 1 + \mathsf{nd}(\varphi) \\
\mathsf{nd}(\langle\alpha\rangle) & = & \mathsf{nd}(\alpha) & \mathsf{nd}(\downarrow) & = & 0 \\
\mathsf{nd}(\langle\alpha \odot \beta\rangle) & = & \max\{\mathsf{nd}(\alpha), \mathsf{nd}(\beta)\} & \mathsf{nd}(\uparrow) & = & 0 \\
\mathsf{nd}((\alpha \cup \beta)\gamma) & = & \max\{\mathsf{nd}(\alpha\gamma), \mathsf{nd}(\beta\gamma)\} & & &
\end{array}$$

where $a \in \mathbb{A}$, $\odot \in \{=, \neq\}$, the operations '+' and 'max' are performed component-wise, and α is any path expression or the empty string λ.

Example 6. Let $\alpha = \uparrow\downarrow\downarrow$, $\varphi = \langle\uparrow[a]\downarrow\downarrow \neq \downarrow\uparrow\uparrow\uparrow\rangle$, $\psi = \neg\langle[a] = \downarrow[\langle\uparrow \neq [b]\rangle]\rangle$. Then: $\mathsf{vd}(\alpha) = (1,1)$ and $\mathsf{nd}(\alpha) = 0$, $\mathsf{vd}(\varphi) = (1,2)$ and $\mathsf{nd}(\varphi) = 1$, and $\mathsf{vd}(\psi) = (1,0)$ and $\mathsf{nd}(\psi) = 2$.

Now, let **(r, s, k)-XPath$_=(\uparrow\downarrow)$** be defined as the set of XPath$_=(\uparrow\downarrow)$ node expressions φ such that $\mathsf{vd}(\varphi) \leq (r, s)$ and $\mathsf{nd}(\varphi) \leq k$. We say that two pointed data trees $\mathcal{T}, x$ and $\mathcal{T}', x'$ are **(r, s, k)-equivalent for XPath$_=(\uparrow\downarrow)$** (notated $\mathcal{T}, x \equiv_{r,s,k}^{\uparrow\downarrow} \mathcal{T}', x'$) iff the truth value of any node expression $\varphi \in (r, s, k)$-XPath$_=(\uparrow\downarrow)$ coincides over both pointed data trees. That is:

$$\mathcal{T}, x \equiv^{\uparrow\downarrow}_{r,s,k} \mathcal{T}', x' \quad \overset{\text{def}}{\Leftrightarrow} \quad \begin{array}{l}\text{for all } (r,s,k)\text{-XPath}_{=}(\uparrow\downarrow)\text{-node expressions } \varphi,\\ \mathcal{T}, x \models \varphi \text{ iff} \mathcal{T}', x' \models \varphi.\end{array}$$

We now state the full theorems from [45], already partially mentioned in §I.1.3, which show the connection between bisimulation and equivalence for the downward and for the vertical fragments of XPath$_{=}$.

Theorem 7. *[45] Given two pointed data trees $\mathcal{T}, x$ and $\mathcal{T}', x'$,*

1. *$\mathcal{T}, x \underline{\leftrightarrow}^{\downarrow} \mathcal{T}', x'$ implies $\mathcal{T}, x \equiv^{\downarrow} \mathcal{T}', x'$. The converse is not true in general, but it holds when $\mathcal{T}$ and $\mathcal{T}'$ are finitely branching.*
2. *$\mathcal{T}, x \underline{\leftrightarrow}^{\downarrow}_{\ell} \mathcal{T}', x'$ iff $\mathcal{T}, x \equiv^{\downarrow}_{\ell} \mathcal{T}', x'$.*

Theorem 8. *[45] Given two pointed data trees $\mathcal{T}, x$ and $\mathcal{T}', x'$, $\mathcal{T}, x \underline{\rightarrow}^{\downarrow} \mathcal{T}', x'$ implies $\mathcal{T}, x \Rightarrow^{\downarrow} \mathcal{T}', x'$. The converse is not true in general, but it holds when $\mathcal{T}$ and $\mathcal{T}'$ are finitely branching.*

Theorem 9. *[45] Given two pointed data trees $\mathcal{T}, x$ and $\mathcal{T}', x'$,*

1. *$\mathcal{T}, x \underline{\leftrightarrow}^{\uparrow\downarrow} \mathcal{T}', x'$ implies $\mathcal{T}, x \equiv^{\uparrow\downarrow} \mathcal{T}', x'$. The converse is not true in general, but it holds when $\mathcal{T}$ and $\mathcal{T}'$ are finitely branching.*
2. *$\mathcal{T}, x \underline{\leftrightarrow}^{\uparrow\downarrow}_{r,s,k\cdot(r+s+2)} \mathcal{T}', x'$ implies $\mathcal{T}, x \equiv^{\uparrow\downarrow}_{r,s,k} \mathcal{T}', x'$.*
3. *$\mathcal{T}, x \equiv^{\uparrow\downarrow}_{r,s,k} \mathcal{T}', x'$ implies $\mathcal{T}, x \underline{\leftrightarrow}^{\uparrow\downarrow}_{r,s,k} \mathcal{T}', x'$.*

1.2.3 Normal forms

The normal form of a node or path expression is a standard or canonical way to represent it; a *normal form theorem* states that any expression has an equivalent[10] expression that is in normal form. If normal forms are chosen adequately, they serve to simplify proofs and definitions. Importantly, the notion of XPath$_{=}(\uparrow\downarrow)$-bisimilarity we have given in §I.1.2, and its equivalence with logical indistinguishability as stated in Theorem 9, rely on the normal form theorem we will soon show.

To begin with, let us observe that a path expression in XPath$_{=}(\downarrow)$ could be of the type $\varepsilon\varepsilon\downarrow[\varphi][\psi][\eta]\varepsilon\downarrow$, but that this representation is 'wasteful' in a certain sense; there is another, more concise path expression which has the same semantics: $\downarrow[\varphi \wedge \psi \wedge \eta]\downarrow$. In general, for any data tree $\mathcal{T}$ and any path expression α, $[\![\alpha]\!]^{\mathcal{T}} = [\![\alpha\varepsilon]\!]^{\mathcal{T}}$, and given node expressions φ and ψ, we also have that $[\![\downarrow[\varphi][\psi]]\!]^{\mathcal{T}} = [\![\downarrow[\varphi \wedge \psi]]\!]^{\mathcal{T}}$. Indeed, continuing in this direction, it is easy to see that any XPath$_{=}(\downarrow)$-path expression is semantically equivalent to some path expression in the form:

$$[\varphi_0]\downarrow[\varphi_1]\downarrow \ldots \downarrow[\varphi_n]. \tag{13}$$

[10] In this chapter we deal with normal forms in a semantical context, that is, we speak about equivalences in respect of the universe of all models, but in Chapter 2 we instead work with normal forms from a proof-theoretical perspective.

An XPath$_=(\downarrow)$-path expression is said to be in **XPath$_=(\downarrow)$-normal form** if it is written in the form of (13), and every other path expression contained in it (such as, e.g., path expressions appearing in the node expressions φ_i) is also in that form. Similarly, an XPath$_=(\downarrow)$-node expression is in XPath$_=(\downarrow)$-normal form if all path expressions that appear in it are in XPath$_=(\downarrow)$-normal form. It can be seen that any XPath$_=(\downarrow)$-formula is equivalent to a formula in XPath$_=(\downarrow)$-normal form.

We could use some of the insights of the case of XPath$_=(\downarrow)$-normal form in order to tentatively propose $[\varphi_0]o_1[\varphi_1]\dots o_n[\varphi_n]$ as a candidate for the normal form of XPath$_=(\uparrow\downarrow)$-path expressions, where $o_i \in \{\uparrow,\downarrow\}$. Surprisingly, there is an even simpler candidate. Let us say that a path expression α of XPath$_=(\uparrow\downarrow)$ is *downward* [respectively, *upward*], if it is of the form $\downarrow^n[\varphi]$ [resp. $[\varphi]\uparrow^n$]. An *up-down* expression is one of the form $\varepsilon, \alpha^\uparrow, \alpha^\downarrow$, or $\alpha^\uparrow\alpha^\downarrow$, where $\alpha^\uparrow$ is upward and $\alpha^\downarrow$ is downward. We say that a XPath$_=(\uparrow\downarrow)$-formula is in **up-down normal form**, or **XPath$_=(\uparrow\downarrow)$ normal form**, if every path expression contained in it is up-down, and all data tests are of the form $\langle\varepsilon \odot \beta^\uparrow\beta^\downarrow\rangle$ with $\odot \in \{=,\neq\}$.

Why should it be possible to transform any path expression into an equivalent one in up-down normal form? The main idea is to take advantage of the fact that we are working over data trees, and thus the operation $\uparrow$ is deterministic; any node 'higher' in the tree can still be unambiguously accessed if we first go down any number of times and then go up the appropriate amount. For instance, one can see that:

$$\downarrow[\varphi]\uparrow^2[\psi]\downarrow^3[\eta] \text{ is equivalent to } [\langle\uparrow^0\downarrow[\varphi]\rangle]\uparrow\downarrow^3[\eta \wedge \langle\uparrow^3\downarrow^0[\psi]\rangle].$$

In [45] it is proven that any XPath$_=(\uparrow\downarrow)$-formula has an equivalent formula in up-down normal form, which furthermore has its same vertical depth and a nesting depth that is polynomially bounded over the nesting and vertical depth of the original formula. It is with the help of this *normal form theorem* that the connection between XPath$_=(\uparrow\downarrow)$-bisimilarity and XPath$_=(\uparrow\downarrow)$-equivalence is proved.

In §1.4.2 we explicitly state some of the results from [45], and we use them to slightly extend the study of normal forms for path expressions.

1.2.4 Ultraproducts

The ultraproduct of a family of structures is a construction that preserves properties that are true 'in almost all' members of that family. Ultraproducts are a very useful theoretical tool to construct structures with certain properties; an example is the construction of the *hyperreal numbers* as an appropriate ultrapower[11] of the real numbers. In order to properly define ultraproduct, we first need some preliminary definitions:

Let I be a set, and U be a set of subsets of I. We say that U is an *ultrafilter* if it has the following properties:

- $\emptyset \notin U, I \in U$

[11] An ultrapower is an ultraproduct where all the structures of the family are the same.

- If $A, B \in U$, then $A \cap B \in U$
- If $A \in U$ and $A \subseteq B \subseteq I$, then $B \in U$
- For each $A \subseteq I$, either $A \in U$ or $I \setminus A \in U$

Given any infinite set I, an important example of an ultrafilter over I is the *Fréchet filter* given by $\mathcal{F} = \{A \subseteq I \mid I \setminus A \text{ is finite}\}$. We say that an ultrafilter U is *principal* if it is generated by a subset A of I, that is: $U = \{B \subseteq I \mid A \subseteq B\}$.

Now, let $\mathcal{L}$ be a signature, I a set, D a non-principal ultrafilter over I, and $\{\mathcal{M}_i\}_{i\in I}$ a family of $\mathcal{L}$-structures, with the corresponding family of universes $\{M_i\}_{i\in I}$. Given $a = \{a_i\}_{i\in I}$ and $b = \{b_i\}_{i\in I}$ elements of $\prod_{i\in I} M_i = \{z : I \to \bigcup M_i \mid z_i \in M_i \text{ for each } i\}$, we say that a and b are equivalent respect to D, notated $a \sim b$ iff '$a_i = b_i$ almost everywhere respect to D', that is:

$$a \sim b \overset{\text{def}}{\Leftrightarrow} \{i \in I \mid a_i = b_i\} \in D.$$

We denote $a/D \overset{\text{def}}{=} \{b \in \prod_{i\in I} M_i \mid a \sim b\}$, and we call this the *ultralimit* of the family $\{a_i\}_{i\in I}$. We are at last in the conditions to give our desired definition. Let $M = \prod_i M_i/D$ the set of ultralimits of the sets M_i. The **ultraproduct** $\mathcal{M} = \prod_{i\in I} \mathcal{M}_i/D$, also notated

$$\prod_D \mathcal{M}_i,$$

is the $\mathcal{L}$-structure with the following components: The universe of $\mathcal{M}$ is M. Constants are interpreted as the ultralimit of constants in all the structures. Functions over appropriately sized tuples of ultralimits are defined as the ultralimits of the functions applied to the tuples of elements. Predicates over tuples of ultralimits are defined to be true iff the set of indexes where they are true for intervening tuples belongs to D (that is, the interpretation over tuples of the models is D-a.e. true). More precisely, given $a^1/D, \ldots, a^n/D$ ultralimits in M with a^k/D being the ultralimit of $\{a^k_i/D\}_{i\in I}$, we define, for every n-ary predicate symbol R, n-ary function symbol f, and constant symbol c in $\mathcal{L}$:

- $c_{\mathcal{M}}$ is the ultralimit of $\{c_{\mathcal{M}_i}\}_{i\in I}$
- $f_{\mathcal{M}}(a^1/D, \ldots, a^n/D)$ is the ultralimit of $\{f_{\mathcal{M}_i}(a^1_i, \ldots, a^n_i)\}_{i\in I}$
- $(a^1/D, \ldots, a^n/D) \in R_{\mathcal{M}}$ iff $(a^1_i, \ldots, a^n_i) \in R_{\mathcal{M}_i}$ D-a.e.

Finally, let us mention the *fundamental theorem of ultraproducts* (see e.g. [23, Thm. 4.1.9]), which states the aforementioned characteristic the ultraproducts have in the preservation of properties that are true almost everywhere.

Theorem 10 (Fundamental Theorem of Ultraproducts)**.** *Let φ be a $\mathcal{L}$-formula with free variables $x_1, \ldots, x_n$, let $\mathcal{M}_i$ be a family of $\mathcal{L}$-structures and let $\mathcal{M}$ be their ultraproduct, and let a^k_i be elements in M_i and a^k/D their ultralimit. Then:*

$$\mathcal{M} \models \varphi(a^1/D, \ldots, a^n/D) \Leftrightarrow \mathcal{M}_i \models \varphi(a^1_i, \ldots, a^n_i) \text{ } D\text{-a.e.}$$

1.3 Definability via node expressions

In this section we develop and apply the tools needed to obtain the main results of definability and separation via node expressions of XPath$_=(\downarrow)$ and XPath$_=(\uparrow\downarrow)$. While in the case of XPath$_=(\downarrow)$ we work over the full universe of pointed data trees, this framework is inadequate for XPath$_=(\uparrow\downarrow)$; for definability and separation results for XPath$_=(\uparrow\downarrow)$, we mostly work over the universe of k-bounded pointed data trees, which are pointed data trees where the selected nodes are at a distance at most k from the root.

1.3.1 Saturation

In [44] it is shown that the reverse implication of Item 1. of Theorem 7 holds over finitely branching trees. However, it does not hold in general. In this section we introduce notions of saturation for the downward and vertical fragments of XPath, and show that the reverse implication of Theorem 7 is true over saturated data trees.

Saturation for the downward fragment. Let $\langle \Sigma_1, \ldots, \Sigma_n \rangle$ and $\langle \Gamma_1, \ldots, \Gamma_m \rangle$ be tuples of sets of XPath$_=(\downarrow)$-node expressions. Given a data tree $\mathcal{T}$ and $u \in T$, we say that $\langle \Sigma_1, \ldots, \Sigma_n \rangle$ and $\langle \Gamma_1, \ldots, \Gamma_m \rangle$ are $\boldsymbol{=^{\downarrow}_{n,m}}$**-satisfiable** [resp. $\boldsymbol{\neq^{\downarrow}_{n,m}}$**-satisfiable**] at $\mathcal{T}, u$ if there exist $v_0 \to v_1 \to \cdots \to v_n \in T$ and $w_0 \to w_1 \to \cdots \to w_m \in T$ such that $u = v_0 = w_0$ and

1. for all $i \in \{1, \ldots, n\}$, $\mathcal{T}, v_i \models \Sigma_i$;
2. for all $j \in \{1, \ldots, m\}$, $\mathcal{T}, w_j \models \Gamma_j$; and
3. $data(v_n) = data(w_m)$ [resp. $data(v_n) \neq data(w_m)$].

In Figure 15 we show an example of two tuples of sets of node expressions that are $=^{\downarrow}_{n,m}$-satisfiable at a pointed data tree $\mathcal{T}, x$.

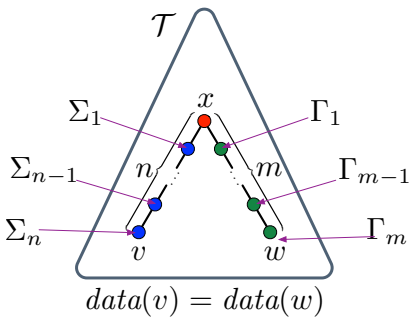


Figure 15: $\langle \Sigma_1, \ldots, \Sigma_n \rangle$ and $\langle \Gamma_1, \ldots, \Gamma_m \rangle$ are $=^{\downarrow}_{n,m}$-satisfiable at $\mathcal{T}, x$.

We say that $\langle \Sigma_1, \ldots, \Sigma_n \rangle$ and $\langle \Gamma_1, \ldots, \Gamma_m \rangle$ are $\mathbf{=^{\downarrow}_{n,m}}$**-finitely satisfiable** [resp. $\mathbf{\neq^{\downarrow}_{n,m}}$**-finitely satisfiable**] at $\mathcal{T}, u$ if for every finite $\Sigma'_i \subseteq \Sigma_i$ and finite $\Gamma'_j \subseteq \Gamma_j$, we have that $\langle \Sigma'_1, \ldots, \Sigma'_n \rangle$ and $\langle \Gamma'_1, \ldots, \Gamma'_m \rangle$ are $=^{\downarrow}_{n,m}$-satisfiable [resp. $\neq^{\downarrow}_{n,m}$-satisfiable] at $\mathcal{T}, u$.

Definition 11. We say that a data tree $\mathcal{T}$ is **$\downarrow$-saturated** if for every $n, m \in \mathbb{N}$, every pair of tuples $\langle \Sigma_1, \ldots, \Sigma_n \rangle$ and $\langle \Gamma_1, \ldots, \Gamma_m \rangle$ of sets of XPath$_=(\downarrow)$-node expressions, every $u \in T$, and $\star \in \{=, \neq\}$, the following is true:

> if $\langle \Sigma_1, \ldots, \Sigma_n \rangle$ and $\langle \Gamma_1, \ldots, \Gamma_m \rangle$ are $\star^{\downarrow}_{n,m}$-finitely satisfiable at $\mathcal{T}, u$ then $\langle \Sigma_1, \ldots, \Sigma_n \rangle$ and $\langle \Gamma_1, \ldots, \Gamma_m \rangle$ are $\star^{\downarrow}_{n,m}$-satisfiable at $\mathcal{T}, u$.

Proposition 12. *Any finitely branching data tree is $\downarrow$-saturated.*

Proof. Suppose by contradiction that there is $u \in T$ and tuples

$$\langle \Sigma_1, \ldots, \Sigma_n \rangle \quad \text{and} \quad \langle \Gamma_1, \ldots, \Gamma_m \rangle$$

of sets of XPath$_=(\downarrow)$-node expressions which are finitely $=^{\downarrow}_{n,m}$-satisfiable at $\mathcal{T}, u$ but not $=^{\downarrow}_{n,m}$-satisfiable at $\mathcal{T}, u$ (the case for $\mathcal{T}$ being $\neq^{\downarrow}_{n,m}$-satisfiable is analogous). Let

$$P = \{(v, w) \in T^2 \mid u \xrightarrow{n} v \wedge u \xrightarrow{m} w \wedge \mathit{data}(v) = \mathit{data}(w)\}.$$

Observe that P is finite because $\mathcal{T}$ is finitely branching. It is clear that if $(v, w) \in P$, so that $u = v_0 \to v_1 \to \cdots \to v_n = v \in T$, and $u = w_0 \to w_1 \to \cdots \to w_m = w \in T$ then either

$$\text{there is } i \in \{1, \ldots, n\} \text{ such that } \mathcal{T}, v_i \not\models \Sigma_i, \text{ or} \tag{14}$$

$$\text{there is } j \in \{1, \ldots, m\} \text{ such that } \mathcal{T}, w_j \not\models \Gamma_j. \tag{15}$$

We will define sets $(\Sigma_{i,v,w})_{1 \leq i \leq n}$ and $(\Gamma_{j,v,w})_{1 \leq j \leq m}$, each one of them with at most one element, as follows: If case (14) holds, assume i_0 is the least such number and define $\Sigma_{i_0,v,w}$ as $\{\rho\}$ for some node expression $\rho \in \Sigma_{i_0}$ such that $\mathcal{T}, v_{i_0} \not\models \rho$, define $\Sigma_{i,v,w} = \emptyset$ for any $i \in \{1, \ldots, n\} \setminus \{i_0\}$, and define $\Gamma_{j,v,w} = \emptyset$ for any $j \in \{1, \ldots, m\}$. If case (14) does not hold then case (15) holds, so assume j_0 is the least such number and define $\Gamma_{j_0,v,w}$ as $\{\rho\}$ for some node expression $\rho \in \Gamma_{j_0}$ such that $\mathcal{T}, w_{j_0} \not\models \rho$, define $\Gamma_{j,v,w} = \emptyset$ for any $j \in \{1, \ldots, m\} \setminus \{j_0\}$, and define $\Sigma_{i,v,w} = \emptyset$ for any $i \in \{1, \ldots, n\}$. Finally, define the finite sets $\Sigma'_i = \bigcup_{(v,w) \in P} \Sigma_{i,v,w}$ and $\Gamma'_j = \bigcup_{(v,w) \in P} \Gamma_{j,v,w}$. By construction, we have $\Sigma'_i \subseteq \Sigma_i$, $\Gamma'_j \subseteq \Gamma_j$ and $\langle \Sigma'_1, \ldots, \Sigma'_n \rangle$ and $\langle \Gamma'_1, \ldots, \Gamma'_m \rangle$ are not $=^{\downarrow}_{n,m}$-satisfiable at $\mathcal{T}, u$ which is a contradiction. □

Proposition 13. *Let $\mathcal{T}$ and $\mathcal{T}'$ be $\downarrow$-saturated data trees, and let $u \in T$ and $u' \in T'$. If $\mathcal{T}, u \equiv^{\downarrow} \mathcal{T}', u'$, then $\mathcal{T}, u \leftrightarrows^{\downarrow} \mathcal{T}', u'$.*

Proof. We show that Z, defined by xZx' iff $\mathcal{T}, x \equiv^{\downarrow} \mathcal{T}', x'$ is a XPath$_=(\downarrow)$-bisimulation between $\mathcal{T}, u$ and $\mathcal{T}', u'$. Clearly uZu', and **Harmony** holds. We only need to show that **Zig** and **Zag** are satisfied. We only check **Zig**, as **Zag** is analogous.

Suppose xZx', $x = v_0 \to v_1 \to \cdots \to v_n$ and $x = w_0 \to w_1 \to \cdots \to w_m$ are paths on $\mathcal{T}$, and $data(v_n) = data(w_m)$ (the case $data(v_n) \neq data(w_m)$ is shown analogously). For $i \in \{1, \ldots, n\}$, let $\Sigma_i = \mathrm{Th}_{\downarrow}(\mathcal{T}, v_i)$, and for $j \in \{1, \ldots, m\}$, let $\Gamma_j = \mathrm{Th}_{\downarrow}(\mathcal{T}, w_j)$. Furthermore, let Σ'_i be a finite subset of Σ_i, and let Γ'_j be a finite subset of Γ_j. Define

$$\varphi = \langle \downarrow[\wedge\Sigma'_1]\downarrow \ldots \downarrow [\wedge\Sigma'_n] = \downarrow[\wedge\Gamma'_1]\downarrow \ldots \downarrow [\wedge\Gamma'_m]\rangle.$$

It is clear that $\mathcal{T}, x \models \varphi$, and since by definition of Z we have $\mathcal{T}, x \equiv^{\downarrow} \mathcal{T}', x'$, we conclude that $\mathcal{T}', x' \models \varphi$. Hence $\langle \Sigma'_1, \ldots, \Sigma'_n\rangle$ and $\langle \Gamma'_1, \ldots, \Gamma'_m\rangle$ are $=^{\downarrow}_{n,m}$-satisfiable at x'. This holds for *any* finite sets $\Sigma'_i \subseteq \Sigma_i$ and $\Gamma'_j \subseteq \Gamma_j$, and so $\langle \Sigma_1, \ldots, \Sigma_n\rangle$ and $\langle \Gamma_1, \ldots, \Gamma_m\rangle$ are $=^{\downarrow}_{n,m}$-finitely satisfiable at x'. Since $\mathcal{T}'$ is $\downarrow$-saturated, $\langle \Sigma_1, \ldots, \Sigma_n\rangle$ and $\langle \Gamma_1, \ldots, \Gamma_m\rangle$ are $=^{\downarrow}_{n,m}$-satisfiable at $\mathcal{T}', x'$, so there are paths $x' = v'_0 \to v'_1 \to \cdots \to v'_n$ and $x' = w'_0 \to w'_1 \to \cdots \to w'_m$ on $\mathcal{T}'$ such that

i. $data(v'_n) = data(w'_m)$.

ii. For all $1 \leq i \leq n$, $\mathcal{T}', v'_i \models \mathrm{Th}_{\downarrow}(\mathcal{T}, v_i)$. This implies $\mathcal{T}, v_i \equiv^{\downarrow} \mathcal{T}', v'_i$: suppose by the way of contradiction that $\mathcal{T}', v'_i \models \varphi$ but $\mathcal{T}, v_i \not\models \varphi$. Then, $\mathcal{T}, v_i \models \neg\varphi$, and thus $\mathcal{T}', v'_i \models \neg\varphi$, a contradiction.

iii. For all $1 \leq j \leq m$, $\mathcal{T}', w'_j \models \mathrm{Th}_{\downarrow}(\mathcal{T}, w_j)$, i.e $\mathcal{T}, w_j \equiv^{\downarrow} \mathcal{T}', w'_j$.

By the definition of Z, conditions i, ii and iii above imply the conditions for the **Zig** clause of $\mathrm{XPath}_{=}(\downarrow)$-bisimulation. □

Saturation for the vertical fragment. Given a data tree $\mathcal{T}$ and $u \in T$, we say that the set of $\mathrm{XPath}_{=}(\uparrow\downarrow)$-node expressions Γ is $\boldsymbol{=^{\uparrow\downarrow}_{n,m}}$**-satisfiable** [resp. $\boldsymbol{\neq^{\uparrow\downarrow}_{n,m}}$**-satisfiable**] at $\mathcal{T}, u$ if there exist $v, w \in T$ such that $v \xrightarrow{n} u$, $v \xrightarrow{m} w$, $w \models \Gamma$ and $data(u) = data(w)$ [resp. $data(u) \neq data(w)$]. We say that Γ is $\boldsymbol{=^{\uparrow\downarrow}_{n,m}}$**-finitely satisfiable** [resp. $\boldsymbol{\neq^{\uparrow\downarrow}_{n,m}}$**-finitely satisfiable**] at $\mathcal{T}, u$ if for every finite $\Gamma' \subseteq \Gamma$, we have that Γ' is $=^{\uparrow\downarrow}_{n,m}$-satisfiable [resp. $=^{\uparrow\downarrow}_{n,m}$-satisfiable] at $\mathcal{T}, u$.

In Figure 16 we show an example of a set of node expressions that is $=^{\uparrow\downarrow}_{n,m}$-satisfiable at a pointed data tree $\mathcal{T}, x$.

Definition 14. We say that a data tree $\mathcal{T}$ is $\boldsymbol{\uparrow\downarrow}$**-saturated** if for every set of $\mathrm{XPath}_{=}(\uparrow\downarrow)$-node expressions Γ, every $u \in T$, every $n, m \in \mathbb{N}$, and $\star \in \{=, \neq\}$, the following is true:

if Γ is $\star^{\uparrow\downarrow}_{n,m}$-finitely satisfiable at $\mathcal{T}, u$ then Γ is $\star^{\uparrow\downarrow}_{n,m}$-satisfiable at $\mathcal{T}, u$.

Proposition 15. *Let $\mathcal{T}$ and $\mathcal{T}'$ be $\uparrow\downarrow$-saturated data trees, and let $u \in T$ and $u' \in T'$. If $\mathcal{T}, u \equiv^{\uparrow\downarrow} \mathcal{T}', u'$, then $\mathcal{T}, u \leftrightarrows^{\uparrow\downarrow} \mathcal{T}', u'$.*

Proof. We show that $Z \subseteq T \times T'$, defined by xZx' iff $\mathcal{T}, x \equiv^{\uparrow\downarrow} \mathcal{T}', x'$ is a $\mathrm{XPath}_{=}(\uparrow\downarrow)$-bisimulation between $\mathcal{T}, u$ and $\mathcal{T}', u'$. Clearly uZu', and **Harmony** also holds, so we only need to show that **Zig** and **Zag** are satisfied. We only check **Zig**, as **Zag** is analogous.

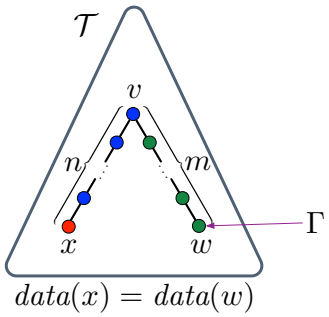


Figure 16: Γ is $=^{\uparrow\downarrow}_{n,m}$-satisfiable at $\mathcal{T}, x$.

Suppose xZx', $y \overset{n}{\to} x$ and $y \overset{m}{\to} z$ are in $\mathcal{T}$, and $data(x) = data(z)$ (the case $data(x) \neq data(z)$ can be shown analogously). Let $\Gamma = \mathrm{Th}_{\uparrow\downarrow}(\mathcal{T}, z)$, and let Γ' be a finite subset of Γ. Define

$$\varphi = \langle \varepsilon = \uparrow^n \downarrow^m [\wedge \Gamma'] \rangle.$$

It is clear that $\mathcal{T}, x \models \varphi$, and since by definition of Z we have $\mathcal{T}, x \equiv^{\uparrow\downarrow} \mathcal{T}', x'$, we conclude that $\mathcal{T}', x' \models \varphi$. Hence Γ' is $=^{\uparrow\downarrow}_{n,m}$-satisfiable at x'. This holds for *any* finite set $\Gamma' \subseteq \Gamma$, and so Γ is $=^{\uparrow\downarrow}_{n,m}$-finitely satisfiable at x'. Since $\mathcal{T}'$ is $\uparrow\downarrow$-saturated, Γ is $=^{\uparrow\downarrow}_{n,m}$-satisfiable at x', and thus there are $y' \overset{n}{\to} x'$ and $y' \overset{m}{\to} z'$ on $\mathcal{T}'$ such that $data(x') = data(z')$ and $\mathcal{T}', z' \models \mathrm{Th}_{\uparrow\downarrow}(\mathcal{T}, z)$, i.e $\mathcal{T}, z \equiv^{\uparrow\downarrow} \mathcal{T}', z'$. By the definition of Z, we have zZz' and hence the **Zig** clause for $\mathrm{XPath}_{=}(\uparrow\downarrow)$-bisimulation is verified. □

1.3.2 Weak data trees and quasi-ultraproducts

From now on we fix σ as the first-order signature from §I.1.3:

$$\sigma = \{\rightsquigarrow, \sim, (P_a)_{a \in \mathbb{A}}\},$$

where we recall that $\rightsquigarrow$ and $\sim$ are symbols for binary relations and we have an unary predicate symbol P_a for each $a \in \mathbb{A}$. As we have shown, every data tree $\mathcal{T}$ can be seen as a first-order σ-structure with universe T. But, for reasons that will become clearer later on, we will need to work with σ-structures which are slightly more general than data trees:

Definition 16. A σ-structure $\mathcal{T}$ is a **weak data tree** if the following conditions hold:

- $\sim$ is an equivalence relation;
- there is exactly one node r with no u such that $u \rightsquigarrow r$ (r is called *root* of $\mathcal{T}$);
- for all nodes $x \neq r$ there is exactly one y such that $y \rightsquigarrow x$; and
- for each $n \geq 0$ the relation $\rightsquigarrow$ has no cycles of length n.

Observe that a weak data tree need not be connected, and that the class of weak data trees is elementary, i.e. definable by a set of first-order σ-sentences (with equality). For a weak data tree $\mathcal{T}$ and $u \in T$, let $\mathcal{T}|_u$ be defined as:

$$\mathcal{T}|_u \stackrel{\text{def}}{=} \text{ the substructure of } \mathcal{T} \text{ induced by } \{v \in T \mid u \rightsquigarrow^* v\}.$$

Observe that in this case $\mathcal{T}|_u$ is a data tree with root u.

The following proposition shows the 'local' aspect of $\mathrm{XPath}_=(\downarrow)$ and $\mathrm{XPath}_=(\uparrow\downarrow)$. It is stated in terms of first-order because models are weak data trees. From now, unless specified otherwise, let Tr be the translation given in §I.1.3.

Proposition 17. *Let $\mathcal{T}$ be a weak data tree and let $r \rightsquigarrow^* u$ in $\mathcal{T}$.*

1. *If φ is a $\mathrm{XPath}_=(\downarrow)$-node expression then $\mathcal{T} \models \mathrm{Tr}_x(\varphi)[u]$ iff $\mathcal{T}|_r \models \mathrm{Tr}_x(\varphi)[u]$.*
2. *If r is the root of $\mathcal{T}$ and $\varphi \in \mathrm{XPath}_=(\uparrow\downarrow)$ then $\mathcal{T} \models \mathrm{Tr}_x(\varphi)[u]$ iff $\mathcal{T}|_r \models \mathrm{Tr}_x(\varphi)[u]$.*

Observe that the condition of r being the root in the second item is needed. Suppose for example we are on the data tree with only 2 nodes, the root r and its child u, with same data value. Consider now $\varphi = \langle \varepsilon = \uparrow \rangle$. Clearly $\mathcal{T} \models \mathrm{Tr}_x(\varphi)[u]$, but $\mathcal{T}|_u \not\models \mathrm{Tr}_x(\varphi)[u]$.

We now give some technical definitions and notation that are needed for the statement of Proposition 19. If $\mathcal{M}$ is a first-order σ-structure and $A \subseteq M$, we denote by σ_A the language obtained by adding to σ constant symbols for each $a \in A$. That is:

$$\sigma_A = \sigma \cup A.$$

$\mathcal{M}$ can be seen as a σ_A structure by interpreting the new symbols as expected: as the corresponding elements of M. Let $\mathrm{Th}_A(\mathcal{M})$ be the set of all σ_A-sentences true in $\mathcal{M}$:

$$\mathrm{Th}_A(\mathcal{M}) \stackrel{\text{def}}{=} \{\sigma_A\text{-sentences } \varphi \mid \mathcal{M} \models \varphi\}$$

Now, let κ be a cardinal. We recall the definition of κ-saturated first-order structures:

Definition 18. We say that the σ-structure $\mathcal{M}$ is **κ-saturated** if for all $A \subseteq M$ and all n, if $|A| < \kappa$ and $\Gamma(x_1, \ldots, x_n)$ is a set of σ_A-formulas with free variables among $x_1, \ldots, x_n$ such that $\Gamma(x_1, \ldots, x_n) \cup \mathrm{Th}_A(\mathcal{M})$ is satisfiable, then $\Gamma(x_1, \ldots, x_n)$ is realized in $\mathcal{M}$.

We now show that 2-saturated data trees are already both downward and vertical saturated, a result that is later used for the proofs of Lemma 23 and Lemma 28. For technical reasons we state this proposition in the more general setting of weak data trees.

Proposition 19. *Let $\mathcal{T}$ be a 2-saturated weak data tree and $r \in T$.*

1. *$\mathcal{T}|_r$ is a $\downarrow$-saturated data tree.*
2. *If r is the root of T then $\mathcal{T}|_r$ is a $\uparrow\downarrow$-saturated data tree.*

Proof. Let $\mathcal{T}' = \mathcal{T}|_r$ and let $u \in T'$. For item 1, let $\langle \Sigma_1, \ldots, \Sigma_n \rangle$ and $\langle \Gamma_1, \ldots, \Gamma_m \rangle$ be tuples of sets of XPath$_=(\downarrow)$-node expressions. Suppose $\langle \Sigma_1, \ldots, \Sigma_n \rangle$ and $\langle \Gamma_1, \ldots, \Gamma_m \rangle$ are $=^{\downarrow}_{n,m}$-finitely satisfiable at $\mathcal{T}', u$ (the case for $\neq^{\downarrow}_{n,m}$-finitely satisfiable is analogous). We show that $\langle \Sigma_1, \ldots, \Sigma_n \rangle$ and $\langle \Gamma_1, \ldots, \Gamma_m \rangle$ are $=^{\downarrow}_{n,m}$-satisfiable at $\mathcal{T}', u$.

Consider the following first-order $\sigma_{\{u\}}$-formula with free variables $\bar{x} = x_1, \ldots, x_n$ and $\bar{y} = y_1, \ldots, y_m$:

$$\varphi(\bar{x}, \bar{y}) = u \rightsquigarrow x_1 \wedge \bigwedge_{i=1}^{n-1} x_i \rightsquigarrow x_{i+1} \wedge u \rightsquigarrow y_1 \wedge \bigwedge_{j=1}^{m-1} y_j \rightsquigarrow y_{j+1} \wedge x_n \sim y_m.$$

Define the following set of first-order $\sigma_{\{u\}}$-formulas:

$$\Delta(\bar{x}, \bar{y}) = \{\varphi(\bar{x}, \bar{y})\} \cup \bigcup_{i=1}^{n} \mathrm{Tr}_{x_i}(\Sigma_i) \cup \bigcup_{j=1}^{m} \mathrm{Tr}_{y_j}(\Gamma_j).$$

Let $\Delta'(\bar{x}, \bar{y})$ be a finite subset of $\Delta(\bar{x}, \bar{y})$. Since $\langle \Sigma_1, \ldots, \Sigma_n \rangle$ and $\langle \Gamma_1, \ldots, \Gamma_m \rangle$ are $=^{\downarrow}_{n,m}$-finitely satisfiable at $\mathcal{T}', u$, then $\Delta'(\bar{x}, \bar{y})$ is satisfiable and, by item 1 of Proposition 17, consistent with $\mathrm{Th}_{\{u\}}(\mathcal{T})$. By compactness, $\Delta(\bar{x}, \bar{y})$ is satisfiable and consistent with $\mathrm{Th}_{\{u\}}(\mathcal{T})$. By 2-saturation, we conclude that $\Delta(\bar{x}, \bar{y})$ is realizable in $\mathcal{T}$, say at $\bar{v} = v_1, \ldots, v_n$ and $\bar{w} = w_1, \ldots, w_m$. Thus we have:

i. $u \rightsquigarrow v_1 \rightsquigarrow \cdots \rightsquigarrow v_n$ and $u \rightsquigarrow w_1 \rightsquigarrow \cdots \rightsquigarrow w_m$ in $\mathcal{T}$, and hence in $\mathcal{T}'$;

ii. for all $i \in \{1, \ldots, n\}$, $\mathcal{T} \models \mathrm{Tr}_{x_i}(\Sigma_i)[v_i]$, and for all $j \in \{1, \ldots, m\}$, $\mathcal{T} \models \mathrm{Tr}_{y_j}(\Gamma_j)[w_j]$; by item 1 of Proposition 17 this implies that $\mathcal{T}' \models \mathrm{Tr}_{x_i}(\Sigma_i)[v_i]$ and $\mathcal{T}' \models \mathrm{Tr}_{y_j}(\Gamma_j)[w_j]$;

iii. $v_n \sim w_m$ in $\mathcal{T}$, and hence in $\mathcal{T}'$.

Since Tr is truth-preserving, we have that for all $i \in \{1, \ldots, n\}$, $\mathcal{T}', v_i \models \Sigma_i$, and for all $j \in \{1, \ldots, m\}$, $\mathcal{T}', w_i \models \Gamma_i$. Together with i and iii we conclude that $\langle \Sigma_1, \ldots, \Sigma_n \rangle$ and $\langle \Gamma_1, \ldots, \Gamma_m \rangle$ are $=^{\downarrow}_{n,m}$-satisfiable at $\mathcal{T}', u$.

For item 2, let Γ be a set of XPath$_=(\uparrow\downarrow)$-node expressions. Suppose Γ is $=^{\uparrow\downarrow}_{n,m}$-finitely satisfiable at $\mathcal{T}', u$ (the case for $\neq^{\uparrow\downarrow}_{n,m}$-finitely satisfiable is analogous). We show that Γ are $=^{\uparrow\downarrow}_{n,m}$-satisfiable at $\mathcal{T}', u$.

Consider the following first-order $\sigma_{\{u\}}$-formula with free variable y:

$$\begin{aligned} \varphi(y) \quad = \quad & (\exists x_0 \ldots \exists x_n)(\exists y_0 \ldots \exists y_m)[x_n = u \wedge y = y_m \wedge x_0 = y_0 \wedge \\ & \bigwedge_{i=0}^{n-1} x_i \rightsquigarrow x_{i+1} \wedge \bigwedge_{j=0}^{m-1} y_j \rightsquigarrow y_{j+1} \wedge x_n \sim y_m]. \end{aligned}$$

Define the following set of first-order $\sigma_{\{u\}}$-formulas: $\Delta(y) = \{\varphi(y)\} \cup \mathrm{Tr}_y(\Gamma)$. Let $\Delta'(y)$ be a finite subset of $\Delta(y)$. Since Γ is $=^{\uparrow\downarrow}_{n,m}$-finitely satisfiable at $\mathcal{T}', u$, then $\Delta'(y)$ is satisfiable and, by item 2 of Proposition 17, consistent with $\mathrm{Th}_{\{u\}}(\mathcal{T})$. By compactness, $\Delta(y)$ is satisfiable and consistent with $\mathrm{Th}_{\{u\}}(\mathcal{T})$. By 2-saturation, we conclude that $\Delta(y)$ is realizable in $\mathcal{T}$, say at w. Thus we have:

iv. There is $v \in T$ such that $v \xrightarrow{n} u$ and $v \xrightarrow{m} w$ in $\mathcal{T}$ and hence in $\mathcal{T}'$.

v. $\mathcal{T} \models \mathrm{Tr}_y(\Gamma)[w]$; by item 2 of Proposition 17 this implies that $\mathcal{T}' \models \mathrm{Tr}_y(\Gamma)[w]$;

vi. $u \sim w$ in $\mathcal{T}$, and hence in $\mathcal{T}'$.

Since Tr is truth-preserving, we have that $\mathcal{T}', w \models \Gamma$. Together with iv and vi we conclude that Γ is $=^{\uparrow\downarrow}_{n,m}$-satisfiable at $\mathcal{T}', u$. □

In what follows, we introduce the notion of *quasi-ultraproduct*, a variant of the usual notion of first-order model theory, which will be needed for the definability theorems.

Let $I \neq \emptyset$, let U be an ultrafilter over I and let $(\mathcal{T}_i)_{i\in I}$ be a family of data trees. As usual, we denote with $\prod_U \mathcal{T}_i$ the ultraproduct of $(\mathcal{T}_i)_{i\in I}$ modulo U. Observe that by the fundamental theorem of ultraproducts (Thm. 10), $\prod_U \mathcal{T}_i$ is a weak data tree σ-structure —though it may not be a data tree because it may be disconnected, as it is shown next:

Example 20. For $i \in \mathbb{N}$, let $\mathcal{T}_i$ as any data tree of height at least i, and let u_i as any node of T_i at distance i from the root of $\mathcal{T}_i$. Let $\varphi_n(x)$ be the first-order property "x is at distance at least n from the root". It is clear that $\mathcal{T}_m \models \varphi_n[u_m]$ for every $m \geq n$. Let u^* be the ultralimit of $(u_i)_{i\in I}$ modulo U. Since $\{m \mid m \geq n\} \in U$ for any non-principal U, we conclude that $\prod_U \mathcal{T}_i \models \varphi_n[u^*]$ for every n, and so u^* is disconnected from the root of $\prod_U \mathcal{T}_i$.

Let $(\mathcal{T}_i, u_i)_{i\in I}$ be a family of pointed data trees. The ultraproduct of such *pointed* data trees is defined, as usual, by $(\prod_U \mathcal{T}_i, u^*)$, where u^* is the ultralimit of $(u_i)_{i\in I}$ modulo U.

Definition 21. Suppose $(\mathcal{T}_i, u_i)_{i\in I}$ is a family of pointed data trees, r_i is the root of $\mathcal{T}_i$, U is an ultrafilter over I, $\mathcal{T}^* = \prod_U \mathcal{T}_i$, and u^* and r^* are the ultralimits of $(u_i)_{i\in I}$ and $(r_i)_{i\in I}$ modulo U respectively.

1. The **↓-quasi ultraproduct** of $(\mathcal{T}_i, u_i)_{i\in I}$ modulo U is the pointed data tree $(\mathcal{T}^*|_{u^*}, u^*)$.

2. The **↑↓-quasi ultraproduct** of $(\mathcal{T}_i, u_i)_{i\in I}$ modulo U is the pair $(\mathcal{T}^*|_{r^*}, u^*)$.

Observe that both $\mathcal{T}^*|_{u^*}$ and $\mathcal{T}^*|_{r^*}$ are data trees. However, while u^* is in the domain of the former, it may not be in the domain of the latter (cf. Example 20). See Figure 17 for a graphical comparison between ultraproduct and ↓-quasi ultraproduct.

Hence, in general, pointed data trees are not closed under ↑↓-quasi ultraproduct. Let $k \geq 0$, let $\mathcal{T}$ be a data tree and let $u \in \mathcal{T}$. We say that $\mathcal{T}, u$ is a ***k*-bounded pointed data tree** if u is at distance at most k from the root of $\mathcal{T}$. In particular, if r is the root of $\mathcal{T}$ then $\mathcal{T}, r$ is a 0-bounded pointed data tree. The following proposition states that k-bounded data trees are closed under ↑↓-quasi ultraproducts.

Proposition 22. *Let $(\mathcal{T}_i, u_i)_{i\in I}$ be a family of k-bounded pointed data trees. Then the ↑↓-quasi ultraproduct of $(\mathcal{T}_i, u_i)_{i\in I}$ is a k-bounded pointed data tree.*

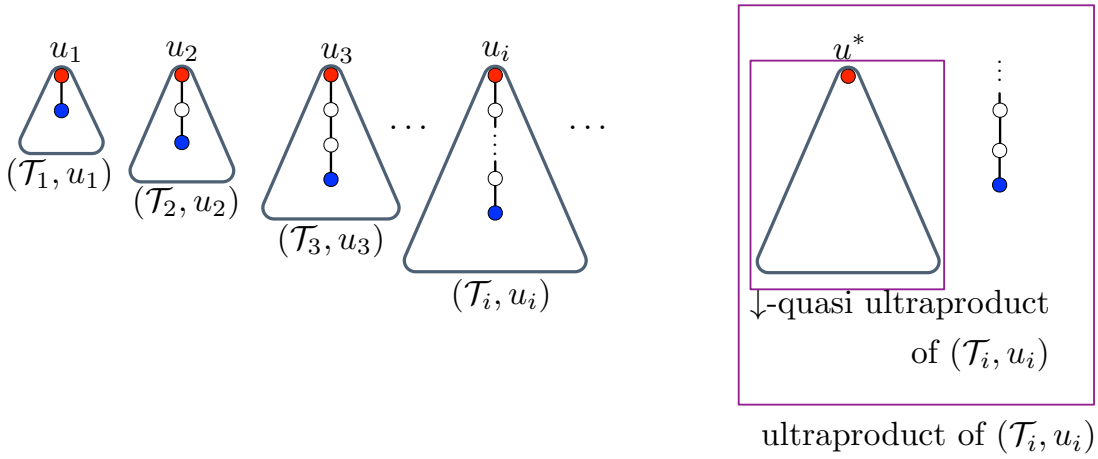


Figure 17: As shown in Example 20, the ultraproduct of the lower marked nodes of each $\mathcal{T}_j$ is disconnected from the ultraproduct of the u_i, so the ultraproduct of $(\mathcal{T}_i, u_i)$ is not a tree.

Proof. Let $(\mathcal{T}^{\uparrow\downarrow}, u^*)$ be the $\uparrow\downarrow$-quasi ultraproduct of $(\mathcal{T}_i, u_i)_{i\in I}$ modulo U. By definition it is clear that $\mathcal{T}^{\uparrow\downarrow}$ is a data tree. To see that $u^* \in T^{\uparrow\downarrow}$, let

$$\varphi(x) \;=\; (\exists r)\,[\neg(\exists y) y \rightsquigarrow r \wedge [r = x \vee r \rightsquigarrow x \vee \bigvee_{1\le i<k} (\exists z_1 \ldots \exists z_i)[r \rightsquigarrow z_1 \wedge z_{i-1} \rightsquigarrow x \wedge \bigwedge_{1\le j<i-1} z_j \rightsquigarrow z_{j+1}]]],$$

which is a first-order formula for "r is the root and x is at distance at most k from r". Since for every $i \in I$ we have $\mathcal{T}_i \models \varphi[u_i]$, we conclude that $\mathcal{T}^{\uparrow\downarrow} \models \varphi[u^*]$ and hence u^* is at distance at most k from the root of $\mathcal{T}^{\uparrow\downarrow}$. □

As a particular case one has the notion of **↓-quasi ultrapower** and **↑↓-quasi ultrapower** of a family of pointed data trees. Observe that if $(\mathcal{T}^{\uparrow\downarrow}, u^*)$ is the $\uparrow\downarrow$-quasi ultrapower of $(\mathcal{T}, u)_{i\in I}$ then u^* belongs to the domain of $\mathcal{T}^{\uparrow\downarrow}$ and so $(\mathcal{T}^{\uparrow\downarrow}, u^*)$ is a pointed data tree.

1.3.3 Definability

In this subsection we state the main results of definability of classes of pointed data trees via node expressions of XPath$_=(\downarrow)$ and XPath$_=(\uparrow\downarrow)$.

We begin with the downward fragment:

Definability via node expressions of XPath$_=(\downarrow)$

Lemma 23. *Let $(\mathcal{T}, u)$ and $(\mathcal{T}', u')$ be two pointed data trees such that $\mathcal{T}, u \equiv^{\downarrow} \mathcal{T}', u'$. Then there exist ↓-quasi ultrapowers $(\mathcal{T}^{\downarrow}, u^*)$ and $(\mathcal{T}'^{\downarrow}, u'^*)$ of $(\mathcal{T}, u)$ and $(\mathcal{T}', u')$ respectively such that $(\mathcal{T}^{\downarrow}, u^*) \leftrightarrows^{\downarrow} (\mathcal{T}'^{\downarrow}, u'^*)$*

Proof. It is known that there is a suitable ultrafilter U such that $\prod_U \mathcal{T}$ and $\prod_U \mathcal{T}'$ are ω-saturated (see e.g. [15, Lemma 2.7.3]). By item 1 Proposition 19, $\mathcal{T}^{\downarrow} = (\prod_U \mathcal{T})|_{u^*}$ and $\mathcal{T}^{\downarrow} = (\prod_U \mathcal{T}')|_{u'^*}$ are $\downarrow$-saturated data trees. By hypothesis $\mathcal{T}, u \equiv^{\downarrow} \mathcal{T}', u'$, and hence $\mathcal{T}^{\downarrow}, u^* \equiv^{\downarrow} \mathcal{T}'^{\downarrow}, u'^*$. Finally, by Proposition 13, $\mathcal{T}^{\downarrow}, u^* \leftrightarrows^{\downarrow} \mathcal{T}'^{\downarrow}, u'^*$. □

Lemma 24. *Let K be a class of pointed data trees and let Σ be a set of XPath$_=$($\downarrow$)-node expressions finitely satisfiable in K. Then Σ is satisfiable in some $\downarrow$-quasi ultraproduct of pointed data trees in K.*

Proof. Let $I = \{\Sigma_0 \subseteq \Sigma \mid \Sigma_0 \text{ is finite}\}$ and for each $\varphi \in \Sigma$, let $\hat{\varphi} = \{i \in I \mid \varphi \in i\}$. Then the set $E = \{\hat{\varphi} \mid \varphi \in \Sigma\}$ has the finite intersection property: $\{\varphi_1, \ldots, \varphi_n\} \in \hat{\varphi}_1 \cap \cdots \cap \hat{\varphi}_n$. By the Ultrafilter Theorem (see [23, Proposition 4.1.3]) E can be extended to an ultrafilter U over I.

Since Σ is finitely satisfiable in K, for each $i \in I$ there is $(\mathcal{T}_i, u_i) \in K$ such that $\mathcal{T}_i, u_i \models i$. Let $(\mathcal{T}^{\downarrow}, u^*)$ be the $\downarrow$-quasi ultraproduct of $(\mathcal{T}_i, u_i)_{i \in I}$ modulo U. We show that $\mathcal{T}^{\downarrow}, u^* \models \Sigma$: let $\varphi \in \Sigma$. Then $\hat{\varphi} \in E \subseteq U$ and $\hat{\varphi} \subset \{i \in I \mid \mathcal{T}_i, u_i \models \varphi\}$. Hence $\{i \in I \mid \mathcal{T}_i, u_i \models \varphi\} \in U$, which implies that $\prod_U \mathcal{T}_i \models \mathrm{Tr}_x(\varphi)[u^*]$, where u^* is the ultralimit of $(u_i)_{i \in I}$ and Tr is the translation into first-order logic given in §I.1.3. Since $\mathcal{T}^{\downarrow} = (\prod_U \mathcal{T}_i)|_{u^*}$, by item 1 of Proposition 17 we conclude that $\mathcal{T}^{\downarrow}, u^* \models \varphi$. □

If K is a class of pointed data trees, we denote its **complement** over the universe of data trees by $\overline{K}$.

Theorem 25. *Let K be a class of pointed data trees. Then K is definable by a set of XPath$_=$($\downarrow$)-node expressions iff K is closed under XPath$_=$($\downarrow$)-bisimulations and $\downarrow$-quasi ultraproducts, and $\overline{K}$ is closed under $\downarrow$-quasi ultrapowers.*

Proof. For ($\Rightarrow$), suppose that K is definable by a set of XPath$_=$($\downarrow$)-node expressions. By Theorem 7 it is clear that K is closed under XPath$_=$($\downarrow$)-bisimulations. By the fundamental theorem of ultraproducts (Thm. 10) together with item 1 of Proposition 17 it is clear that K is closed under $\downarrow$-quasi ultraproducts. It is also clear that the fundamental theorem of ultraproducts and the fact that any XPath$_=$($\downarrow$)-node expression is expressible in first-order imply that $\mathcal{T}, u \equiv^{\downarrow} \mathcal{T}^{\downarrow}, u^*$ for any $(\mathcal{T}^{\downarrow}, u^*)$ $\downarrow$-quasi ultrapower modulo U, and therefore that $\overline{K}$ is closed under $\downarrow$-quasi ultrapowers.

For ($\Leftarrow$), suppose K is closed under bisimulations and $\downarrow$-quasi ultraproducts, and $\overline{K}$ is closed under $\downarrow$-quasi ultrapowers. We show that $\Gamma = \bigcap_{(\mathcal{T},u) \in K} \mathrm{Th}_{\downarrow}(\mathcal{T}, u)$ defines K. It is clear that if $(\mathcal{T}, u) \in K$ then $\mathcal{T}, u \models \Gamma$.

Now suppose that $\mathcal{T}, u \models \Gamma$ and consider $\Sigma = \mathrm{Th}_{\downarrow}(\mathcal{T}, u)$. Let Δ be a finite subset of Σ, and assume that Δ is not satisfiable in K. Then, $\neg \bigwedge \Delta$ is true in every pointed data tree of K, so $\neg \bigwedge \Delta \in \Gamma$. Therefore $\mathcal{T}, u \models \neg \bigwedge \Delta$ which is a contradiction because $\Delta \subseteq \Sigma$. This shows that Σ is finitely satisfiable in K.

By Lemma 24, Σ is satisfiable in some $\downarrow$-quasi ultraproduct of pointed data trees in K, and since K is closed under $\downarrow$-quasi ultraproducts, Σ is satisfiable in K. Then there exists $(\mathcal{T}', u') \in K$ such that $\mathcal{T}', u' \models \Sigma$ and therefore $\mathcal{T}, u \equiv^{\downarrow} \mathcal{T}', u'$. By Lemma 23, there exist

$\downarrow$-quasi ultrapowers $(\mathcal{T}^{\downarrow}, u^*)$ and $(\mathcal{T}'^{\downarrow}, u'^*)$ of $(\mathcal{T}, u)$ and $(\mathcal{T}', u')$ respectively such that $(\mathcal{T}^{\downarrow}, u^*) \leftrightarrows^{\downarrow} (\mathcal{T}'^{\downarrow}, u'^*)$. Since K is closed under XPath$_=(\downarrow)$-bisimulations, $(\mathcal{T}^{\downarrow}, u^*) \in K$. Suppose $(\mathcal{T}, u) \in \overline{K}$. Since $\overline{K}$ is closed under $\downarrow$-quasi ultrapowers, $(\mathcal{T}^{\downarrow}, u^*) \in \overline{K}$, and this is a contradiction. Hence we conclude $(\mathcal{T}, u) \in K$. □

Theorem 26. *Let K be a class of pointed data trees. Then K is definable by an XPath$_=(\downarrow)$-node expression iff both K and $\overline{K}$ are closed under XPath$_=(\downarrow)$-bisimulations and $\downarrow$-quasi ultraproducts.*

Proof. For ($\Rightarrow$) suppose that K is definable by an XPath$_=(\downarrow)$-node expression. By Theorem 7 it is clear that K and $\overline{K}$ are closed under bisimulations. By the fundamental theorem of ultraproducts together with item 1 of Proposition 17 it is clear that K and $\overline{K}$ are closed under $\downarrow$-quasi ultraproducts.

For ($\Leftarrow$) suppose K and $\overline{K}$ are closed under bisimulations and $\downarrow$-quasi ultraproducts. Then, by Theorem 25, there exist sets Γ_1 and Γ_2 of XPath$_=(\downarrow)$-node expression defining K and $\overline{K}$ respectively. Consider the set of XPath$_=(\downarrow)$-node expressions $\Gamma_1 \cup \Gamma_2$. This set is clearly inconsistent and so, by compactness, there are finite sets Δ_1 and Δ_2 such that $\Delta_i \subseteq \Gamma_i$ $(i = 1, 2)$ and

$$\mathcal{T}, u \models \bigwedge \Delta_1 \rightarrow \neg \bigwedge \Delta_2 \tag{16}$$

for any pointed data tree $(\mathcal{T}, u)$. We show that $\varphi = \bigwedge \Delta_1$ defines K. On the one hand, it is clear that if $(\mathcal{T}, u) \in K$ then $\mathcal{T}, u \models \varphi$. On the other hand, suppose that $\mathcal{T}, u \models \varphi$. From (16) we conclude $\mathcal{T}, u \models \neg \bigwedge \Delta_2$ and so $\mathcal{T}, u \not\models \Gamma_2$. Then $(\mathcal{T}, u) \notin \overline{K}$ as we wanted to prove.

□

Like Theorem 26, the following result characterizes when a class of pointed data trees is definable by a single XPath$_=(\downarrow)$-node expression. However, instead of using the rather abstract notion of $\downarrow$-quasi ultraproducts, it uses the perhaps more natural notion of ℓ-bisimulation.

Theorem 27. *Let K be a class of pointed data trees. Then K is definable by a node expression of XPath$_=(\downarrow)$ iff K is closed by ℓ-bisimulations for some ℓ.*

Proof. ($\Rightarrow$) is a direct consequence of Theorem 7. Let us see ($\Leftarrow$). Given $\mathcal{T}, u$ a pointed data tree in K, we know [44, Corollary 3.2] that $\{\mathcal{T}', u' \mid \mathcal{T}, u \equiv_{\ell}^{\downarrow} \mathcal{T}', u'\}$ is definable by an XPath$_=(\downarrow)$-node expression $\chi_{\ell,\mathcal{T},u}$ of downward depth $\leq \ell$. We show that

$$\varphi = \bigvee_{(\mathcal{T},u)\in K} \chi_{\ell,\mathcal{T},u}$$

defines K. In [44, Proposition 3.1] it is shown that $\equiv_{\ell}^{\downarrow}$ has finite index, and therefore the above disjunction is equivalent to a finite one. On the one hand, if $(\mathcal{T}', u') \in K$ then it is clear that $\mathcal{T}', u' \models \chi_{\ell,\mathcal{T}',u'}$ and so $\mathcal{T}', u' \models \varphi$. On the other hand, we have $\mathcal{T}', u' \models \varphi$ iff there is $(\mathcal{T}, u) \in K$ such that $\mathcal{T}', u' \models \chi_{\ell,\mathcal{T},u}$ iff there is $(\mathcal{T}, u) \in K$ such that $\mathcal{T}, u \leftrightarrows_{\ell}^{\downarrow} \mathcal{T}', u'$. Hence since K is closed under $\leftrightarrows_{\ell}^{\downarrow}$, if $\mathcal{T}', u' \models \varphi$ we have $(\mathcal{T}', u') \in K$. □

We now turn to the vertical fragment:

Definability via node expressions of XPath$_=(\uparrow\downarrow)$

Lemma 28. *Let $(\mathcal{T}, u)$ and $(\mathcal{T}', u')$ be two pointed data trees such that $\mathcal{T}, u \equiv^{\uparrow\downarrow} \mathcal{T}', u'$. Then there exist $\uparrow\downarrow$-quasi ultrapowers $(\mathcal{T}^{\uparrow\downarrow}, u^*)$ and $(\mathcal{T}'^{\uparrow\downarrow}, u'^*)$ of $(\mathcal{T}, u)$ and $(\mathcal{T}', u')$ respectively such that $(\mathcal{T}^{\uparrow\downarrow}, u^*) \leftrightarrows^{\uparrow\downarrow} (\mathcal{T}'^{\uparrow\downarrow}, u'^*)$*

Proof. The proof is analogous to the proof of Lemma 23 but using item 2 instead of item 1 of Proposition 19 and Proposition 15 instead of Proposition 13. □

Lemma 29. *Let K be a class of k-bounded pointed data trees and let Σ be a set of XPath$_=(\uparrow\downarrow)$-node expressions finitely satisfiable in K. Then Σ is satisfiable in some $\uparrow\downarrow$-quasi ultraproduct of pointed data trees in K.*

Proof. The proof is analogous to the proof of Lemma 24 but taking $\uparrow\downarrow$-quasi ultraproducts instead of $\downarrow$-quasi ultraproducts and using item 2 instead of item 1 of Proposition 17. To apply this Proposition, one has to note that $u^* \in \mathcal{T}^{\uparrow\downarrow}$ since the $\mathcal{T}_i, u_i$ are k-bounded pointed data trees. □

In the next two theorems, the universe of pointed data trees is restricted to those which are k-bounded (for any fixed k). Therefore, the operations of closure and complement must be taken with respect to this universe.

Theorem 30. *Over k-bounded pointed data trees: K is definable by a set of XPath$_=(\uparrow\downarrow)$-node expressions iff K is closed under XPath$_=(\uparrow\downarrow)$-bisimulations and $\uparrow\downarrow$-quasi ultraproducts, and $\overline{K}$ is closed under $\uparrow\downarrow$-quasi ultrapowers.*

Proof. The proof is analogous to the proof of Theorem 25 but replacing pointed data trees for k-bounded pointed data trees and every occurrence of $\downarrow$ for $\uparrow\downarrow$. Also, for $(\Rightarrow)$, one has to use item 2 instead of item 1 of Proposition 17 and for $(\Leftarrow)$, Lemmas 29 and 28 instead of Lemmas 24 and 23. □

Theorem 31. *Over k-bounded pointed data trees: K is definable by an XPath$_=(\uparrow\downarrow)$-node expression iff both K and $\overline{K}$ are closed under XPath$_=(\uparrow\downarrow)$-bisimulations and $\uparrow\downarrow$-quasi ultraproducts.*

As in Theorem 27, one can also restate Theorem 31 in terms of (r, s, k)-bisimulations for XPath$_=(\uparrow\downarrow)$.

Theorem 32. *Let K be a class of pointed data trees. Then K is definable by a node expression of XPath$_=(\uparrow\downarrow)$ iff K is closed by (r, s, k)-bisimulations for some r, s, k.*

Applications

We now present some examples of application of these definability theorems for pointed data trees:

Example 33. **A class of pointed data trees definable by a single XPath$_=$(↑↓)-node expression but not definable by a set of XPath$_=$(↓)-node expressions.** Let $dist_3(x)$ be the property stating that there are nodes y, z so that $x{\rightarrow}y{\rightarrow}z$ and x, y, z have pairwise distinct data values. It can be checked that the XPath$_=$(↑↓)-node expression $\varphi = \langle \varepsilon \neq \downarrow\downarrow[\langle \varepsilon \neq \uparrow[\langle \varepsilon \neq \uparrow\rangle]\rangle]\rangle$ expresses $dist_3(x)$. We have already seen in §I.1.3 that this property is not expressible with a single XPath$_=$(↓)-node expression. Now, let K be the class of pointed data trees $(\mathcal{T}, u)$, where $dist_3(u)$ holds. Figure 12 shows that K is not closed under XPath$_=$(↓)-bisimulations and hence, by Theorem 25, K is not definable even by a set of XPath$_=$(↓)-node expressions.

Example 34. **A class of pointed data trees definable in first-order (over data trees) but not definable by a set of XPath$_=$(↑↓)-node expressions.** Let K be the class of pointed data trees $(\mathcal{T}, u)$ where u is the root of $\mathcal{T}$ and $\mathcal{T}$ has some node labeled a. On the one hand, K is definable by a first-order σ-formula over the class of data trees. On the other, K is closed under XPath$_=$(↑↓)-bisimulations but not closed under ↑↓-quasi ultraproducts: for $i \in \mathbb{N}$ define $\mathcal{T}_i$ as any tree of height i whose only node labeled a is at distance i from the root, and define u_i as the root of $\mathcal{T}_i$. By an argument similar to the one used in Example 20 one can show that if $(\mathcal{T}^{\uparrow\downarrow}, u^*)$ is any ↑↓-quasi ultraproduct of $(\mathcal{T}_i, u_i)_{i\in\mathbb{N}}$ then no node of $\mathcal{T}^{\uparrow\downarrow}$ has label a. By Theorem 30, K is not definable by a set of XPath$_=$(↑↓)-node expression.

Example 35. **A class of pointed data trees definable by set of XPath$_=$(↓)-node expressions but not by single XPath$_=$(↑↓)-node expression** Let K be the class of pointed data trees $(\mathcal{T}, u)$, where u is the root of $\mathcal{T}$, and $\mathcal{T}$ has infinite height. It is clear that K is definable by the set of XPath$_=$(↓)-node expressions $\{\langle\downarrow^n\rangle \mid n \geq 0\}$. However, by Theorem 31, it is not definable by an XPath$_=$(↑↓)-node expressions, as $\overline{K}$ is not closed under ↑↓-quasi ultraproducts (as any ↑↓-quasi ultraproduct of finite pointed data trees with increasing height has infinite height).

Example 36. **A class of pointed data trees definable by set of XPath$_=$(↑↓)-node expressions but not definable by single XPath$_=$(↑↓)-node expression.** Let K be the class of pointed data trees $(\mathcal{T}, u)$, where u is the root of $\mathcal{T}$, and for all $v \in T$ we have $dist_3(v)$. On the one hand, K is definable by the set of XPath$_=$(↑↓)-node expressions $\{\neg\langle\downarrow^n [\neg\varphi_4]\rangle \mid n \geq 0\}$. On the other, for $i \in \mathbb{N}$, let $(\mathcal{T}_i, u_i)$ be any pointed data tree not in K, of height at least $i+1$, where u_i is the root of $\mathcal{T}_i$, and such that for all $v \in T_i$ at distance at most i from u_i we have $dist_3(v)$. Let $(\mathcal{T}^{\uparrow\downarrow}, u^*)$ be any ↑↓-quasi ultraproduct of $(\mathcal{T}_i, u_i)_{i\in\mathbb{N}}$. One can see that all nodes of $v \in T^{\uparrow\downarrow}$ satisfy $dist_3(v)$, and so $(\mathcal{T}^{\uparrow\downarrow}, u^*) \in K$. Therefore $\overline{K}$ is not closed under ↑↓-quasi ultraproducts and by Theorem 31, K is not definable by an XPath$_=$(↑↓)-node expression.

1.3.4 Separation

The theorems of separation are closely related to definability: they provide conditions to separate two disjoint classes of models K_1 and K_2 by means of node expressions, i.e. to

find a class K, definable by a set of node expressions or a single node expression, such that $K_1 \subseteq K$ and $K \cap K_2 = \emptyset$.

Separation via node expressions of XPath$_=(\downarrow)$

Theorem 37. *Let K_1 and K_2 be two disjoint classes of pointed data trees such that K_1 is closed under XPath$_=(\downarrow)$-bisimulations and $\downarrow$-quasi ultraproducts and K_2 is closed under XPath$_=(\downarrow)$-bisimulations and $\downarrow$-quasi ultrapowers. Then there exists a third class K which is definable by a set of XPath$_=(\downarrow)$-node expressions, contains K_1 and is disjoint from K_2.*

Proof. Let $K = \{(\mathcal{T}', u') \mid$ there is $(\mathcal{T}, u) \in K_1$ such that $\mathcal{T}, u \equiv^{\downarrow} \mathcal{T}', u'\}$. Clearly, $K_1 \subseteq K$. We first show that $K \cap K_2 = \emptyset$. Suppose that there is a pointed model $(\mathcal{T}', u') \in K \cap K_2$. Then, there exists $(\mathcal{T}, u) \in K_1$ such that $\mathcal{T}, u \equiv^{\downarrow} \mathcal{T}', u'$ and, by Lemma 23, there exist $\downarrow$-quasi ultrapowers $(\mathcal{T}^{\downarrow}, u^*)$ and $(\mathcal{T}'^{\downarrow}, u'^*)$ of $(\mathcal{T}, u)$ and $(\mathcal{T}', u')$ respectively such that $\mathcal{T}^{\downarrow}, u^* \leftrightarrows^{\downarrow} \mathcal{T}'^{\downarrow}, u'^*$. Since K_1 is closed under $\downarrow$-quasi ultraproducts and XPath$_=(\downarrow)$-bisimulations and K_2 is closed under $\downarrow$-quasi ultrapowers, $(\mathcal{T}'^{\downarrow}, u'^*) \in K_1 \cap K_2$ which is a contradiction.

To conclude the proof, we show that K is definable by a set of XPath$_=(\downarrow)$-node expressions. By Theorem 25, it is enough to check that K is closed under XPath$_=(\downarrow)$-bisimulations and $\downarrow$-quasi ultraproducts and $\overline{K}$ is closed under $\downarrow$-quasi ultrapowers. Clearly, K is closed under XPath$_=(\downarrow)$-bisimulations, as $\leftrightarrows^{\downarrow}$ implies $\equiv^{\downarrow}$. Now, let $(\mathcal{T}'_i, u'_i)_{i \in I}$ be a family of pointed data trees contained in K. Then, for all $i \in I$, there is $(\mathcal{T}_i, u_i) \in K_1$ such that $\mathcal{T}_i, u_i \equiv^{\downarrow} \mathcal{T}'_i, u'_i$. By the fundamental theorem of ultraproducts, if U is an ultrafilter over I and $\mathcal{T}^*, u^*$, $\mathcal{T}'^*, u'^*$ are the ultraproducts of the families $(\mathcal{T}_i, u_i)_{i \in I}$ and $(\mathcal{T}'_i, u'_i)_{i \in I}$ respectively, then $(\mathcal{T}^*, u^*) \equiv^{\downarrow} (\mathcal{T}'^*, u'^*)$, and by Proposition 17 $(\mathcal{T}^{\downarrow}, u^*) \equiv^{\downarrow} (\mathcal{T}'^{\downarrow}, u'^*)$. Now, since K_1 is closed under $\downarrow$-quasi ultraproducts, $(\mathcal{T}'^{\downarrow}, u'^*) \in K$ which proves that K is closed under $\downarrow$-quasi ultraproducts. Finally, let $(\mathcal{T}', u') \in \overline{K}$. Suppose that $(\mathcal{T}'^{\downarrow}, u'^*)$, some $\downarrow$-quasi ultrapower of $(\mathcal{T}', u')$, belongs to K. By the fundamental theorem of ultraproducts, $(\mathcal{T}'^{\downarrow}, u'^*) \equiv^{\downarrow} (\mathcal{T}', u')$. So, since K is closed under $\equiv^{\downarrow}$, $(\mathcal{T}', u') \in K$, which is a contradiction. □

Theorem 38. *Let K_1 and K_2 be two disjoint classes of pointed data trees closed under XPath$_=(\downarrow)$-bisimulations and $\downarrow$-quasi ultraproducts. Then there exists a third class K which is definable by an XPath$_=(\downarrow)$-node expression, contains K_1 and is disjoint from K_2.*

Proof. By Theorem 37, there exists a class K' definable by a set of XPath$_=(\downarrow)$-node expressions Γ_1, containing K_1 and disjoint from K_2. Observe that as a consequence of Theorem 7, such K' is closed under XPath$_=(\downarrow)$-bisimulations and $\downarrow$-quasi ultraproducts. Using Theorem 37 again for K_2 and K', we have another class K'' also definable by a set of XPath$_=(\downarrow)$-node expressions Γ_2, containing K_2 and disjoint from K'.

Now consider the set of XPath$_=(\downarrow)$-node expressions $\Gamma_1 \cup \Gamma_2$. This set is clearly inconsistent and so, by compactness, there are finite sets Δ_1 and Δ_2 such that $\Delta_i \subseteq \Gamma_i$ $(i = 1, 2)$ and $\bigwedge \Delta_1 \wedge \bigwedge \Delta_2$ is unsatisfiable. Now let $K = \{\mathcal{T}, u \mid \mathcal{T}, u \models \bigwedge \Delta_1\}$. This K satisfies the desired properties, as $K_1 \subset K' \subset K$ and $K_2 \cap K \subset K'' \cap K = \emptyset$. □

Separation via node expressions of XPath$_=(\uparrow\downarrow)$

The same proofs apply for the XPath$_=(\uparrow\downarrow)$ version of these theorems, using the corresponding notions of bisimulations and quasi ultraproducts and Lemmas 28 and 30 instead of Lemmas 23 and 25, with the proviso that the universe of pointed data trees is restricted to those which are k-bounded (and so operations of closure and complement must be taken with respect to this universe):

Theorem 39. *Let K_1 and K_2 be two disjoint classes of k-bounded pointed data trees such that K_1 is closed under XPath$_=(\uparrow\downarrow)$-bisimulations and $\uparrow\downarrow$-quasi ultraproducts and K_2 is closed under XPath$_=(\uparrow\downarrow)$-bisimulations and $\uparrow\downarrow$-quasi ultrapowers. Then there exists a third class K which is definable by a set of XPath$_=(\uparrow\downarrow)$-node expressions, contains K_1 and is disjoint from K_2.*

Theorem 40. *Let K_1 and K_2 be two disjoint classes of k-bounded pointed data trees closed under XPath$_=(\uparrow\downarrow)$-bisimulations and $\uparrow\downarrow$-quasi ultraproducts. Then there exists a third class K which is definable by an XPath$_=(\uparrow\downarrow)$-node expression, contains K_1 and is disjoint from K_2.*

1.4 Binary bisimulations

We introduce notions of binary bisimulations for the downward and vertical fragments. These notions are suitable in the sense that they capture the idea of *indistinguishability by path expressions*. For the case of the downward fragment, we show a van Benthem-like characterization theorem.

1.4.1 Downward

Some facts about path expressions over XPath$_=(\downarrow)$

The proofs of Theorem 7 or Theorem 9 of [44] assume that node expressions of XPath$_=(\downarrow)$ do not contain any $\cup$. Indeed, as explained in §I.1.2, any $\cup$ of a path expression can be simulated with a $\vee$ within a suitable node expression. However, we have seen that it is not true that any XPath$_=(\downarrow)$-path expression is equivalent to a $\cup$-free one. Hence, in our context of studying a notion of binary bisimulation which captures the idea of *indistinguishability by path expressions*, we need to develop first some results that allow us to deal with the $\cup$ operator. Another difference with respect of the previous work is that there is no intersection nor complementation of path expressions. But, as we will see next we can define them under certain contexts within the language of XPath$_=(\downarrow)$.

Definition 41. If α is of the form $\alpha = [\varphi_0] \downarrow [\varphi_1] \downarrow \ldots \downarrow [\varphi_n]$, we say that it is in **simple normal form**, and we say that the **length** of α (notated $\mathsf{len}(\alpha)$) is n.

Fact 42. *For each $\cup$-free XPath$_=(\downarrow)$-path expression α there is an XPath$_=(\downarrow)$-path expression β in simple normal form such that $\mathsf{dd}(\beta) = \mathsf{dd}(\alpha)$ and for all data tree $\mathcal{T}$, we have $[\![\alpha]\!]^{\mathcal{T}} = [\![\beta]\!]^{\mathcal{T}}$.*

Fact 43. *If α is a $\cup$-free XPath$_=(\downarrow)$-path expression then $\mathcal{T}, x, y \models \alpha$ implies $x \xrightarrow{n} y$ in $\mathcal{T}$, where $n = \mathsf{len}(\alpha)$.*

We observe that the $\cup$ operator is unessential for distinguishing two pairs of nodes:

Lemma 44. *If $\mathcal{T}, x, y \models \alpha$ and $\mathcal{T}', x', y' \not\models \alpha$ then there is a $\cup$-free XPath$_=(\downarrow)$-path expression $\tilde{\alpha}$ with $\mathsf{dd}(\tilde{\alpha}) \leq \mathsf{dd}(\alpha)$ such that $\mathcal{T}, x, y \models \tilde{\alpha}$ and $\mathcal{T}', x', y' \not\models \tilde{\alpha}$.*

Proof. We show it by induction on α. The only interesting case is when $\alpha = \alpha_1 \cup \alpha_2$. Since $\mathcal{T}, x, y \models \alpha$ then there is $i \in \{1, 2\}$ such that $\mathcal{T}, x, y \models \alpha_i$. Since $\mathcal{T}', x', y' \not\models \alpha$ then $\mathcal{T}', x', y' \not\models \alpha_i$. By inductive hypothesis there is $\tilde{\alpha}_i$ which is $\cup$-free and such that $\mathcal{T}, x, y \models \tilde{\alpha}_i$ and $\mathcal{T}', x', y' \not\models \tilde{\alpha}_i$. □

The following lemma gives us a restricted form of negation for path expressions:

Lemma 45. *Let $x \xrightarrow{n} y$ in $\mathcal{T}$ and $x' \xrightarrow{n} y'$ in $\mathcal{T}'$. If α is an $\cup$-free XPath$_=(\downarrow)$-path expression such that $\mathcal{T}, x, y \models \alpha$ and $\mathcal{T}', x', y' \not\models \alpha$ then there is a $\cup$-free path expression $\overline{\alpha}$ such that $\mathsf{dd}(\alpha) = \mathsf{dd}(\overline{\alpha})$ and $\mathcal{T}, x, y \not\models \overline{\alpha}$ and $\mathcal{T}', x', y' \models \overline{\alpha}$.*

Proof. By Fact 42 we can assume that α is in simple normal form, say $\alpha = [\varphi_0] \downarrow [\varphi_1] \downarrow \ldots \downarrow [\varphi_n]$. Let $x = x_0 \to x_1 \to \ldots \to x_n = y$ and $x' = x'_0 \to x'_1 \to \ldots \to x'_n = y'$. Since $\mathcal{T}, x, y \models \alpha$ and $\mathcal{T}', x', y' \not\models \alpha$ there is i such that $x_i \models \varphi_i$ and $x'_i \not\models \varphi_i$. One can check that $\overline{\alpha} = \downarrow^i [\neg\varphi_i] \downarrow^{n-i}$ is as we wanted. □

The following lemma simplifies many of the proofs, and it will be used frequently and without mention.

Lemma 46. *If α is a XPath$_=(\downarrow)$-path expression, it is equivalent to a XPath$_=(\downarrow)$-path expression of the form $\beta_1 \cup \cdots \cup \beta_n$, with the β_i in simple normal form.*

Definition 47. If $\alpha = [\varphi_0] \downarrow [\varphi_1] \downarrow \ldots \downarrow [\varphi_i]$ and $\beta = [\psi_0] \downarrow [\psi_1] \downarrow \ldots \downarrow [\psi_i]$ are XPath$_=(\downarrow)$-path expressions of the same length in simple normal form, we define the **intersection** of α and β as

$$\alpha \cap \beta := [\varphi_0 \wedge \psi_0] \downarrow [\varphi_1 \wedge \psi_1] \downarrow \ldots \downarrow [\varphi_i \wedge \psi_i]. \tag{17}$$

Fact 48. *If α and β are XPath$_=(\downarrow)$-path expressions in simple normal form of the same length, then $\mathsf{dd}(\alpha \cap \beta) = \max\{\mathsf{dd}(\alpha), \mathsf{dd}(\beta)\}$, and for every data tree $\mathcal{T}$, we have $[\![\alpha \cap \beta]\!]^{\mathcal{T}} = [\![\alpha]\!]^{\mathcal{T}} \cap [\![\beta]\!]^{\mathcal{T}}$.*

Equivalence for XPath$_=(\downarrow)$-path expressions

For a data tree $\mathcal{T}$, let us define

$$D(\mathcal{T}) = \{(u, v) \in T^2 \mid u \xrightarrow{*} v\},$$

and for $\ell \geq 0$,

$$D_\ell(\mathcal{T}) = \{(u, v) \in T^2 \mid u \xrightarrow{\leq \ell} v\}.$$

We say that $(x, y) \in D(\mathcal{T})$ and $(x', y') \in D(\mathcal{T}')$ are **equivalent for XPath$_=$(↓)-path expressions** (notated $\mathcal{T}, x, y \equiv^{\downarrow} \mathcal{T}', x', y'$) iff the truth value of any path expression $\alpha \in \text{XPath}_=(\downarrow)$ coincides over both two-pointed data trees. That is:

$$\mathcal{T}, x, y \equiv^{\downarrow} \mathcal{T}', x', y' \overset{\text{def}}{\Leftrightarrow} \text{ for all XPath}_=(\downarrow)\text{-path expressions } \alpha,\ \mathcal{T}, x, y \models \alpha \text{ iff } \mathcal{T}', x', y' \models \alpha.$$

We say that $(x, y) \in D_\ell(\mathcal{T})$ and $(x', y') \in D_\ell(\mathcal{T}')$ are **ℓ-equivalent for XPath$_=$(↓)-path expressions** (notated $\mathcal{T}, x, y \equiv^{\downarrow}_\ell \mathcal{T}', x', y'$) iff the truth value of any path expression $\alpha \in \text{XPath}_=(\downarrow)$ with $\mathsf{dd}(\alpha) \leq \ell$ coincides over both two-pointed data trees. That is:

$$\mathcal{T}, x, y \equiv^{\downarrow}_\ell \mathcal{T}', x', y' \quad \overset{\text{def}}{\Leftrightarrow} \quad \begin{array}{l}\text{for all XPath}_=(\downarrow)\text{-path expressions } \alpha \text{ with } \mathsf{dd}(\alpha) \leq \ell,\\ \mathcal{T}, x, y \models \alpha \text{ iff } \mathcal{T}', x', y' \models \alpha.\end{array}$$

Of course, one could have defined $\mathcal{T}, x, y \equiv^{\downarrow} \mathcal{T}', x', y'$ even for pairs $(x, y) \notin D(\mathcal{T})$ or for pairs $(x', y') \notin D(\mathcal{T}')$. For instance, if x is not an ancestor of y then $\mathcal{T}, x, y$ does not verify *any* path expression, and so one could say that $\mathcal{T}, x, y \equiv^{\downarrow} \mathcal{T}', x', y'$ only when x', y' does not verify any path expression (in other words, when x' is not an ancestor of y', i.e. $(x', y') \notin D(\mathcal{T})$). We restricted the equivalences $\equiv$ to $D(\mathcal{T}) \times D(\mathcal{T}')$ for reasons of clarity when comparing logical equivalence with binary bisimulations, as we will see next.

Notice that if $\mathcal{T}, x, y \equiv^{\downarrow} \mathcal{T}', x', y'$ and $x \xrightarrow{n} y$ then $\mathcal{T}, x, y \models \downarrow^n$ and hence $\mathcal{T}', x', y' \models \downarrow^n$, which means $x' \xrightarrow{n} y'$ in $\mathcal{T}'$. The same holds in case $\mathcal{T}, x, y \equiv^{\downarrow}_\ell \mathcal{T}', x', y'$ when $n \leq \ell$.

Lemma 49. *Let $u \xrightarrow{m} v$ in $\mathcal{T}$ and $u' \xrightarrow{n} v'$ in $\mathcal{T}'$, and let $n, m \leq \ell$. If $\mathcal{T}, u, v \not\equiv^{\downarrow}_\ell \mathcal{T}', u', v'$ then there is a $\cup$-free XPath$_=$(↓)-path expression α such that $\mathsf{dd}(\alpha) \leq \ell$, $\mathcal{T}, u, v \models \alpha$ and $\mathcal{T}', u', v' \not\models \alpha$.*

Proof. If $n \neq m$ then $\mathcal{T}, u, v \models \downarrow^m$ and $\mathcal{T}', u', v' \not\models \downarrow^m$. Suppose that $n = m$ and that there is an XPath$_=$(↓)-path expression α, $\mathsf{dd}(\alpha) \leq \ell$, such that $\mathcal{T}, u, v \models \alpha$ and $\mathcal{T}', u', v' \not\models \alpha$. By Lemma 44, α can be taken $\cup$-free and we are done. The same argument applies in case $\mathcal{T}, u, v \not\models \alpha$ and $\mathcal{T}', u', v' \models \alpha$, via Lemma 45. □

Proposition 50. *$\equiv^{\downarrow}_\ell$ has finite index in the context of path expressions, that is, there are finitely many non-equivalent path expressions of downward depth at most ℓ.*

Proof. Let qr be the quantifier rank of a first-order formula, i.e., the depth of nesting of its quantifiers. It can be easily shown by induction that for any path expression α of XPath$_=$(↓) with bounded downward depth and unnecessary uses of ε (recall that $\alpha\varepsilon\beta \equiv \alpha\beta$) we have that $\mathsf{qr}(\text{Tr}_{x,y}(\alpha))$ is bounded (recall that Tr is the translation into first-order logic given in §I.1.3). It is a well-known result of first-order that there are finitely many nonequivalent formulas of bounded quantifier rank. Hence there are finitely many nonequivalent node expressions of bounded downward depth. □

Corollary 51. *Suppose $u \xrightarrow{n} v$, with $n \leq \ell$. Then $\{\mathcal{T}', u', v' \mid \mathcal{T}, u, v \equiv^{\downarrow}_\ell \mathcal{T}', u', v'\}$ is definable by an ℓ-XPath$_=$(↓)-path expression $\gamma_{\ell,\mathcal{T},u,v}$.*

Proof. Let

$$A = \{\alpha \mid \mathcal{T}, u, v \models \alpha,\ \alpha \text{ is } \cup\text{-free and } \mathsf{dd}(\alpha) \leq \ell\}.$$

First, observe that by Fact 42, each $\alpha \in A$ can be written in simple normal form, and all of them have the same length. Hence it makes sense to take the intersection between finitely many elements of A. Second, notice that by Proposition 50 there are finitely many non-equivalent $\alpha \in A$, and hence the infinite intersection $\beta = \bigcap A$ is equivalent to a finite one.

It is clear by Fact 48 that $\mathsf{dd}(\beta) \leq \ell$ and that $\mathcal{T}, u, v \models \beta$. Let us show that

$$\mathcal{T}', u', v' \models \beta \quad \text{iff} \quad \mathcal{T}, u, v \equiv_\ell^{\downarrow} \mathcal{T}', u', v'.$$

The right-to-left direction is straightforward. For the left-to-right direction, suppose by contradiction that $\mathcal{T}', u', v' \models \beta$ and $\mathcal{T}, u, v \not\equiv_\ell^{\downarrow} \mathcal{T}', u', v'$. By hypothesis, $\mathcal{T}, u, v \models \downarrow^n$ (where $n \leq \ell$), and thus, since $\mathcal{T}', u', v' \models \beta$, we have $\mathcal{T}', u', v' \models \downarrow^n$. By Lemma 49, there is a $\cup$-free XPath$_=(\downarrow)$-path expression γ such that $\mathsf{dd}(\gamma) \leq \ell$ and $\mathcal{T}, u, v \models \gamma$ and $\mathcal{T}', u', v' \not\models \gamma$. Since $\gamma \in A$ and $\mathcal{T}', u', v' \models \beta$ then $\mathcal{T}', u', v' \models \gamma$, which is a contradiction. □

Binary bisimulation for XPath$_=(\downarrow)$

We introduce a new notion of binary bisimulation between pairs of nodes (x, y) in one data-tree $\mathcal{T}$ and pairs of nodes (x', y') in another data tree $\mathcal{T}'$. For simplicity we only define binary bisimulation as a relation in $D(\mathcal{T}) \times D(\mathcal{T}')$. But it can be naturally extended to $T^2 \times T'^2$ if the definitions of $\equiv$ are likewise extended.

We say that a relation $Z \subseteq D(\mathcal{T}) \times D(\mathcal{T}')$ is a **binary XPath$_=(\downarrow)$-bisimulation**, or simply an **XPath$_=(\downarrow)$-bisimulation** when the binary context is clear, if for all $x, y \in T$ and $x', y' \in T'$ we have:

- (**Harmony**) If $(x, y)Z(x', y')$ then $label(x) = label(x')$.
- (**Equidistance**) If $(x, y)Z(x', y')$ then there is k such that $x \xrightarrow{k} y$ and $x' \xrightarrow{k} y'$.
- (**Split**) If $(x, y)Z(x', y')$, $x \xrightarrow{n} z \xrightarrow{m} y$ and $x' \xrightarrow{n} z' \xrightarrow{m} y'$ then $(x, z)Z(x', z')$ and $(z, y)Z(z', y')$.
- (**Zig**) If $(x, y)Z(x', y')$, $x \xrightarrow{n} v$ and $x \xrightarrow{m} w$ then there are $v', w' \in T'$ such that: $x' \xrightarrow{n} v'$; $x' \xrightarrow{m} w'$; $(x, v)Z(x', v')$; $(x, w)Z(x', w')$; and $data(v) = data(w)$ iff $data(v') = data(w')$.
- (**Zag**) If $(x, y)Z(x', y')$, $x' \xrightarrow{n} v'$ and $x' \xrightarrow{m} w'$ then there are $v, w \in T$ such that: $x \xrightarrow{n} v$; $x \xrightarrow{m} w$; $(x, v)Z(x', v')$; $(x, w)Z(x', w')$; and $data(v) = data(w)$ iff $data(v') = data(w')$.

In Figure 18 we show an example of the process of checking **Split** for a particular pair of pairs of nodes in a binary XPath$_=(\downarrow)$-bisimulation, and in Figure 19 we show an example of checking **Zig**.

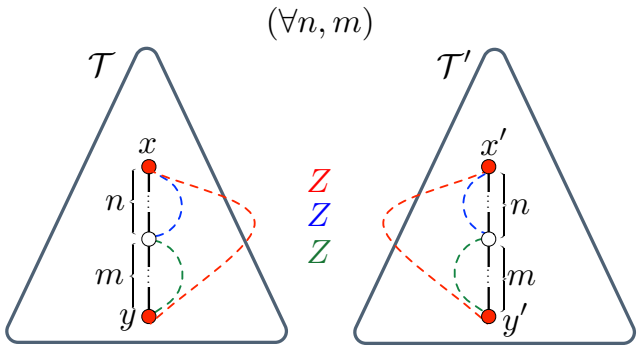


Figure 18: The process of checking the condition **Split** of binary XPath$_=(\downarrow)$-bisimulation for $(x, y)Z(x', y')$ entails checking that all divisions of the path between x and y into two different paths have a corresponding division into two paths between x' and y'. As there are different pairs with nodes in common, to avoid graphical superposition we represent pairs of nodes as a dashed line between them.

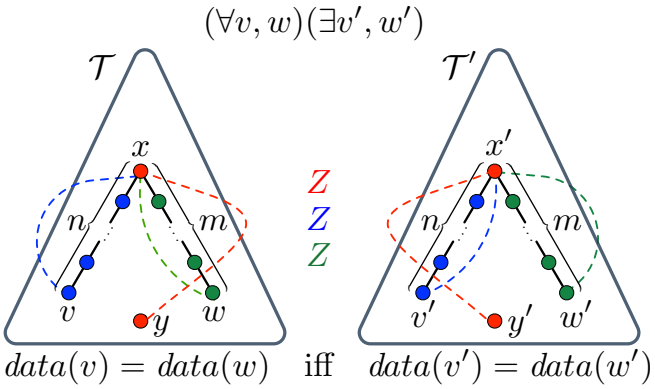


Figure 19: The process of checking **Zig** for a binary XPath$_=(\downarrow)$-bisimulation Z between $\mathcal{T}$ and $\mathcal{T}'$ for $(x, y)Z(x', y')$. The conditions we need to verify are akin to those of checking in an unary XPath$_=(\downarrow)$-bisimulation the condition **Zig** for (x, x'); however, we only have to check that the paths themselves (and not every intermediate node) are connected via Z.

We define when two pairs of nodes $(t, u) \in D(\mathcal{T})$[12] and $(t', u') \in D(\mathcal{T}')$ are said to be **XPath$_=$(↓)-bisimilar**, notated: $\mathcal{T}, t, u \leftrightarrows^{\downarrow} \mathcal{T}', t', u'$:

$\mathcal{T}, t, u \leftrightarrows^{\downarrow} \mathcal{T}', t', u' \overset{\text{def}}{\Leftrightarrow}$ there is binary XPath$_=$(↓)-bisimulation Z such that $(t, u)Z(t', u')$.

See Figure 20 for an example of a binary XPath$_=$(↓)-bisimilarity between two different pairs of nodes on the same data tree.

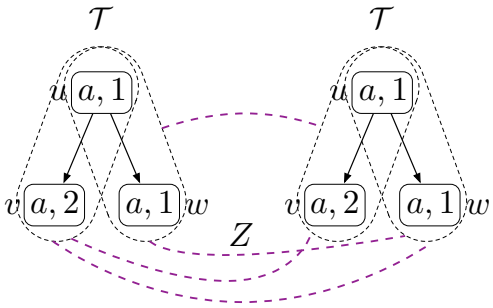


Figure 20: $\mathcal{T}, u, w \leftrightarrows^{\downarrow} \mathcal{T}, u, v$, as witnessed by the represented binary XPath$_=$(↓)-bisimulation Z between the same data tree $\mathcal{T}$. Pairs $(u, u)Z(u, u)$, $(v, v)Z(v, v)$, $(w, w)Z(w, w)$, $(w, w)Z(v, v)$ and $(v, v)Z(w, w)$ are not shown.

We say that a family of relations $(Z_j)_{j \leq \ell}$ in $D_j(\mathcal{T}) \times D_j(\mathcal{T}')$ forms a **binary ℓ-bisimulation**, or simply an **ℓ-bisimulation** when the binary context is clear, if for all $j \leq \ell$, $(x, y) \in D_j(\mathcal{T})$ and $(x', y') \in D_j(\mathcal{T}')$ we have:

- (**Harmony**) If $(x, y)Z_j(x', y')$ then $label(x) = label(x')$.
- (**Equidistance**) If $(x, y)Z_j(x', y')$ then there is $k \leq j$ such that $x \xrightarrow{k} y$ and $x' \xrightarrow{k} y'$.
- (**Split**) If $(x, y)Z_j(x', y')$, $x \xrightarrow{n} z \xrightarrow{m} y$ and $x' \xrightarrow{n} z' \xrightarrow{m} y'$ then $(x, z)Z_j(x', z')$ and $(z, y)Z_{j-n}(z', y')$.
- (**Zig**) If $(x, y)Z_j(x', y')$, $x \xrightarrow{n} v$ and $x \xrightarrow{m} w$, with $n, m \leq j$, then there are $v', w' \in T'$ such that: $x' \xrightarrow{n} v'$, $x' \xrightarrow{m} w'$, $(x, v)Z_j(x', v')$, $(x, w)Z_j(x', w')$, and $data(v) = data(w)$ iff $data(v') = data(w')$.
- (**Zag**) If $(x, y)Z_j(x', y')$, $x' \xrightarrow{n} v'$ and $x' \xrightarrow{m} w'$, with $n, m \leq j$, then there are $v, w \in T$ such that: $x \xrightarrow{n} v$, $x \xrightarrow{m} w$, $(x, v)Z_j(x', v')$, $(x, w)Z_j(x', w')$, and $data(v) = data(w)$ iff $data(v') = data(w')$.

[12] The definition can be extended to $(u, v) \in \mathcal{T} \times \mathcal{T}$ if the definition of $\equiv^{\downarrow}$ is likewise extended. Observe that whenever $u \xrightarrow{*} v$ and $u' \xrightarrow{*} v'$ do not hold, $\mathcal{T}, u, v \models \alpha \Leftrightarrow \mathcal{T}', u', v' \models \alpha$ is trivially true for all XPath$_=$(↓)-path expressions α.

Notice that, because of the **Split** condition, the rules **Zig** and **Zag** for binary bisimulations only require Z to relate (x, v) and (x', v') on one hand and (x, w) and (x', w') on the other, instead of relating *all* nodes along the path from x to v to the corresponding nodes in the path from x' to v', and the same for the paths from x to w and x' to w'.

We now define when $(t, u) \in D_\ell(\mathcal{T})$ is said to be **ℓ-bisimilar** to $(t', u') \in D_\ell(\mathcal{T}')$, notated $\mathcal{T}, t, u \leftrightarrows^{\downarrow}_\ell \mathcal{T}', t', u'$:

$$\mathcal{T}, t, u \leftrightarrows^{\downarrow}_\ell \mathcal{T}', t', u' \stackrel{\text{def}}{\Leftrightarrow} \text{ there is a binary } \ell\text{-bisimulation } (Z_j)_{j\le\ell} \text{ such that } (t, u) Z_\ell (t', u')$$

Recall that for a data tree $\mathcal{T}$ and $u \in T$, $\mathcal{T}|_u$ denotes the subtree of $\mathcal{T}$ induced by $\{v \in T \mid (\exists n)\ u \xrightarrow{n} v\}$. Observe that the root of $\mathcal{T}|_u$ is u. The following results are straightforward consequences of the definition of binary bisimulation:

Proposition 52. *If $(u, v) \in D(\mathcal{T})$ then $\mathcal{T}, u, v \leftrightarrows^{\downarrow} (\mathcal{T}|_u), u, v$.*

Proposition 53. *If $\mathcal{T}$ is a subtree of $\mathcal{T}'$ and $(u, v) \in D(\mathcal{T})$ then $\mathcal{T}, u, v \leftrightarrows^{\downarrow} \mathcal{T}', u, v$.*

For a data tree $\mathcal{T}$ and $u \in T$, we denote $\mathcal{T}|_{\ell u}$ as:

$$\mathcal{T}|_{\ell u} \stackrel{\text{def}}{=} \text{ the subtree of } \mathcal{T} \text{ induced by } \{v \in T \mid (\exists n \le \ell)\ u \xrightarrow{n} v\}.$$

Proposition 54. *If $(u, v) \in D_\ell(\mathcal{T})$ then $\mathcal{T}, u, v \leftrightarrows^{\downarrow}_\ell (\mathcal{T}|_{\ell u}), u, v$.*

Proof. Define the family $(Z_j)_{j\le\ell}$, $Z_j \subseteq D_j(\mathcal{T}|_{\ell u}) \times D_j(\mathcal{T})$ as following: given $j \le \ell$, if $x \xrightarrow{\le j} y$ and $u \xrightarrow{\ell-j} x$, then $(x, y) Z_j (x, y)$ (observe that $j = \ell - (\ell - j)$; intuitively, start with a Z_ℓ which matches all identical pairs of nodes in $D(\mathcal{T}|_{\ell u})$, then consider $Z_{\ell-1}$ the subset where the first coordinate of the pairs must be at a downward distance of 1 from u, and so on). We see that this family defines an ℓ-bisimulation. It is clear that **Harmony** holds, as $(x, y) Z_j (w, z)$ implies $x = w, y = z$. **Equidistance** holds for similar reasons, as also $(x, y) Z_j (x, y)$ implies that $x \xrightarrow{\le j} y$. For **Split**, let $(x, y) Z_j (x, y)$ and $x \xrightarrow{m} z \xrightarrow{n} y$ and $x \xrightarrow{m} z' \xrightarrow{n} y$. Note that it must be $z = z'$. We want to see that $(x, z) Z_j (x, z)$ and $(z, y) Z_{j-m} (z, y)$. That $(x, z) Z_j (x, z)$ follows from the definition of Z_j, as $x \xrightarrow{\le j} y$ and then $x \xrightarrow{\le j} z$. $(z, y) Z_{j-m} (z, y)$ follows from the definition of Z_{j-m} and the fact that if $u \xrightarrow{\ell-j} x$ and $x \xrightarrow{\le j} y$ then $u \xrightarrow{\ell-(j-m)} x$ and $z \xrightarrow{\le j-m} y$. Now it only remains to verify **Zig** and **Zag**. We only prove **Zag**, as **Zig** is simpler. Let $(x, y) Z_j (x, y)$, $x \xrightarrow{m} v'$, $x \xrightarrow{n} w'$, $m, n \le j$, and $data(v') \star data(w')$, with $\star \in \{=, \neq\}$. We want to prove that there are $v, w \in \mathcal{T}|_{\ell u}$ such that $x \xrightarrow{m} v$, $x \xrightarrow{n} w$, $(x, v) Z_j (x', v')$, $(x, w) Z_j (x', w')$, and $data(v) \star data(w)$. Now, $(x, y) Z_j (x, y)$ implies that $u \xrightarrow{\ell-j} x$ and thus (since $m, n \le j$) $u \xrightarrow{\le \ell} v'$, $u \xrightarrow{\le \ell} w'$. But then v' and w' are in $\mathcal{T}|_{\ell u}$, and the result follows from taking $v = v'$ and $w = w'$. □

Proposition 55. *Suppose $\mathcal{T}$ and $\mathcal{T}'$ have height at most ℓ, $(u, v) \in D_\ell(\mathcal{T})$, and $(u', v') \in D_\ell(\mathcal{T}')$. Then $\mathcal{T}, u, v \leftrightarrows^{\downarrow}_\ell \mathcal{T}', u', v'$ iff $\mathcal{T}, u, v \leftrightarrows^{\downarrow} \mathcal{T}', u', v'$.*

We now show that in the new context of path expressions of $\mathrm{XPath}_{=}(\downarrow)$ we have an analog of Theorem 7 for binary bisimulations and path equivalence, i.e., $\leftrightarrows^{\downarrow}$ coincides with $\equiv^{\downarrow}$ on finitely branching data trees, and $\leftrightarrows^{\downarrow}_{\ell}$ always coincides with $\equiv^{\downarrow}_{\ell}$.

Theorem 56. *Let $\mathcal{T}, u, v$ and $\mathcal{T}', u', v$ be two-pointed data trees. Then:*

1. *$\mathcal{T}, u, v \leftrightarrows^{\downarrow} \mathcal{T}', u', v'$ implies $\mathcal{T}, u, v \equiv^{\downarrow} \mathcal{T}', u', v'$. The converse also holds when $\mathcal{T}$ and $\mathcal{T}'$ are finitely branching.*
2. *$\mathcal{T}, u, v \leftrightarrows^{\downarrow}_{\ell} \mathcal{T}', u', v'$ iff $\mathcal{T}, u, v \equiv^{\downarrow}_{\ell} \mathcal{T}', u', v'$.*

Item 2 of the above theorem is a consequence of the next two propositions. Item 1 can be shown analogously (the set P that will appear in the proof of Proposition 58 for showing **Zig** is finite when $\mathcal{T}'$ is finitely branching, and its version over $\mathcal{T}$ for showing **Zag** is finite when $\mathcal{T}$ is finitely branching).

Proposition 57. *If $\mathcal{T}, t, u \leftrightarrows^{\downarrow}_{\ell} \mathcal{T}', t', u'$ then $\mathcal{T}, t, u \equiv^{\downarrow}_{\ell} \mathcal{T}, t', u'$.*

Proof. We actually show that if $\mathcal{T}, t, u \leftrightarrows^{\downarrow}_{\ell} \mathcal{T}', t', u'$ via $(Z_i)_{i \leq \ell}$ then for all $0 \leq n \leq j \leq \ell$, for all φ with $\mathsf{dd}(\varphi) \leq j$, and for all α with $\mathsf{dd}(\alpha) \leq j$:

1. If $(x, x) Z_j (x', x')$ then $\mathcal{T}, x \models \varphi$ iff $\mathcal{T}', x' \models \varphi$,
2. If $(x, y) Z_j (x', y')$ then $\mathcal{T}, x, y \models \alpha$ iff $\mathcal{T}', x', y' \models \alpha$.

We show 1 and 2 by induction on $|\varphi| + |\alpha|$.

Let us see item 1. The base case is $\varphi = a$ for some $a \in \mathbb{A}$. By **Harmony**, $\mathit{label}(x) = \mathit{label}(x')$ and then $\mathcal{T}, x \models \varphi$ iff $\mathcal{T}', x' \models \varphi$. The Boolean cases for φ are straightforward.

Suppose $\varphi = \langle \alpha = \beta \rangle$. We will show $\mathcal{T}, x \models \varphi \Rightarrow \mathcal{T}', x' \models \varphi$, so assume $\mathcal{T}, x \models \varphi$. Suppose there are $v, w \in T$ and $n, m \leq j$ such that $x \xrightarrow{n} v$, $x \xrightarrow{m} w$, $\mathcal{T}, x, v \models \alpha$, $\mathcal{T}, x, w \models \beta$ and $\mathit{data}(v) = \mathit{data}(w)$. By **Zig**, there are $v', w' \in T'$ such that $x' \xrightarrow{n} v'$, $x' \xrightarrow{m} w'$, $(x, v) Z_j (x', v')$, $(x, w) Z_j (x', w')$ and $\mathit{data}(v') = \mathit{data}(w')$. By inductive hypothesis 2 (twice), $\mathcal{T}', x', v' \models \alpha$ and $\mathcal{T}', x', w' \models \beta$. Hence $\mathcal{T}', x' \models \varphi$. The implication $\mathcal{T}', x' \models \varphi \Rightarrow \mathcal{T}, x \models \varphi$ is analogous. The case $\varphi = \langle \alpha \neq \beta \rangle$ is shown similarly.

Let us now analyze item 2. We only show the 'only if' direction, as the 'if' is analogous. The base case is when $\alpha \in \{\varepsilon, \downarrow\}$. If $\alpha = \varepsilon$, we have:

$$\begin{aligned} \mathcal{T}, x, y \models \alpha \quad & \text{iff} \quad x \xrightarrow{0} y \\ & \text{iff} \quad x' \xrightarrow{0} y' \qquad \text{(\textbf{Equidistance})} \\ & \text{iff} \quad \mathcal{T}', x', y' \models \alpha \end{aligned}$$

If $\alpha = \downarrow$, we have the same argument but with $\xrightarrow{1}$ instead of $\xrightarrow{0}$. For the inductive step, suppose $\alpha = \beta\gamma$ and assume $\mathcal{T}, x, y \models \alpha$. Then there is $z \in T$ such that $x \xrightarrow{n} z \xrightarrow{m} y$, $\mathcal{T}, x, z \models \beta$ and $\mathcal{T}, z, y \models \gamma$. By **Split** we have $(x, z) Z_j (x', z')$ and $(z, y) Z_{j-n} (z', y')$ Observe that $\mathsf{dd}(\beta) \leq \mathsf{dd}(\alpha) \leq j$ and $\mathsf{dd}(\gamma) \leq \mathsf{dd}(\alpha) - n \leq j - n$. where z' is the only node such that

$x' \xrightarrow{n} z' \xrightarrow{m} y'$ (observe that by **Equidistance**, $x' \xrightarrow{n+m} y'$). By inductive hypothesis 2 (again, twice), we conclude $\mathcal{T}', x', z' \models \beta$ and $\mathcal{T}', z', y' \models \gamma$, and hence $\mathcal{T}', x', y' \models \alpha$.

Suppose $\alpha = \alpha_1 \cup \alpha_2$ and assume $\mathcal{T}, x, y \models \alpha$. We have $\mathcal{T}, x, y \models \alpha_i$ for some $i \in \{1, 2\}$. By inductive hypothesis we have $\mathcal{T}', x', y' \models \alpha_i$, and so $\mathcal{T}', x', y' \models \alpha$

Finally, suppose $\alpha = [\varphi]$ and assume $\mathcal{T}, x, y \models \alpha$. By semantics, we must have $x = y$ and $\mathcal{T}, x \models \varphi$. By inductive hypothesis, $\mathcal{T}', x' \models \varphi$, and by **Equidistance** we have $x' = y'$. Hence we conclude $\mathcal{T}', x', y' \models \alpha$. □

Proposition 58. *If* $\mathcal{T}, t, u \equiv_\ell^\downarrow \mathcal{T}', t', u'$ *then* $\mathcal{T}, t, u \leftrightarrows_\ell^\downarrow \mathcal{T}', t', u'$.

Proof. Fix $(t, u) \in D_\ell(\mathcal{T})$ and $(t', u') \in D_\ell(\mathcal{T}')$ such that $\mathcal{T}, t, u \equiv_\ell^\downarrow \mathcal{T}', t', u'$. Define $(Z_j)_{j \leq \ell}$ by

$$(x, y) Z_j (x', y') \quad \text{iff} \quad \mathcal{T}, x, y \equiv_j^\downarrow \mathcal{T}', x', y'$$

for all $(x, y) \in D_\ell(\mathcal{T})$ and all $(x', y') \in D_\ell(\mathcal{T}')$. We show that $(Z_j)_{j \leq \ell}$ is an ℓ-bisimulation between $\mathcal{T}, u, v$ and $\mathcal{T}', u', v'$.

By hypothesis, $(t, u) Z_\ell (t', u')$. To check all the rules of ℓ-bisimulation for $\mathrm{XPath}_=(\downarrow)$, suppose $x \xrightarrow{k} y$ for some $k \leq j$, and assume $(x, y) Z_j (x', y')$. To see **Harmony**, let $a = \mathit{label}(x)$ and let $\alpha = [a]\downarrow^k$, of downward depth $k \leq j$. It is clear that $\mathcal{T}, x, y \models \alpha$, and so $\mathcal{T}, x', y' \models \alpha$, which means that $\mathit{label}(x') = a$. The implication $\mathit{label}(x') = a \Rightarrow \mathit{label}(x) = a$ is seen analogously.

For **Equidistance**, since $\mathcal{T}, x, y \models \downarrow^k$, then $\mathcal{T}', x', y' \models \downarrow^k$, and so $x' \xrightarrow{k} y'$. The implication $x' \xrightarrow{k} y' \Rightarrow x \xrightarrow{k} y$ is seen analogously.

Let us see **Split**. Suppose $x \xrightarrow{n} z \xrightarrow{m} y$ and $x' \xrightarrow{n} z' \xrightarrow{m} y'$, where $k = m + n \leq j$. We prove that:

1. $\mathcal{T}, x, z \equiv_j^\downarrow \mathcal{T}', x', z'$ and
2. $\mathcal{T}, z, y \equiv_{j-n}^\downarrow \mathcal{T}', z', y'$.

To see 1, assume by contradiction that α is path expression with $\mathsf{dd}(\alpha) \leq j$ such that $\mathcal{T}, x, z \models \alpha$ and $\mathcal{T}', x', z' \not\models \alpha$ (the other case is analogous). Observe that $\mathsf{len}(\alpha) = n$. Now, $\mathcal{T}, x, y \models \alpha\downarrow^m$ and $\mathcal{T}', x', y' \not\models \alpha\downarrow^m$. But $\mathsf{dd}(\alpha\downarrow^m) = \max\{\mathsf{dd}(\alpha), m + \mathsf{len}(\alpha)\} \leq j$, so, since $\mathcal{T}, x, y \equiv_j^\downarrow \mathcal{T}', x', y'$, we have $\mathcal{T}', x', y' \models \alpha \downarrow^m$, a contradiction.

To see 2, assume by contradiction that α is a path expression with $\mathsf{dd}(\alpha) \leq j - n$ such that $\mathcal{T}, z, y \models \alpha$ and $\mathcal{T}', z', y' \not\models \alpha$ (the other case is analogous). Observe that $\mathsf{len}(\alpha) = m$. Now, $\mathcal{T}, x, y \models\downarrow^n \alpha$ and $\mathcal{T}', x', y' \not\models \downarrow^n\alpha$. But $\mathsf{dd}(\downarrow^n\alpha) = n + \mathsf{dd}(\alpha) \leq n + j - n = j$, so, since $\mathcal{T}, x, y \equiv_j^\downarrow \mathcal{T}', x', y'$, we have $\mathcal{T}', x', y' \models\downarrow^n \alpha$, a contradiction.

Finally, let us show **Zig** (the case for **Zag** is analogous). Suppose $x \xrightarrow{n} v$, $x \xrightarrow{m} w$, where $n, m \leq j$, and $\mathit{data}(v) = \mathit{data}(w)$ (the case $\neq$ is analogous).

Let $P \subseteq T'^2$ be defined by:

$$P = \{(v', w') \mid x' \xrightarrow{n} v' \wedge x' \xrightarrow{m} w' \wedge \mathit{data}(v') = \mathit{data}(w')\}.$$

Observe that $\mathcal{T}, x \models \langle \downarrow^n = \downarrow^m \rangle$. Hence $\mathcal{T}, x, y \models [\langle \downarrow^n = \downarrow^m \rangle]\downarrow^k$ and so $\mathcal{T}', x', y' \models [\langle \downarrow^n = \downarrow^m \rangle]\downarrow^k$, which implies $\mathcal{T}', x' \models \langle \downarrow^n = \downarrow^m \rangle$. Therefore $P \neq \emptyset$.

We next show that there exists $(v', w') \in P$ such that $\mathcal{T}, x, v \equiv_j^{\downarrow} \mathcal{T}', x', v'$ and $\mathcal{T}, x, w \equiv_j^{\downarrow} \mathcal{T}', x', w'$, and hence **Zig** is satisfied by Z_j.

Suppose by way of contradiction that for all $(v', w') \in P$, either $\mathcal{T}, x, v \not\equiv_j^{\downarrow} \mathcal{T}', x', v'$ or $\mathcal{T}, x, w \not\equiv_j^{\downarrow} \mathcal{T}', x', w'$. Because of Lemma 49, for all (v', w') in P, either there exists a $\cup$-free path expression $\alpha_{v',w'}$ such that $\mathsf{dd}(\alpha_{v',w'}) \leq j$ and $\mathcal{T}, x, v \models \alpha_{v',w'}$ but $\mathcal{T}', x', v' \not\models \alpha_{v',w'}$, or there exists a path expression $\beta_{v',w'}$ such that $\mathsf{dd}(\beta_{v',w'}) \leq j$ and $\mathcal{T}, x, w \models \beta_{v',w'}$ but $\mathcal{T}', x', w' \not\models \beta_{v',w'}$.

Call A the set of pairs of the first type, and B the set of pairs of the second type.

$$\alpha = \begin{cases} \bigcap_{(v',w') \in A} \alpha_{v',w'} & \text{if } A \neq \emptyset; \\ \downarrow^n & \text{otherwise.} \end{cases} \qquad \text{and} \qquad \beta = \begin{cases} \bigcap_{(v',w') \in B} \beta_{v',w'} & \text{if } B \neq \emptyset; \\ \downarrow^m & \text{otherwise.} \end{cases}$$

Now, by Proposition 50, there are only finitely many non-equivalent path expressions of downward depth at most ℓ, so the intersections that define α and β can be considered finite. Notice that by Fact 42 we may take all the $\alpha_{v',w'}$ involved in simple normal form, and they will all have the same length (namely, n, the distance from x to v). An analog argument holds for the $\beta_{v',w'}$ expressions. Therefore, it makes sense to take the operation $\cap$ among all the $\alpha_{v',w'}$ and among all the $\beta_{v',w'}$. Let $\psi = \langle \alpha = \beta \rangle$. By construction, $\mathcal{T}, x \models \psi$, and so $\mathcal{T}, x, y \models [\psi]\downarrow^k$. Furthermore, since A or B are nonempty, $\mathcal{T}', x' \not\models \psi$, and so $\mathcal{T}', x', y' \not\models [\psi]\downarrow^k$. Since $\mathsf{dd}(\psi) \leq j$ (by Fact 48) and $k \leq j$ we have $\mathsf{dd}([\psi]\downarrow^k) = \max\{\mathsf{dd}(\psi), k\} \leq j$. Hence $\mathcal{T}, x, y \not\equiv_j \mathcal{T}', x', y'$, which is a contradiction. This concludes the proof. □

The following corollary shows that binary downward bisimulations subsume unary ones.

Corollary 59. *$\mathcal{T}, x \equiv^{\downarrow} \mathcal{T}', x'$ iff $\mathcal{T}, x, x \equiv^{\downarrow} \mathcal{T}', x', x'$. Thus, if $\mathcal{T}$ and $\mathcal{T}'$ are finitely branching, then $\mathcal{T}, x \leftrightarrows^{\downarrow} \mathcal{T}', x'$ iff $\mathcal{T}, x, x \leftrightarrows^{\downarrow} \mathcal{T}', x', x'$.*

Proof. The second part follows from the first part, item 1 of Theorem 56 and the corresponding result for nodes [44].

For the left-to-right implication, let $\mathcal{T}, x \equiv^{\downarrow} \mathcal{T}', x'$. Take $\alpha = [\varphi_0] \downarrow \ldots \downarrow [\varphi_n]$ (we can assume α has this form from Lemma 44 and Fact 42). Suppose $\mathcal{T}, x, x \models \alpha$ and let us see that $\mathcal{T}, x', x' \models \alpha$ (the other implication is analogous). We have $n = 0$ and thus $\alpha = [\varphi_0]$, so $\mathcal{T}, x \models \varphi_0$. Then $\mathcal{T}', x' \models \varphi_0$, and $\mathcal{T}', x', x' \models [\varphi_0]$.

For the right-to-left implication, assume $\mathcal{T}, x, x \equiv^{\downarrow} \mathcal{T}', x', x'$. In particular, $\mathcal{T}, x, x \models [\varphi]$ iff $\mathcal{T}', x', x' \models [\varphi]$. Since $\mathcal{T}, x, x \models [\varphi]$ iff $\mathcal{T}, x \models \varphi$ and $\mathcal{T}', x', x' \models [\varphi]$ iff $\mathcal{T}', x' \models \varphi$, we arrive to $\mathcal{T}, x \models \varphi$ iff $\mathcal{T}', x' \models \varphi$, as we wanted. □

Characterization for XPath$_=(\downarrow)$ paths

In this section we show that for each formula $\varphi(x, y)$ of first order, over the appropriate signature and with two free variables x and y: there is a path expression α of XPath$_=(\downarrow)$

such that its translation into first-order logic (given in §I.1.3) $\mathrm{Tr}_{x,y}(\alpha)$ is equivalent to $\varphi(x,y)$ if and only if φ is a 'forward property' (defined below), and it is bisimulation-invariant over data trees. We begin with some definitions.

We say that $\varphi(x,y) \in \mathrm{FO}(\sigma)$ is **$\leftrightarrows^{\downarrow}$-invariant** [resp. **$\leftrightarrows^{\downarrow}_{\ell}$-invariant**] if for all data trees $\mathcal{T}$ and $\mathcal{T}'$, $u \xrightarrow{*} v$ [resp. $u \xrightarrow{\leq \ell} v$] in $\mathcal{T}$, $u' \xrightarrow{*} v'$ [resp. $u' \xrightarrow{\leq \ell} v'$] in $\mathcal{T}'$, and $\mathcal{T}, u, v \leftrightarrows^{\downarrow} \mathcal{T}', u', v'$ [resp. $\mathcal{T}, u, v \leftrightarrows^{\downarrow}_{\ell} \mathcal{T}', u', v'$] we have $\mathcal{T} \models \varphi[u,v]$ iff $\mathcal{T}' \models \varphi[u',v']$.

A first-order σ-formula $\varphi(x,y)$ is said to be a **forward property** if for every σ-structure $\mathcal{A}$ and $u, v \in A$, we have that $\mathcal{A} \models \varphi(u,v)$ implies $u \rightsquigarrow^{*} v$ in $\mathcal{A}$. By Compactness, $\varphi(x,y)$ is a forward property iff there is k such that $\mathcal{A} \models \varphi(u,v)$ implies $u \rightsquigarrow^{\leq k} v$ in $\mathcal{A}$. In this case we say that $\varphi(x,y)$ is a ***k*-forward property**.

Recall that for a data tree $\mathcal{T}$ and $u \in T$, we denote by $\mathcal{T}|_{\ell u}$ the subtree of $\mathcal{T}$ induced by $\{v \in T \mid (\exists n \leq \ell)\ u \xrightarrow{n} v\}$. Let $k \leq \ell$. We say that a first-order formula $\varphi(x,y)$ with two free variables is **(k,ℓ)-local** whenever $\mathcal{T} \models \varphi[u,v]$ iff $\mathcal{T}|_{\ell u} \models \varphi[u,v]$ for all $(u,v) \in D_k(\mathcal{T})$.

We now state some lemmas that will be used for the proof. They all have two versions: one over all data trees and the other restricted to finite data trees.

Lemma 60. *Let $\varphi(x,y) \in \mathrm{FO}(\sigma)$ be $\leftrightarrows^{\downarrow}$-invariant over [finite] data trees. Then for each k there is ℓ (large enough, depending on the quantifier rank of φ and k) such that φ is (k,ℓ)-local.*

Proof. A straightforward modification of the proof in [44, Prop 6.2], which, in turn, follows Otto's idea [89]. □

Lemma 61. *If $\varphi(x,y) \in \mathrm{FO}(\sigma)$ is a k-forward property, $\leftrightarrows^{\downarrow}$-invariant over [finite] data trees and (k,ℓ)-local, then $\varphi(x,y)$ is $\leftrightarrows^{\downarrow}_{\ell}$-invariant.*

Proof. Since $\varphi(x,y)$ is k-forward, it suffices to show that for $\mathcal{T}, u, v$ and $\mathcal{T}', u', v'$ such that $\mathcal{T}, u, v \leftrightarrows^{\downarrow}_{\ell} \mathcal{T}', u', v'$ and $u \xrightarrow{\leq k} v$ (and so $u' \xrightarrow{\leq k} v'$) we have $\mathcal{T} \models \varphi[u,v]$ iff $\mathcal{T}' \models \varphi[u',v']$.

Now for such $\mathcal{T}, u, v$ and $\mathcal{T}', u', v'$ we have

$$\begin{aligned} \mathcal{T}, u, v \leftrightarrows^{\downarrow}_{\ell} \mathcal{T}', u', v' &\text{ iff } (\mathcal{T}|_{\ell u}), u, v \leftrightarrows^{\downarrow}_{\ell} (\mathcal{T}'|_{\ell u'}), u', v' && \text{(Prop. 54)} \\ &\text{iff } (\mathcal{T}|_{\ell u}), u, v \leftrightarrows^{\downarrow} (\mathcal{T}'|_{\ell u'}), u', v'. && \text{(Prop. 55)} \end{aligned}$$

By (k,ℓ)-locality, we have $\mathcal{T} \models \varphi[u,v]$ iff $\mathcal{T}|_{\ell u} \models \varphi[u,v]$. By $\leftrightarrows^{\downarrow}$-invariance, $\mathcal{T}|_{\ell u} \models \varphi[u,v]$ iff $\mathcal{T}'|_{\ell u'} \models \varphi[u',v']$ and by (k,ℓ)-locality again, $\mathcal{T} \models \varphi[u,v]$ iff $\mathcal{T}' \models \varphi[u',v']$. □

Lemma 62. *If $\varphi(x,y) \in \mathrm{FO}(\sigma)$ is a k-forward property which is $\leftrightarrows^{\downarrow}_{\ell}$-invariant over [finite] data trees, then there is an $XPath_{=}(\downarrow)$-path expression δ such that $\mathsf{dd}(\delta) \leq \ell$ and for all [finite] data trees $\mathcal{T}$ and $u, v \in T$ we have $\mathcal{T}, u, v \models \delta$ iff $\mathcal{T} \models \varphi[u,v]$.*

Proof. By Corollary 51, for every $\mathcal{T}, u, v$, with $u \xrightarrow{\leq \ell} v$, there is an ℓ-$\mathrm{XPath}_{=}(\downarrow)$-path expression $\gamma_{\ell,\mathcal{T},u,v}$ such that $\mathcal{T}, u, v \equiv^{\downarrow}_{\ell} \mathcal{T}', u', v'$ iff $\mathcal{T}', u', v' \models \gamma_{\ell,\mathcal{T},u,v}$. Let

$$\delta = \bigcup_{\mathcal{T} \models \varphi[u,v]} \gamma_{\ell,\mathcal{T},u,v}.$$

Since $\gamma_{\ell,\mathcal{T},u,v} \in \ell\text{-XPath}_=(\downarrow)$ and, by Proposition 50, $\equiv^{\downarrow}_{\ell}$ has finite index, it follows that δ is equivalent to a finite union.

We now show that $\varphi \equiv \mathrm{Tr}_{x,y}(\delta)$. Let us first see that $\varphi \models \mathrm{Tr}_{x,y}(\delta)$. Suppose $\mathcal{T} \models \varphi[u,v]$. Since $\varphi(x,y)$ is a k-forward property, we have $u \xrightarrow{n} v$ for some $n \leq k \leq \ell$. Since $\mathcal{T}, u, v \models \gamma_{\ell,\mathcal{T},u,v}$, we have $\mathcal{T}, u, v \models \delta$ and so $\mathcal{T} \models \mathrm{Tr}_{x,y}(\delta)[u,v]$, as we wanted to see.

Let us now see that $\mathrm{Tr}_{x,y}(\delta) \models \varphi$. Assume $\mathcal{T} \models \mathrm{Tr}_{x,y}(\delta)[u,v]$, and so $\mathcal{T}, u, v \models \delta$. Then there exists $\mathcal{T}', u', v'$ such that $\mathcal{T}' \models \varphi[u',v']$ and $\mathcal{T}, u, v \models \gamma_{\ell,\mathcal{T}',u',v'}$. By the property of $\gamma_{\ell,\mathcal{T}',u',v'}$, we have $\mathcal{T}, u, v \equiv^{\downarrow}_{\ell} \mathcal{T}', u', v'$ and since φ is $\leftrightarrows^{\downarrow}_{\ell}$-invariant (and hence $\equiv^{\downarrow}_{\ell}$-invariant by Theorem 56) we conclude $\mathcal{T} \models \varphi[u,v]$. □

The main result has two readings: one classical, and one restricted to finite models.

Theorem 63 (Characterization)**.** *Let $\varphi(x,y) \in \mathrm{FO}(\sigma)$. The following are equivalent:*

(i) $\varphi(x,y)$ is a forward property $\leftrightarrows^{\downarrow}$-invariant over [finite] data trees.

(ii) $\varphi(x,y)$ is logically equivalent over [finite] data trees to a path expression of XPath$_=(\downarrow)$.

Observe that the condition on $\varphi(x,y)$ to be a forward property is necessary. Indeed, if $\varphi(x,y)$ is universally valid then it is trivially $\leftrightarrows^{\downarrow}$-invariant over [finite] data trees, but it is clearly not XPath$_=(\downarrow)$-expressible, as its semantics includes pairs of nodes with arbitrarily large distance between them, or even pairs (x,y) where y is not descendant of x.

Proof of Theorem 63. The implication (ii) ⇒ (i) follows straightforwardly from Item 1 of Theorem 56, by using that binary XPath$_=(\downarrow)$-bisimilarity implies equivalence for XPath$_=(\downarrow)$-path expressions. The proof of (i) ⇒ (ii) goes as in the proof of [44, Th. 6.1], by Lemma 60, Lemma 61, and Lemma 62. □

1.4.2 Vertical

Some facts about path expressions over XPath$_=(\uparrow\downarrow)$

As was previewed in §1.2.3, here we prove some results related to normal forms for path expressions. Importantly for the rest of this chapter, we obtain a fragment of XPath$_=(\uparrow\downarrow)$ that is semantically equivalent to the full XPath$_=(\uparrow\downarrow)$, but whose node and path expressions are in some normal forms that greatly simplify proofs (see Remark 67). We begin by stating two needed results from [45]:

Proposition 64. *[45, Prop. 12] Given a XPath$_=(\uparrow\downarrow)$-node expression φ, there is $\varphi^{\uparrow\downarrow}$ in up-down normal form such that $\varphi \equiv \varphi^{\uparrow\downarrow}$.*

Proposition 65. *[45, Lem. 13] Given a $\cup$-free XPath$_=(\uparrow\downarrow)$-path expression α, there is $\alpha^{\uparrow\downarrow}$ in up-down normal form such that $\alpha \equiv \alpha^{\uparrow\downarrow}$.*

We say that a path expression α is in $\cup$-NF (**union normal form**) if $\alpha = \beta_1 \cup \beta_2 \cup \cdots \cup \beta_n$ and the β_i are in up-down normal form (and thus $\cup$-free).

Proposition 66. *For all path expressions α in XPath$_=(\uparrow\downarrow)$, there is α' in $\cup$-NF such that $\alpha \equiv \alpha'$.*

Proof. We proceed by structural induction over α. If $\alpha = \varepsilon$ or $\alpha = \downarrow$ or $\alpha = \uparrow$, the result holds trivially. If $\alpha = [\varphi]$, with φ a node expression, we can take, by Proposition 64, a node expression ψ in up-down normal form (and therefore $\cup$-free) with $\psi \equiv \varphi$. Finally, for the concatenation $\alpha = \beta\gamma$, we can assume by induction that $\beta \equiv \beta_1 \cup \cdots \cup \beta_m$, and $\gamma \equiv \gamma_1 \cup \cdots \cup \gamma_n$, with β_i, γ_i being in up-down normal form. The conclusion follows from the fact that

$$(\beta_1 \cup \cdots \cup \beta_m)(\gamma_1 \cup \cdots \cup \gamma_n) \equiv \beta_1\gamma_1 \cup \beta_1\gamma_2 \cup \cdots \cup \beta_1\gamma_n \cup \beta_2\gamma_1 \cup \cdots \cup \beta_m\gamma_n$$

and the application of Proposition 65 on the $\cup$-free path expressions $\beta_i\gamma_j$. □

Remark 67. *From now on, we only consider the fragment of XPath$_=(\uparrow\downarrow)$ where all path expressions are in $\cup$-NF and all node expressions are in up-down normal form. Observe that, by Proposition 64 and Proposition 66, this fragment is semantically equivalent to full XPath$_=(\uparrow\downarrow)$.*

The following Lemma gives a restricted form of XPath$_=(\uparrow\downarrow)$-path expression negation:

Lemma 68. *Let $y \xrightarrow{n} x$, $y \xrightarrow{m} z$ in $\mathcal{T}$ and $y' \xrightarrow{n} x'$, $y' \xrightarrow{m} z'$ in $\mathcal{T}'$. If α is an XPath$_=(\uparrow\downarrow)$-path expression (in $\cup$-NF) such that $\mathcal{T}, x, z \models \alpha$ and $\mathcal{T}', x', z' \not\models \alpha$ then there is a path expression $\overline{\alpha}$ in up-down form such that $\mathcal{T}, x, z \not\models \overline{\alpha}$ and $\mathcal{T}', x', z' \models \overline{\alpha}$.*

Proof. Let $\alpha = \beta_1 \cup \beta_2 \cup \cdots \cup \beta_n$, with $\beta_i = [\varphi_i]\uparrow^{n_i}\downarrow^{m_i}[\psi_i]$. Let β_j be such that $\mathcal{T}, x, z \models \beta_j$. Since for all i we have $\mathcal{T}', x', z' \not\models \beta_i$, we have that either $\mathcal{T}', x' \not\models \langle\uparrow^{n_j}\downarrow^{m_j}\rangle$ (recall that both $y \xrightarrow{n} x$, $y \xrightarrow{m} z$, and $y' \xrightarrow{n} x'$, $y' \xrightarrow{m} z'$), or $\mathcal{T}', x' \not\models \varphi_j$, or $\mathcal{T}', z' \not\models \psi_j$. So either $\mathcal{T}', x' \models \neg\langle\uparrow^{n_j}\downarrow^{m_j}\rangle$ or $\mathcal{T}', x' \models \neg\varphi_j$ or $\mathcal{T}', z' \models \neg\psi_j$. In the first case, let $\overline{\alpha} = [\neg\langle\uparrow^{n_j}\downarrow^{m_j}\rangle]\uparrow^n\downarrow^m$, in the second case, let $\overline{\alpha} = [\neg\varphi_j]\uparrow^n\downarrow^m$, and in the third case, let $\overline{\alpha} = \uparrow^n\downarrow^m[\neg\psi_j]$. □

Binary bisimulation for XPath$_=(\uparrow\downarrow)$

Let $\mathcal{T}$ and $\mathcal{T}'$ be data trees. We say that $Z \subseteq T^2 \times T'^2$ is a **binary XPath$_=(\uparrow\downarrow)$-bisimulation**, or simply an **XPath$_=(\uparrow\downarrow)$-bisimulation** when the binary context is clear, if for all $x, y \in T$ and $x', y' \in T'$ we have:

- (**Harmony**) If $(x, y)Z(x', y')$ then $label(x) = label(x')$.
- (**Reverse**) $(x, y)Z(x', y')$ iff $(y, x)Z(y', x')$.
- (**Split-Zig**) If $(x, y)Z(x', y')$, then for all z such that $z \xrightarrow{m} x, z \xrightarrow{n} y$, there is z' such that $z' \xrightarrow{m} x', z' \xrightarrow{n} y'$, $(x, z)Z(x', z')$, and $(z, y)Z(z', y')$.
- (**Split-Zag**) If $(x, y)Z(x', y')$, then for all z' such that $z' \xrightarrow{m} x', z' \xrightarrow{n} y'$, there is z such that $z \xrightarrow{m} x, z \xrightarrow{n} y$, $(x, z)Z(x', z')$, and $(z, y)Z(z', y')$.

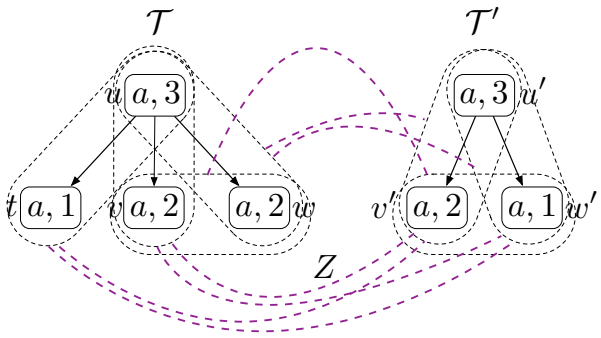


Figure 21: Part of an XPath$_=$(↑↓) binary bisimulation Z between $\mathcal{T}$ and $\mathcal{T}'$. Not shown: bisimulation of the all the reverse and singleton paths, and of the other two pairs of leaves in $\mathcal{T}$.

- (**Zig**) If $(x,y)Z(x',y')$, then for all z,w such that $z \xrightarrow{m} x, z \xrightarrow{n} w$, there are z',w' such that $z' \xrightarrow{m} x', z' \xrightarrow{n} w'$, $(z,w)Z(z',w')$, and $data(x) = data(w)$ iff $data(x') = data(w')$.
- (**Zag**) If $(x,y)Z(x',y')$, then for all z',w' such that $z' \xrightarrow{m} x', z' \xrightarrow{n} w'$, there are z,w such that $z \xrightarrow{m} x, z \xrightarrow{n} w$, $(z,w)Z(z',w')$, and $data(x) = data(w)$ iff $data(x') = data(w')$.

Observe that any of **Split-Zig** or **Split-Zag** imply that $(x,y)Z(x',y') \Rightarrow (x,x)Z(x',x')$, and this property in conjunction with **Reverse** implies that $(x,y)Z(x',y') \Rightarrow (y,y)Z(y',y')$. We call these two implications **Endpoints**. See Figure 21 for an (incomplete) example of a XPath$_=$(↑↓)-bisimulation.

We now define when $(u,v) \in T^2$ is said to be **XPath$_=$(↑↓)-bisimilar** to $(u',v') \in T'^2$, notated $\mathcal{T},u,v \leftrightarrows^{\uparrow\downarrow} \mathcal{T}',u',v'$:

$$\mathcal{T},u,v \leftrightarrows^{\uparrow\downarrow} \mathcal{T}',u',v' \overset{\text{def}}{\Leftrightarrow} \text{ there is binary XPath}_=(\uparrow\downarrow)\text{-bisimulation } Z \text{ s.t. } (u,v)Z(u',v').$$

We say that $(x,y) \in T^2$ and $(x',y') \in T'^2$ are **equivalent for XPath$_=$(↑↓)-path expressions** (notated $\mathcal{T},x,y \equiv^{\uparrow\downarrow} \mathcal{T}',x',y'$) iff the truth value of any XPath$_=$(↑↓)-path expression coincides over both two-pointed data trees. That is:

$$\mathcal{T},x,y \equiv^{\uparrow\downarrow} \mathcal{T}',x',y' \quad \overset{\text{def}}{\Leftrightarrow} \quad \begin{array}{l}\text{for all XPath}_=(\uparrow\downarrow)\text{-path expressions } \alpha,\\ \mathcal{T},x,y \models \alpha \text{ iff } \mathcal{T}',x',y' \models \alpha.\end{array}$$

Again, in the context of path expressions of XPath$_=$(↑↓) we have an analog of Theorem 9 for binary bisimulations and path equivalence.

Theorem 69. *$\mathcal{T},u,v \leftrightarrows^{\uparrow\downarrow} \mathcal{T}',u',v'$ implies $\mathcal{T},u,v \equiv^{\uparrow\downarrow} \mathcal{T}',u',v'$. The converse also holds when $\mathcal{T}$ and $\mathcal{T}'$ are finitely branching.*

Proof. We first show that if $\mathcal{T},u,v \leftrightarrows^{\uparrow\downarrow} \mathcal{T}',u',v'$ then $\mathcal{T},u,v \equiv^{\uparrow\downarrow} \mathcal{T}',u',v'$. We actually show that if $\mathcal{T},u,v \leftrightarrows^{\uparrow\downarrow} \mathcal{T}',u',v'$ via Z, then:

1. If $(x,x)Z(x',x')$, then $\mathcal{T},x,x \models [\varphi]$ iff $\mathcal{T}',x',x' \models [\varphi]$.

2. If $(x,y)Z(x',y')$, then $\mathcal{T},x,y \models \alpha$ iff $\mathcal{T}',x',y' \models \alpha$.

We show 1 and 2 by structural induction on $|\varphi|+|\alpha|$. We start with the Item 1. The base case for item 1 is $\varphi = a$, for some label a. Suppose $\mathcal{T},x,x \models [a]$. By **Harmony**, since $(x,x)Z(x',x')$, $label(x) = label(x')$, so $\mathcal{T}',x',x' \models [a]$. The case for $\mathcal{T}',x',x' \models [a]$ is identical. The Boolean cases for φ are straightforward.

Now suppose $\varphi = \langle \varepsilon = \alpha^{\uparrow\downarrow} \rangle$, and further assume that $\alpha^{\uparrow\downarrow} = [\psi_1]\uparrow^m\downarrow^n[\psi_2]$ (the cases with data inequality are analogous). Observe that by inductive hypothesis, it is enough to check $\mathcal{T},x,x \models [\langle \varepsilon = \uparrow^m\downarrow^n[\psi_2]\rangle]$ iff $\mathcal{T}',x',x' \models [\langle \varepsilon = \uparrow^m\downarrow^n[\psi_2]\rangle]$. We show the left-to-right implication, as the reverse is analogous. So, suppose $\mathcal{T},x,x \models \langle \varepsilon = \uparrow^m\downarrow^n[\psi_2]\rangle$. There exist z,w such that $z \xrightarrow{m} x$, $z \xrightarrow{n} w$, $\mathcal{T},x,w \models \uparrow^m\downarrow^n[\psi_2]$, and $data(x) = data(w)$. By **Zig**, there are z',w' such that $z' \xrightarrow{m} x'$, $z' \xrightarrow{n} w'$, $(x,z)Z(x',z')$, $(z,w)Z(z',w')$, and $data(x') = data(w')$. By inductive hypothesis, since $(z,w)Z(z',w')$ and $\mathcal{T},z,w \models \downarrow^n[\psi_2]$ we have $\mathcal{T}',z',w' \models \downarrow^n[\psi_2]$. Since also $\mathcal{T}',x',z' \models \uparrow^m$, we conclude $\mathcal{T}',x',w' \models \uparrow^m\downarrow^n[\psi_2]$, and therefore (because $data(x') = data(w')$), $\mathcal{T}',x',x' \models \langle \varepsilon = \uparrow^m\downarrow^n[\psi_2]\rangle$, as we wanted.

We now proceed to Item 2. We only show the left-to-right direction, as the reverse is analogous. The base case is when $\alpha \in \{\varepsilon, \uparrow, \downarrow\}$. If $\alpha = \varepsilon$ then $\mathcal{T},x,y \models \alpha$ iff $x = y$. Thus, taking $z = x$ (and thus $m = n = 0$) in **Split-Zig**, it follows that $x' = y'$ and therefore $\mathcal{T}',x',y' \models \alpha$. If $\alpha = \uparrow$ then $\mathcal{T},x,y \models \alpha$ implies $y \rightarrow x$. Now, $\mathcal{T},y,x \models \downarrow$, and, from **Split-Zig** we deduce, $\mathcal{T},y',x' \models \downarrow$. Therefore we conclude $\mathcal{T},x',y' \models \alpha$, as we wanted. If $\alpha = \downarrow$, we proceed as before.

Finally, for the general case where $\alpha = \alpha^{\uparrow\downarrow}$. Suppose without loss of generality that $\mathcal{T},x,y \models [\varphi]\uparrow^m\downarrow^n[\psi]$. Then, there exists z such that $\mathcal{T},x,z \models [\varphi]\uparrow^m$ and $\mathcal{T},z,y \models \downarrow^n[\psi]$. Since $z \xrightarrow{m} x$ and $z \xrightarrow{n} y$, by **Split-Zig**, we have a corresponding z' such that $z' \xrightarrow{m} x'$ and $z' \xrightarrow{n} y'$, $(x,z)Z(x',z')$, and $(z,y)Z(z',y')$. If $m = n = 0$, then $x = y$, and the problem consists of the already considered case $\mathcal{T},x,x \models [\varphi]$. If $m \neq 0$ or $n \neq 0$, then $|[\varphi]\uparrow^m| < |\alpha|$ and $|\downarrow^n[\psi]| < |\alpha|$, and thus, since $(x,z)Z(x',z')$ and $(z,y)Z(z',y')$, we can use the inductive hypothesis to conclude that $\mathcal{T}',x',z' \models [\varphi]\uparrow^m$ and $\mathcal{T}',z',y' \models \downarrow^n[\psi]$, and therefore $\mathcal{T}',x',y' \models [\varphi]\uparrow^m\downarrow^n[\psi]$, as we wanted.

We now show that if $\mathcal{T}$ and $\mathcal{T}'$ are finitely branching, then $\mathcal{T},u,v \equiv^{\uparrow\downarrow} \mathcal{T}',u',v'$ implies $\mathcal{T},u,v \leftrightarrows^{\uparrow\downarrow} \mathcal{T}',u',v'$. Let $\mathcal{T},u,v \equiv^{\uparrow\downarrow} \mathcal{T}',u',v'$. Define the relation Z by:

$$(x,y)Z(x',y') \text{ iff } \mathcal{T},x,y \equiv^{\uparrow\downarrow} \mathcal{T}',x',y'.$$

We show that Z is a bisimulation between $\mathcal{T},u,v$ and $\mathcal{T}',u',v'$.

First of all, by construction, it holds that $(u,v)Z(u',v')$.

To prove **Harmony**, let $(x,y)Z(x',y')$. We will see that if $label(x) = a$ then $label(x') = a$ (the other implication is analogous). Note that, since $\mathcal{T}$ is a tree, there are m,n such that $\mathcal{T},x,y \models \uparrow^m\downarrow^n$. Also, if $label(x) = a$, $\mathcal{T},x,y \models [a]\uparrow^m\downarrow^n$. Therefore, since $\mathcal{T},x,y \equiv^{\uparrow\downarrow} \mathcal{T}',x',y'$, we have $\mathcal{T}',x',y' \models [a]\uparrow^m\downarrow^n$, and thus $label(x') = a = label(x)$.

Now we check **Reverse**. Let $(x,y)Z(x',y')$. Observe first that it is enough to check that $(x,y)Z(x',y') \Rightarrow (y,x)Z(y',x')$. Now, let β be a path expression such that $\mathcal{T},y,x \models \beta$,

which using Remark 67 we can assume to be in up-down normal form[13] $\beta = [\psi]\uparrow^n\downarrow^m[\varphi]$. Then, $\mathcal{T}, x, y \models [\varphi]\uparrow^m\downarrow^n[\psi]$, and, since $(x,y)Z(x',y')$, this implies that $\mathcal{T}', x', y' \models [\varphi]\uparrow^m\downarrow^n[\psi]$. In turn, this implies that $\mathcal{T}', y', x' \models [\psi]\uparrow^n\downarrow^m[\varphi] = \beta$, as we wanted.

Now we check **Split-Zig** (**Split-Zag** is analogous). Let $(x,y)Z(x',y')$. We prove that if $z \xrightarrow{m} x$ and $z \xrightarrow{n} y$ then there is z' in T' such that $z' \xrightarrow{m} x'$, $z' \xrightarrow{n} y'$, $(x,z)Z(x',z')$, and $(z,y)Z(z',y')$. We have $\mathcal{T}, x, y \models \uparrow^m\downarrow^n$, and then so does $\mathcal{T}', x', y' \models \uparrow^m\downarrow^n$. In particular, there exists z' such that $z' \xrightarrow{m} x'$, $z' \xrightarrow{n} y'$. To verify $(x,z)Z(x',z')$, we see that if α is a path expression such that $\mathcal{T}, x, z \models \alpha$, then $\mathcal{T}', x', z' \models \alpha$ (the other implication is analogous). Observe that $\mathcal{T}, x, y \models \alpha\downarrow^n$, which implies that $\mathcal{T}', x', y' \models \alpha\downarrow^n$. As there is only one w such that $w \xrightarrow{n} y'$, namely z', we conclude that $\mathcal{T}', x', z' \models \alpha$, as we wanted. To verify $(z,y)Z(z',y')$, we see that if α is a path expression such that $\mathcal{T}, z, y \models \alpha$, then, $\mathcal{T}', z', y' \models \alpha$ (the other implication is analogous). Now, $\mathcal{T}, z, y \models \alpha$ implies $\mathcal{T}, x, y \models \uparrow^m\alpha$, and then $\mathcal{T}', x', y' \models \uparrow^m\alpha$. Since $\mathcal{T}'$ is a tree, this in turn implies that $\mathcal{T}', z', y' \models \alpha$, as we wanted.

For the last step, we check that **Zig** holds (**Zag** is analogous). Suppose $\mathcal{T}, x, y \equiv^{\uparrow\downarrow} \mathcal{T}', x', y'$ (that is, $(x,y)Z(x',y')$). Let z, w be such that $z \xrightarrow{m} x$, $z \xrightarrow{n} w$, and assume that $data(x) = data(w)$ (the case for $\neq$ is analogous). We want to see that there are z', w' in T' such that $z' \xrightarrow{m} x'$, $z' \xrightarrow{n} w'$, $data(x') = data(w')$, and $(z,w)Z(z',w')$. By **Split-Zig**, let $z' \in T'$ such that $z' \xrightarrow{m} x'$ and $(x,z)Z(x',z')$. Let

$$P = \{w' \in T' \mid z' \xrightarrow{n} w' \text{ and } data(x') = data(w')\}.$$

Notice that P is finite since $\mathcal{T}'$ is **finitely branching**. We show that there is $w' \in P$ such that $(z,w)Z(z',w')$. By **Split-Zig** we had $\mathcal{T}, x, z \equiv^{\uparrow\downarrow} \mathcal{T}', x', z'$, and thus $\mathcal{T}, x, z \models [\langle\varepsilon = \uparrow^m\downarrow^n\rangle]\uparrow^m$ implies $\mathcal{T}', x', z' \models [\langle\varepsilon = \uparrow^m\downarrow^n\rangle]\uparrow^m$, and so there is w' such that $z' \xrightarrow{n} w'$ and $data(x') = data(w')$. Hence $P \neq \emptyset$.

Now, suppose by the way of contradiction that for all $w' \in P$, we have $\mathcal{T}, z, w \not\equiv^{\uparrow\downarrow} \mathcal{T}', z', w'$. That is, for every $w' \in P$, there exists a path expression, which we can assume is in up-down form $\alpha_{w'} = [\varphi_{w'}]\uparrow^{a_{w'}}\downarrow^{b_{w'}}[\psi_{w'}]$, such that either

1. $\mathcal{T}, z, w \models \alpha_{w'}$ and $\mathcal{T}', z', w' \not\models \alpha_{w'}$, or

2. $\mathcal{T}, z, w \not\models \alpha_{w'}$ and $\mathcal{T}', z', w' \models \alpha_{w'}$.

First we are going to see that we can assume that $\alpha_{w'}$ is of the form $[\varphi_{w'}]\downarrow^n[\psi_{w'}]$. First of all, observe that since $\mathcal{T}, x, z \equiv^{\uparrow\downarrow} \mathcal{T}', x', z'$, by **Endpoints** we have that $(z,z)Z(z',z')$. Now suppose by the way of contradiction that $\uparrow^{a_{w'}}\downarrow^{b_{w'}}$ holds in $\mathcal{T}, z, w$ but not in $\mathcal{T}', z', w'$ (the other case is analogous). Since $z \xrightarrow{n} w$, it must be that $b_{w'} - a_{w'} = n$. Since also $z' \xrightarrow{n} w'$ but $\mathcal{T}', z', w' \not\models \uparrow^{a_{w'}}\downarrow^{b_{w'}}$, we have $\mathcal{T}', z', z' \not\models [\langle\uparrow^{a_{w'}}\downarrow^{b_{w'}}\rangle]$, or, equivalently, $\mathcal{T}', z', z' \models [\neg\langle\uparrow^{a_{w'}}\downarrow^{b_{w'}}\rangle]$. But then $\mathcal{T}, z, z \models [\neg\langle\uparrow^{a_{w'}}\downarrow^{b_{w'}}\rangle]$, and this implies that $\mathcal{T}, z, w \not\models \uparrow^{a_{w'}}\downarrow^{b_{w'}}$, a contradiction. So we can assume without loss of generality that always $\uparrow^{a_{w'}}\downarrow^{b_{w'}} = \downarrow^n$.

[13] In the case $\beta = \beta_1 \cup \beta_2 \cup \cdots \cup \beta_n$ with the β_i in up-down normal form, it is enough to take a β_j such that $\mathcal{T}, y, x \models \beta_j$ and see that $\mathcal{T}', y', x' \models \beta_j$.

Now, by Lemma 68, we can always assume that case 1 applies. We take:

$$\alpha = [\bigwedge_{w' \in P} \varphi_{w'}]\downarrow^n[\bigwedge_{w' \in P} \psi_{w'}]\uparrow^n$$

and observe that $\mathcal{T}, z, z \models \alpha$ but $\mathcal{T}', z', z' \not\models \alpha$, a contradiction. □

Corollary 70. *$\mathcal{T}, x \equiv^{\uparrow\downarrow} \mathcal{T}', x'$ iff $\mathcal{T}, x, x \equiv^{\uparrow\downarrow} \mathcal{T}', x', x'$. Thus, if $\mathcal{T}$ and $\mathcal{T}'$ are finitely branching, then $\mathcal{T}, x \leftrightarrows^{\uparrow\downarrow} \mathcal{T}', x'$ iff $\mathcal{T}, x, x \leftrightarrows^{\uparrow\downarrow} \mathcal{T}', x', x'$.*

Proof. The proof is similar to that of Corollary 59. For the second part we use that if $\mathcal{T}$ and $\mathcal{T}'$ are finitely branching, then $\leftrightarrows^{\uparrow\downarrow}$ and $\equiv^{\uparrow\downarrow}$ coincide (Theorem 69)

Assume first that $x \equiv^{\uparrow\downarrow} x'$. Suppose that $\mathcal{T}, x, x \models \alpha$ and let us prove that $\mathcal{T}', x', x' \models \alpha$ (the other implication is analogous). Without loss of generality we can assume that $\alpha = [\varphi]\uparrow^m\downarrow^n[\psi]$. So $n = m$, $\mathcal{T}, x \models \varphi$, and $\mathcal{T}, x \models \psi$. Since $\mathcal{T}, x \equiv^{\uparrow\downarrow} \mathcal{T}', x'$, we conclude that $\mathcal{T}', x', x' \models [\varphi]\uparrow^m\downarrow^n[\psi]$.

For the other implication, assume $x, x \equiv^{\uparrow\downarrow} x', x'$. In particular, $\mathcal{T}, x, x \models [\varphi]$ iff $\mathcal{T}', x', x' \models [\varphi]$. As $\mathcal{T}, x, x \models [\varphi]$ iff $\mathcal{T}, x \models \varphi$ and $\mathcal{T}', x', x' \models [\varphi]$ iff $\mathcal{T}', x' \models \varphi$, we get $\mathcal{T}, x \models \varphi$ iff $\mathcal{T}', x' \models \varphi$, as we wanted. □

1.5 Definability via path expressions

In this section we develop and apply the tools needed to obtain the main results of definability and separation via path expressions of $\mathrm{XPath}_{=}(\downarrow)$ and $\mathrm{XPath}_{=}(\uparrow\downarrow)$. Here we work over more restricted universes than in the case of node expressions, in order to deal with the lack of complementation or intersection of path expression over the full universe of two-pointed data trees.

1.5.1 Saturation

Saturation for the downward fragment. Let Σ and Γ be sets of $\mathrm{XPath}_{=}(\downarrow)$-path expressions. Given a data tree $\mathcal{T}$ and $u \in T$, we say that Σ and Γ are **$=^{\downarrow}$-satisfiable** [resp. **$\neq^{\downarrow}$-satisfiable**] at $\mathcal{T}, u$ if there exist $v, w \in T$ such that $\mathcal{T}, u, v \models \Sigma$, $\mathcal{T}, u, w \models \Gamma$, and $data(v) = data(w)$ [resp. $data(v) \neq data(w)$]. We say that Σ and Γ are **$=^{\downarrow}$-finitely satisfiable** [resp. **$\neq^{\downarrow}$-finitely satisfiable**] at $\mathcal{T}, u$ if for every finite $\Sigma' \subseteq \Sigma$ and finite $\Gamma' \subseteq \Gamma$, we have that Σ' and Γ' are $=^{\downarrow}$-satisfiable [resp. $\neq^{\downarrow}$-satisfiable] at $\mathcal{T}, u$.

Definition 71. We say that a data tree $\mathcal{T}$ is **binary $\downarrow$-saturated** if for every pair of sets Σ, Γ of $\mathrm{XPath}_{=}(\downarrow)$-path expressions, every $u \in T$, and $\star \in \{=, \neq\}$, the following is true:

> if Σ and Γ are $\star^{\downarrow}$-finitely satisfiable at $\mathcal{T}, u$ then Σ and Γ are $\star^{\downarrow}$-satisfiable at $\mathcal{T}, u$.

Proposition 72. *Any finitely branching data tree is binary $\downarrow$-saturated.*

Proof. Suppose by contradiction that there is $u \in T$ and sets of XPath$_=(\downarrow)$-path expressions Σ, Γ which are finitely $=^{\downarrow}$-satisfiable at $\mathcal{T}, u$ but not $=^{\downarrow}$-satisfiable at $\mathcal{T}, u$ (the case for $\mathcal{T}$ being $\neq^{\downarrow}$-satisfiable is analogous). Assume, without loss of generality by Lemma 46 that all path expressions in Σ and Γ are the finite union of path expressions in simple normal form. Let $\alpha_s = \bigcup_{i \in I} \alpha_{s,i}$ be one formula in Σ. By Fact 43, there are m_i such that if $\mathcal{T}, u, v \models \alpha_{s,i}$, then $u \xrightarrow{n_i} v$. Similarly, take an $\alpha_g = \bigcup_{j \in J} \alpha_{g,j}$ for Γ. We are going to construct finite subsets of Σ and Γ, containing α_s and α_g respectively, such that they are not $=^{\downarrow}$-satisfiable at $\mathcal{T}, u$. Let

$$P = \{(v, w) \in T^2 \mid u \xrightarrow{m_i} v \wedge u \xrightarrow{n_j} w \wedge \mathit{data}(v) = \mathit{data}(w) \wedge i \in I \wedge j \in J\}.$$

Observe that P is finite because I and J are finite and $\mathcal{T}$ is finitely branching. As we are assuming Σ, Γ not $=^{\downarrow}$-satisfiable at $\mathcal{T}, u$, it is clear that if $(v, w) \in P$ then either

1. $\mathcal{T}, u, v \not\models \Sigma$, or
2. $\mathcal{T}, u, w \not\models \Gamma$.

We will define sets $\Sigma_{v,w}$ and $\Gamma_{v,w}$, each one of them with at most one element, as follows: If case 14 holds, define $\Sigma_{v,w}$ as $\{\rho\}$ for some path expression $\rho \in \Sigma$ such that $\mathcal{T}, u, v \not\models \rho$, and define $\Gamma_{v,w} = \emptyset$. If case 14 does not hold then case 15 holds, so define $\Gamma_{v,w}$ as $\{\rho\}$ for some path expression $\rho \in \Gamma$ such that $\mathcal{T}, u, w \not\models \rho$, and define $\Sigma_{v,w} = \emptyset$. Finally, define the finite sets $\Sigma' = \bigcup_{(v,w) \in P} \Sigma_{v,w} \cup \alpha_s$ and $\Gamma' = \bigcup_{(v,w) \in P} \Gamma_{v,w} \cup \alpha_g$. Observe that if $\mathit{data}(v) = \mathit{data}(w)$, $\mathcal{T}, u, v \models \alpha_s$, and $\mathcal{T}, u, w \models \alpha_g$ then $(v, w) \in P$. Thus, by construction of Σ' and Γ', we have finite $\Sigma' \subseteq \Sigma$, $\Gamma' \subseteq \Gamma$ such that Σ' and Γ' are not $=^{\downarrow}$-satisfiable at $\mathcal{T}, u$ which is a contradiction. □

Proposition 73. *Let $\mathcal{T}$ and $\mathcal{T}'$ be $\downarrow$-saturated data trees, and let $u, v \in T$ and $u', v' \in T'$. If $\mathcal{T}, u, v \equiv^{\downarrow} \mathcal{T}', u', v'$, then $\mathcal{T}, u, v \leftrightarrows^{\downarrow} \mathcal{T}', u', v'$.*

Proof. We show that Z, defined by $(x, y)Z(x', y')$ iff $\mathcal{T}, x, y \equiv^{\downarrow} \mathcal{T}', x', y'$ is a XPath$_=(\downarrow)$-bisimulation between $\mathcal{T}, u, v$ and $\mathcal{T}', u', v'$. Clearly $(u, v)Z(u', v')$, and **Harmony** also holds. For **Equidistance**, if $(x, y)Z(x', y')$, assume $x \xrightarrow{k} y$. Then, since $(x, y) \equiv^{\downarrow} (x', y')$, $\mathcal{T}, x, y \models{\downarrow}^k$ iff $\mathcal{T}', x', y' \models{\downarrow}^k$. For **Split**, let $(x, y)Z(x', y')$, $x \xrightarrow{n} z \xrightarrow{m} y$, and $x' \xrightarrow{n} z' \xrightarrow{m} y'$. We check first that $(x, z)Z(x', z')$: $\mathcal{T}, x, z \models \alpha \Leftrightarrow \mathcal{T}, x, y \models \alpha\downarrow^m \Leftrightarrow \mathcal{T}', x', y' \models \alpha\downarrow^m \Leftrightarrow \mathcal{T}', x', z' \models \alpha$. The proof is similar for checking $(z, y)Z(z', y')$: $\mathcal{T}, z, y \models \alpha \Leftrightarrow \mathcal{T}, x, y \models{\downarrow}^n \alpha \Leftrightarrow \mathcal{T}', x', y' \models{\downarrow}^n \alpha \Leftrightarrow \mathcal{T}', z', y' \models \alpha$.

We now need to show that **Zig** and **Zag** are satisfied. We only check **Zig**, as **Zag** is analogous. Suppose $(x, y)Z(x', y')$, $x \xrightarrow{m} v$, $x \xrightarrow{n} w$ and $\mathit{data}(v) = \mathit{data}(w)$ (the case with $\mathit{data}(v) \neq \mathit{data}(w)$ is analogous). Let

$$\Sigma = \{\alpha \mid \mathcal{T}, x, v \models \alpha \text{ and } \alpha \text{ is } \cup\text{-free}\} \quad \text{and} \quad \Gamma = \{\alpha \mid \mathcal{T}, x, w \models \alpha \text{ and } \alpha \text{ is } \cup\text{-free}\}.$$

That is, Σ and Γ are the $\cup$-free theories of $\mathcal{T}, x, v$ and $\mathcal{T}, x, w$, respectively. Furthermore, let Σ' be a finite subset of Σ, and let Γ' be a finite subset of Γ. Observe that, being in

$\mathrm{XPath}_=(\downarrow)$, all path expressions in Γ and Σ are of the same length, and thus we have a notion of intersection as in Equation 17.

Now define $\varphi = \langle \cap\Sigma' = \cap\Gamma' \rangle$. Observe that, from **Split**, $(x, y) \equiv^{\downarrow} (x', y')$ implies $(x, x) \equiv^{\downarrow} (x', x')$ implies (Corollary 59) $x \equiv^{\downarrow} x'$. Now, it is clear that $\mathcal{T}, x \models \varphi$, and thus $\mathcal{T}', x' \models \varphi$. Therefore, there exist v', w' such that $\mathcal{T}', x', v' \models \Sigma'$ (in particular, $x' \xrightarrow{m} v'$), $\mathcal{T}', x', w' \models \Gamma'$ (in particular, $x' \xrightarrow{n} w'$), and $data(v') = data(w')$. Hence Σ' and Γ' are $=^{\downarrow}$-satisfiable at x', for *any* finite sets Σ', Γ' and thus Σ and Γ are $=^{\downarrow}$-finitely satisfiable at x'. Since $\mathcal{T}'$ is $\downarrow$-saturated, this implies that Σ and Γ are $=^{\downarrow}$-satisfiable at x', for some v' and w'.

Finally, we see that $\mathcal{T}', x', v' \models \Sigma$ implies that $\mathrm{Th}_{\downarrow}(\mathcal{T}, x, v) = \mathrm{Th}_{\downarrow}(\mathcal{T}', x', v')$ and thus $(x, v) \equiv^{\downarrow} (x', v')$ (the case for $(x, w) \equiv^{\downarrow} (x', b')$ is analogous). We are only going to prove that $\mathcal{T}', x', v' \models \alpha \Rightarrow \mathcal{T}, x, v \models \alpha$, as the other implication is clear. Suppose by way of contradiction that there is an α, which by Lemma 44 can be assumed to be $\cup$-free, such that $\mathcal{T}', x', v' \models \alpha$ but $\mathcal{T}, x, v \not\models \alpha$. Then, by Lemma 45, there is a $\cup$-free path expression $\bar{\alpha}$ such that $\mathcal{T}', x', v' \not\models \bar{\alpha}$ and $\mathcal{T}, x, v \models \bar{\alpha}$. Then, since $\mathcal{T}', x', v' \models \Sigma$, we have that $\mathcal{T}', x', v' \models \bar{\alpha}$, a contradiction. □

Saturation for the vertical fragment. Given a data tree $\mathcal{T}$ and $u \in T$, we say that the set of $\mathrm{XPath}_=(\uparrow\downarrow)$-path expressions Γ is **$=^{\uparrow\downarrow}$-satisfiable** [resp. **$\neq^{\uparrow\downarrow}$-satisfiable**] at $\mathcal{T}, u$ if there exist $v, w \in T$ such that $v \xrightarrow{*} u$, $v \xrightarrow{*} w$, $\mathcal{T}, u, w \models \Gamma$ and $data(u) = data(w)$ [resp. $data(u) \neq data(w)$]. We say that Γ is **$=^{\uparrow\downarrow}_{n,m}$-satisfiable** [resp. **$\neq^{\uparrow\downarrow}_{n,m}$-satisfiable**] at $\mathcal{T}, u$ if for every finite Γ', we have that Γ' is $=^{\uparrow\downarrow}$-satisfiable [resp. $=^{\uparrow\downarrow}$-satisfiable] at $\mathcal{T}, u$.

Definition 74. We say that a data tree $\mathcal{T}$ is **binary $\uparrow\downarrow$-saturated** if for every set of $\mathrm{XPath}_=(\uparrow\downarrow)$-path expressions Γ, every $u \in T$ and $\star \in \{=, \neq\}$, the following is true:

> if Γ is $\star^{\uparrow\downarrow}$-finitely satisfiable at $\mathcal{T}, u$ then Γ is $\star^{\uparrow\downarrow}$-satisfiable at $\mathcal{T}, u$.

Proposition 75. *Let $\mathcal{T}$ and $\mathcal{T}'$ be binary $\uparrow\downarrow$-saturated data trees, and let $u, v \in T$ and $u', v' \in T'$. If $\mathcal{T}, u, v \equiv^{\uparrow\downarrow} \mathcal{T}', u', v'$, then $\mathcal{T}, u, v \leftrightarrows^{\uparrow\downarrow} \mathcal{T}', u', v'$.*

Proof. We show that $Z \subseteq T^2 \times T'^2$, defined by

$$(x, y) Z (x', y') \text{ iff } \mathcal{T}, x, y \equiv^{\uparrow\downarrow} \mathcal{T}', x', y'$$

is a $\mathrm{XPath}_=(\uparrow\downarrow)$-bisimulation between $\mathcal{T}, u, v$ and $\mathcal{T}', u', v'$. Clearly $(u, v) Z (u', v')$. **Harmony**, **Reverse**, **Split-Zig**, and **Split-Zag** hold with the same proofs as in the second part of the proof of Theorem 69.

We now need to show that **Zig** and **Zag** are satisfied. We only check **Zig**, as **Zag** is analogous.

Suppose $(x, y) Z (x', y')$, $s \xrightarrow{c} x$, $s \xrightarrow{d} y$, $z \xrightarrow{m} x$, $z \xrightarrow{n} w$, and $data(x) = data(w)$ (the case $\neq$ is analogous). We want to see that there are $z', w' \in T'$ such that $z' \xrightarrow{m} x'$, $z' \xrightarrow{n} w'$, $(z, w) Z (z', w')$, and $data(x') = data(w')$. Let

$$\Gamma = \{\beta \mid \mathcal{T}, x, w \models \beta \text{ and } \beta \text{ is of the form } [\varphi] \uparrow^m \downarrow^n [\psi], \text{ for some } \varphi \text{ and } \psi\},$$

and let Γ' be a finite subset of Γ. If $\beta_1 = [\varphi_1] \uparrow^m\downarrow^n [\psi_1]$ and $\beta_2 = [\varphi_2] \uparrow^m\downarrow^n [\psi_2]$, we will define $\beta_1 \cap \beta_2 = [\varphi_1 \cap \varphi_2] \uparrow^m\downarrow^n [\psi_1 \cap \psi_2]$.

Now, define

$$\alpha = [\langle \varepsilon = \cap\Gamma' \rangle] \uparrow^c\downarrow^d .$$

It can be seen that $\mathcal{T}, x, y \models \alpha$, and thus, since by definition of Z we have $\mathcal{T}, x, y \equiv^{\uparrow\downarrow} \mathcal{T}', x', y'$, we conclude $\mathcal{T}', x', y' \models \alpha$. This implies that there are p', q' such that $p' \xrightarrow{m} x'$, $p' \xrightarrow{n} q'$, $data(x') = data(q')$, and $\mathcal{T}', x', q' \models \Gamma'$. Therefore, Γ is $=^{\uparrow\downarrow}$-finitely satisfiable at $\mathcal{T}', x'$. Since $\mathcal{T}'$ is binary $\uparrow\downarrow$-saturated, this implies that Γ is $=^{\uparrow\downarrow}$-satisfiable at $\mathcal{T}', x'$, and therefore there exist nodes $z', w' \in T'$ such that $t \xrightarrow{m} x'$, $t \xrightarrow{n} w'$, $data(x') = data(w')$, and $\mathcal{T}', x', w' \models \Gamma$.

It remains to prove that $\mathrm{Th}_{\uparrow\downarrow}(\mathcal{T}, x, w) = \mathrm{Th}_{\uparrow\downarrow}(\mathcal{T}', x', w')$, as this property in conjunction with **Split-Zig** will imply that $(z, w)Z(z', w')$.

First we prove that $\mathrm{Th}_{\uparrow\downarrow}(\mathcal{T}, x, w) \subseteq \mathrm{Th}_{\uparrow\downarrow}(\mathcal{T}', x', w')$. Let $\beta \in \mathrm{Th}_{\uparrow\downarrow}(\mathcal{T}, x, w)$. Without loss of generality, we can assume that β is $\cup$-free, and thus of the form $\beta = [\varphi] \uparrow^j\downarrow^k [\psi]$. Since $z' \xrightarrow{m} x'$ and $z' \xrightarrow{n} w'$, $\mathcal{T}', x', w' \models \beta$ iff $\mathcal{T}', x', w' \models \gamma$, with $\gamma = [\varphi \wedge \langle \uparrow^j\downarrow^k \rangle] \uparrow^m\downarrow^n [\psi]$. But $\gamma \in \Gamma$, and thus $\mathcal{T}', x', w' \models \gamma$.

We now see that $\mathrm{Th}_{\uparrow\downarrow}(\mathcal{T}', x', w') \subseteq \mathrm{Th}_{\uparrow\downarrow}(\mathcal{T}, x, w)$. Suppose by way of contradiction that there is a β (which can be assumed to be $\cup$-free) such that $\beta = [\varphi] \uparrow^j\downarrow^k [\psi]$ and $\mathcal{T}', x', w' \models \beta$ but $\mathcal{T}, x, w \not\models \beta$. As $z \xrightarrow{m} x$ and $z \xrightarrow{n} w$, $\mathcal{T}, x, w \models \beta$ iff $\mathcal{T}, x, w \models [\varphi \wedge \langle \uparrow^j\downarrow^k \rangle] \uparrow^m\downarrow^n [\psi]$, and as $z' \xrightarrow{m} x'$ and $z' \xrightarrow{n} w'$, we also have $\mathcal{T}', x', w' \models \beta$ iff $\mathcal{T}', x', w' \models [\varphi \wedge \langle \uparrow^j\downarrow^k \rangle] \uparrow^m\downarrow^n [\psi]$. So from our supposition we have $\mathcal{T}, x, w \not\models [\varphi \wedge \langle \uparrow^j\downarrow^k \rangle] \uparrow^m\downarrow^n [\psi]$. Since $\mathcal{T}, x, w \models \uparrow^m\downarrow^n$, it must be that either $\mathcal{T}, x, w \models [\neg(\varphi \wedge \langle \uparrow^j\downarrow^k \rangle)] \uparrow^m\downarrow^n$ or $\mathcal{T}, x, w \models \uparrow^m\downarrow^n [\neg\psi]$. But since $\mathcal{T}', x', w' \models \Gamma$, this implies that either $\mathcal{T}', x', w' \models [\neg(\varphi \wedge \langle \uparrow^j\downarrow^k \rangle)] \uparrow^m\downarrow^n$ or $\mathcal{T}', x', w' \models \uparrow^m\downarrow^n [\neg\psi]$, which contradicts the fact that $\mathcal{T}', x', w' \models [\varphi \wedge \langle \uparrow^j\downarrow^k \rangle] \uparrow^m\downarrow^n [\psi]$.

So we have $(x, w)Z(x', w')$. As $z \xrightarrow{m} x$ and $z \xrightarrow{n} w$, we can use **Split-Zig** to finally obtain $(z, w)Z(z', w')$, as we wanted.

□

1.5.2 Weak data trees and quasi-ultraproducts

Recall that a σ-structure $\mathcal{T}$ is a **weak data tree** if: $\sim$ is an equivalence relation; there is exactly one node r with no u such that $u \rightsquigarrow r$ (r is called *root* of $\mathcal{T}$); for all nodes $x \neq r$ there is exactly one y such that $y \rightsquigarrow x$; and for each $n \geq 0$ the relation $\rightsquigarrow$ has no cycles of length n. Recall also that a weak data tree need not be connected, and that the class of weak data trees is elementary, i.e. definable by a set of first-order σ-sentences (with equality). For a weak data tree $\mathcal{T}$ and $u \in T$, remember that $\mathcal{T}|_u$ denotes the substructure of $\mathcal{T}$ induced by $\{v \in T \mid u \rightsquigarrow^* v\}$, and that $\mathcal{T}|_u$ is a data tree.

The following proposition shows the 'local' aspect of $\mathrm{XPath}_=(\downarrow)$ and $\mathrm{XPath}_=(\uparrow\downarrow)$ for paths, whereas Proposition 17 showed it for nodes. It is stated in terms of first-order because models are weak data trees. Recall that Tr is the translation into first-order logic given in §I.1.3.

Proposition 76. *Let $\mathcal{T}$ be a weak data tree and let both $r \rightsquigarrow^* u$ and $r \rightsquigarrow^* v$ in $\mathcal{T}$.*

1. *If α is a XPath$_=(\downarrow)$-path expression then $\mathcal{T} \models \mathrm{Tr}_{x,y}(\alpha)[u,v]$ iff $\mathcal{T}|_r \models \mathrm{Tr}_{x,y}(\alpha)[u,v]$.*

2. *If r is the root of $\mathcal{T}$ and $\alpha \in$ XPath$_=(\uparrow\downarrow)$ then $\mathcal{T} \models \mathrm{Tr}_{x,y}(\alpha)[u,v]$ iff $\mathcal{T}|_r \models \mathrm{Tr}_{x,y}(\alpha)[u,v]$.*

We now show that 2-saturated (recall Definition 18) data trees are already both binary $\downarrow$-saturated and binary $\uparrow\downarrow$-saturated. For technical reasons we state these results in the more general setting of weak data trees.

Proposition 77. *Let $\mathcal{T}$ be a 2-saturated weak data tree and $r \in T$.*

1. *$\mathcal{T}|_r$ is a binary $\downarrow$-saturated data tree.*

2. *If r is the root of T then $\mathcal{T}|_r$ is a binary $\uparrow\downarrow$-saturated data tree.*

Proof. The proof is similar to that of Proposition 19.

Let $\mathcal{T}' = \mathcal{T}|_r$ and let $u \in T'$. For item 1, let Σ and Γ be sets of XPath$_=(\downarrow)$-path expressions. Suppose Σ and Γ are $=^{\downarrow}$-finitely satisfiable at $\mathcal{T}', u$ (the case for $\neq^{\downarrow}$-finitely satisfiable is analogous). We show that Σ and Γ are $=^{\downarrow}$-satisfiable at $\mathcal{T}', u$. Consider the following first-order $\sigma_{\{u\}}$-formula with free variables $\bar{x} = x_1, \ldots, x_n$ and $\bar{y} = y_1, \ldots, y_m$:

$$\varphi(\bar{x}, \bar{y}) = u \rightsquigarrow x_1 \wedge \bigwedge_{i=1}^{n-1} x_i \rightsquigarrow x_{i+1} \wedge u \rightsquigarrow y_1 \wedge \bigwedge_{j=1}^{m-1} y_j \rightsquigarrow y_{j+1} \wedge x_n \sim y_m.$$

Define the following set of first-order $\sigma_{\{u\}}$-formulas:

$$\Delta(\bar{x}, \bar{y}) = \{\varphi(\bar{x}, \bar{y})\} \cup \mathrm{Tr}_{x_n}(\Sigma) \cup \mathrm{Tr}_{y_m}(\Gamma).$$

Let $\Delta'(\bar{x}, \bar{y})$ be a finite subset of $\Delta(\bar{x}, \bar{y})$. Since Σ and Γ are $=^{\downarrow}$-finitely satisfiable at $\mathcal{T}', u$, then $\Delta'(\bar{x}, \bar{y})$ is satisfiable and, by item 1 of Proposition 76, consistent with $\mathrm{Th}_{\{u\}}(\mathcal{T})$. By compactness, $\Delta(\bar{x}, \bar{y})$ is satisfiable and consistent with $\mathrm{Th}_{\{u\}}(\mathcal{T})$. By 2-saturation, we conclude that $\Delta(\bar{x}, \bar{y})$ is realizable in $\mathcal{T}$, say at $\bar{v} = v_1, \ldots, v_n$ and $\bar{w} = w_1, \ldots, w_m$. Thus we have:

i. $u \rightsquigarrow v_1 \rightsquigarrow \cdots \rightsquigarrow v_n$ and $u \rightsquigarrow w_1 \rightsquigarrow \cdots \rightsquigarrow w_m$ in $\mathcal{T}$, and hence in $\mathcal{T}'$;

ii. $\mathcal{T} \models \mathrm{Tr}_{x_n}(\Sigma)[v_n]$ and $\mathcal{T} \models \mathrm{Tr}_{y_m}(\Gamma)[w_m]$; by item 1 of Proposition 76 this implies that $\mathcal{T}' \models \mathrm{Tr}_{x_n}(\Sigma_i)[v_n]$ and $\mathcal{T}' \models \mathrm{Tr}_{y_j}(\Gamma)[w_m]$;

iii. $v_n \sim w_m$ in $\mathcal{T}$, and hence in $\mathcal{T}'$.

Since Tr is truth-preserving, we have that for $\mathcal{T}', v_n \models \Sigma$ and $\mathcal{T}', w_m \models \Gamma$. Together with i and iii we conclude that Σ and Γ are $=^{\downarrow}$-satisfiable at $\mathcal{T}', u$.

For item 2, let Γ be a set of XPath$_=(\uparrow\downarrow)$-path expressions. Suppose Γ is $=^{\uparrow\downarrow}$-finitely satisfiable at $\mathcal{T}', u$ (the case for $\neq^{\uparrow\downarrow}$ -finitely satisfiable is analogous). We show that Γ is $=^{\uparrow\downarrow}$-satisfiable at $\mathcal{T}', u$.

Consider the following first-order $\sigma_{\{u\}}$-formula with free variable y:

$$\varphi(y) \;=\; (\exists x_0 \dots \exists x_n)(\exists y_0 \dots \exists y_m)[x_n = u \wedge y = y_m \wedge x_0 = y_0 \wedge \bigwedge_{i=0}^{n-1} x_i \rightsquigarrow x_{i+1} \wedge \bigwedge_{j=0}^{m-1} y_j \rightsquigarrow y_{j+1} \wedge x_n \sim y_m].$$

Define the following set of first-order $\sigma_{\{u\}}$-formulas: $\Delta(y) = \{\varphi(y)\} \cup \mathrm{Tr}_y(\Gamma)$. Let $\Delta'(y)$ be a finite subset of $\Delta(y)$. Since Γ is $=^{\uparrow\downarrow}_{n,m}$-finitely satisfiable at $\mathcal{T}', u$, then $\Delta'(y)$ is satisfiable and, by item 2 of Proposition 76, consistent with $\mathrm{Th}_{\{u\}}(\mathcal{T})$. By compactness, $\Delta(y)$ is satisfiable and consistent with $\mathrm{Th}_{\{u\}}(\mathcal{T})$. By 2-saturation, we conclude that $\Delta(y)$ is realizable in $\mathcal{T}$, say at w. Thus we have:

iv. There is $v \in T$ such that $v \xrightarrow{n} u$ and $v \xrightarrow{m} w$ in $\mathcal{T}$ and hence in $\mathcal{T}'$.

v. $\mathcal{T} \models \mathrm{Tr}_y(\Gamma)[w]$; by item 2 of Proposition 76 this implies that $\mathcal{T}' \models \mathrm{Tr}_y(\Gamma)[w]$;

vi. $u \sim w$ in $\mathcal{T}$, and hence in $\mathcal{T}'$.

Since Tr is truth-preserving, we have that $\mathcal{T}', w \models \Gamma$. Together with iv and vi we conclude that Γ is $=^{\uparrow\downarrow}$-satisfiable at $\mathcal{T}', u$. □

Let $(\mathcal{T}_i, u_i, v_i)_{i\in I}$ be a family of two-pointed data trees. The **ultraproduct** of such *two-pointed* data trees is defined, as usual, by $(\prod_U \mathcal{T}_i, u^*, v^*)$, where u^* and v^* are the ultralimits of $(u_i)_{i\in I}$ and $(u_i)_{i\in I}$ modulo U, respectively.

Example 78. For $i \in \mathbb{N}$, let $\mathcal{T}_i$ be any data tree of height at least i, and let u_i, v_i be any pair of nodes of T_i at distance i from each other. Let $\rho_n(x, y)$ be the first-order property "x is at distance at least n from y". It is clear that $\mathcal{T}_m \models \rho_n[u_m, v_m]$ for every $m \geq n$. Let u^* and v^* be the ultralimits of $(u_i)_{i\in I}$ and $(v_i)_{i\in I}$ modulo U. Since $\{m \mid m \geq n\} \in U$ for any non-principal U, we conclude that $\prod_U \mathcal{T}_i \models \rho_n[u^*, v^*]$ for every n, and so u^* is disconnected from v^* in $\prod_U \mathcal{T}_i$.

Hence, in general, two-pointed data trees are not closed under $\uparrow\downarrow$-quasi ultraproduct.

Let $k \geq 0$, let $\mathcal{T}$ be a data tree and let $u, v \in \mathcal{T}$. We say that $(\mathcal{T}, u, v)$ is a ***k*-bounded two-pointed data tree** if u, v are at distance at most k from the root of $\mathcal{T}$. In particular, if r is the root of $\mathcal{T}$ then $(\mathcal{T}, r, r)$ is a 0-bounded two-pointed data tree.

Let $n \geq 0$, let $\mathcal{T}$ be a data tree and let $u, v \in \mathcal{T}$. We say that a two-pointed data tree $\mathcal{T}, u, v$ is ***n*-two-pointed** if the minimum distance between u and v is at most n. That is, if w is the first common ancestor of u and v (i.e. the closest common ancestor), and $w \xrightarrow{c} u, w \xrightarrow{d} v$, then $c + d \leq n$.

Definition 79. Suppose $(\mathcal{T}_i, u_i, v_i)_{i\in I}$ is a family of n-two-pointed data trees, r_i is the root of $\mathcal{T}_i$, U is an ultrafilter over I, $\mathcal{T}^* = \prod_U \mathcal{T}_i$, and u^*, v^* and r^* are the ultralimits of $(u_i)_{i\in I}$, $(v_i)_{i\in I}$, and $(r_i)_{i\in I}$ modulo U respectively. Then

1. If $u_i \xrightarrow{*} v_i$, the **$\downarrow$-quasi ultraproduct** of $(\mathcal{T}_i, u_i, v_i)_{i\in I}$ modulo U is the n-two-pointed data tree $(\mathcal{T}^*|_{u^*}, u^*, v^*)$.

2. If $(\mathcal{T}_i, u_i, v_i)_{i\in I}$ is also a family of k-bounded data trees, the **$\uparrow\downarrow$-quasi ultraproduct** of $(\mathcal{T}_i, u_i, v_i)_{i\in I}$ modulo U is the k-bounded n-two-pointed data tree $(\mathcal{T}^*|_{r^*}, u^*, v^*)$.

Observe that in the definition of $\uparrow\downarrow$-quasi ultraproduct, u^* and v^* are effectively in $\mathcal{T}^*|_{r^*}$ for similar reasons as those in Proposition 22.

1.5.3 Definability and separation

In this subsection we state our definability results for two-pointed data trees. When we want to extend the proofs for the case of node expressions into the framework of path expressions, we find a problem: the language of path expressions does not have complementation or intersection in the general case (unlike node expressions, which have $\neg$ and $\wedge$). In order to be able to express these operations, we deal with restricted universes of two-pointed data trees. We start with the downward fragment:

Definability and separation via path expressions of XPath$_=(\downarrow)$

We work with n-two-pointed data trees which are **forward**, that is, data trees of the form $\mathcal{T}, u, v$ where $u \overset{\leq n}{\Rightarrow} v$. If α, β are XPath$_=(\downarrow)$-path expressions, we say that $\alpha \equiv^n \beta$ if for every forward n-two-pointed data tree $\mathcal{T}, u, v$ we have $\mathcal{T}, u, v \models \alpha$ iff $\mathcal{T}, u, v \models \beta$. For a path expression $\alpha = [\varphi_0]\downarrow \ldots \downarrow[\varphi_i]$ in simple normal form, we define the complement (over the class of forward n-two-pointed data trees) as

$$\sim_n \alpha = \begin{cases} \top & \text{if } i > n; \\ \bigcup_{0\leq j\leq i} \downarrow^j [\neg\varphi_j] \downarrow^{i-j} \cup \bigcup_{0\leq j\leq n, i\neq j} \downarrow^j & \text{otherwise.} \end{cases}$$

(We represent ε as $\downarrow^0$, and $\top$ as $\bigcup_{0\leq j\leq n} \downarrow^j$.) $\sim_n \alpha$ is thus true for all downward paths of a length at most n and different to that of α, and for paths of the same length as α but that do not satisfy some intermediate node expression $[\varphi_j]$. So for every forward n-two-pointed data tree $\mathcal{T}, u, v$, we have $\mathcal{T}, u, v \models \alpha$ iff $\mathcal{T}, u, v \not\models \sim_n \alpha$, that is, $\sim_n \alpha$ works as a kind of path expression negation over this restricted class of data trees. Notice that it is not possible to negate path expressions without a restriction on the class of data trees.

Recall that for XPath$_=(\downarrow)$-path expressions α, β in simple normal form and of the same length we have defined the intersection $\alpha \cap \beta$ in Definition 47. We extend this definition of intersection to path expressions in simple normal form, and of length at most n. Let α and β be path expressions in simple normal form, with $\mathsf{len}(\alpha) = i$, $\mathsf{len}(\beta) = j$. We define $\alpha \cap_n \beta$ as $\bot$ (we let $\bot$ to be $[\langle \varepsilon \neq \varepsilon \rangle]$) in case $i \neq j$ and as in (17) of Definition 47 otherwise. It is clear that for every forward n-two-pointed data tree $\mathcal{T}, u, v$, we have $\mathcal{T}, u, v \models \alpha \cap_n \beta$ iff $\mathcal{T}, u, v \models \alpha$ and $\mathcal{T}, u, v \models \beta$. These observations allow us to extend, *over the class of forward n-two-pointed data trees*, the operations of complement and intersection to any XPath$_=(\downarrow)$-path expression:

$$\alpha \equiv^n \bot \qquad (\alpha \text{ in simple normal form and } \mathsf{len}(\alpha) > n)$$

$$\sim_n (\alpha \cup \beta) \equiv^n (\sim_n \alpha) \cap_n (\sim_n \beta) \quad (\alpha, \beta \text{ in simple normal form and } \mathsf{len}(\alpha), \mathsf{len}(\beta) \leq n)$$
$$\sim_n (\alpha \cap_n \beta) \equiv^n (\sim_n \alpha) \cup (\sim_n \beta) \quad \text{(idem)}$$
$$(\alpha \cup \beta) \cap_n \gamma \equiv^n (\alpha \cap_n \gamma) \cup (\beta \cap_n \gamma) \quad \text{(idem)}$$
$$\alpha \cap_n \beta \equiv^n \beta \cap_n \alpha \quad \text{(idem)}$$

Therefore, when restricted to n-two pointed data trees, we can pretend to have complementation and intersection of path expressions with the standard meaning and within the language. This allows us to prove the results of definability for path expressions by making the adequate modifications to the proofs of Theorems 25 and 26. It is important to remark that these results are true only when restricting the universe to forward n-two-pointed data trees. In what follows, the universe of two-pointed data trees is restricted to those which are forward and n-two-pointed (for fixed n). Therefore, the operations of closure and complement must be taken with respect to this universe.

Theorem 80. *Let K be a class of forward n-two-pointed data trees. Then K is definable by a set of XPath$_=(\downarrow)$-path expressions iff K is closed under XPath$_=(\downarrow)$-bisimulations and $\downarrow$-quasi ultraproducts, and $\overline{K}$ is closed under $\downarrow$-quasi ultrapowers.*

Theorem 81. *Let K be a class of forward n-two-pointed data trees. Then K is definable by an XPath$_=(\downarrow)$-path expression iff both K and $\overline{K}$ are closed under XPath$_=(\downarrow)$-bisimulations and $\downarrow$-quasi ultraproducts.*

Like Theorem 80, the following result characterizes when a class of two-pointed data trees is definable by a single XPath$_=(\downarrow)$-path expression. However, instead of using the rather abstract notion of $\downarrow$-quasi ultraproducts, it uses the perhaps more natural notion of ℓ-bisimulation. It is analogous to Theorem 27 in the context of binary bisimulations.

Theorem 82. *Let K be a class of forward n-two-pointed data trees. Then K is definable by a path expression of XPath$_=(\downarrow)$ iff K is closed by ℓ-bisimulations for some ℓ.*

Theorems 37 and 38 can also be straightforwardly adapted:

Theorem 83. *Over forward n-two-pointed data trees: Let K_1 and K_2 be two disjoint classes such that K_1 is closed under XPath$_=(\downarrow)$-bisimulations and $\downarrow$-quasi ultraproducts and K_2 is closed under XPath$_=(\downarrow)$-bisimulations and $\downarrow$-quasi ultrapowers. Then there exists a third class K which is definable by a set of XPath$_=(\downarrow)$-path expressions, contains K_1, and is disjoint from K_2.*

Theorem 84. *Over forward n-two-pointed data trees: Let K_1 and K_2 be two disjoint classes closed under XPath$_=(\downarrow)$-bisimulations and $\downarrow$-quasi ultraproducts. Then there exists a third class K which is definable by an XPath$_=(\downarrow)$-path expression, contains K_1, and is disjoint from K_2.*

An example of use of these theorems is the following:

Example 85. **A class of two-pointed data trees definable by a single XPath$_=$(↑↓)-path expression but not definable by set of XPath$_=$(↓)-path expressions.** Let K be the class of forward 1-two-pointed data trees $\mathcal{T}, u, v$ such that v is a child of u and they have the same data value. On the one hand, this class is definable by the path expression $\alpha = \downarrow[\langle \varepsilon = \uparrow \rangle]$. On the other hand, the autobisimulation on $\mathcal{T}$ shown in Figure 20 shows that K is not closed under binary bisimulations for XPath$_=$(↓), since $\mathcal{T}, u, w$ is bisimilar to $\mathcal{T}, u, v$ but $data(u) \neq data(v)$. Thus, by Theorem 81, K is not definable by a set of XPath$_=$(↓)-path expressions.

Let us now move to the vertical fragment:

Definability and separation via path expressions of XPath$_=$(↑↓)

Not having complementation or intersection is more cumbersome in this case. We will state definability theorems restricted to classes of special two-pointed data trees which we denote ***n, m, k*-two-pointed** data trees. These are two-pointed data trees $\mathcal{T}, u, v$ such that if w is the first common ancestor of u and v (i.e. the closest common ancestor) then $w \xrightarrow{n} u$, $w \xrightarrow{m} v$, and w is at distance k from the root of $\mathcal{T}$. Observe that any n, m, k-two-pointed data tree is $k + \max\{n, m\}$-bounded.

If α, β are XPath$_=$(↑↓)-path expressions, we say that $\alpha \equiv^{n,m,k} \beta$ if for every n, m, k-two-pointed data tree $\mathcal{T}, u, v$ we have $\mathcal{T}, u, v \models \alpha$ iff $\mathcal{T}, u, v \models \beta$. The following equivalences, which are straightforward to verify, allows us to express in XPath$_=$(↑↓) the complementation $\sim_{n,m,k}$ and intersection $\cap_{n,m,k}$ over the class of n, m, k-two-pointed data trees. Here we use $\bot$ to represent the path expression $[\langle \varepsilon \neq \varepsilon \rangle] \uparrow^n \downarrow^m$.

$$[\varphi]\uparrow^{n'}\downarrow^{m'}[\psi] \equiv^{n,m,k} \begin{cases} [\varphi]\uparrow^n\downarrow^m[\psi] & \text{if } n - n' = m - m' \text{ and } n \leq n' \leq n + k \\ \bot & \text{otherwise} \end{cases}$$

$$\sim_{n,m,k} ([\varphi]\uparrow^n\downarrow^m[\psi]) \equiv^{n,m,k} ([\neg\varphi]\uparrow^n\downarrow^m) \cup (\uparrow^n\downarrow^m[\neg\psi])$$

$$([\varphi]\uparrow^n\downarrow^m[\psi]) \cap_{n,m,k} ([\varphi']\uparrow^n\downarrow^m[\psi']) \equiv^{n,m,k} [\varphi \wedge \varphi']\uparrow^n\downarrow^m[\psi \wedge \psi']$$

$$\sim_{n,m,k} (\alpha \cap_{n,m,k} \beta) \equiv^{n,m,k} (\sim_{n,m,k} \alpha) \cup (\sim_{n,m,k} \beta)$$

$$(\alpha \cup \beta) \cap_{n,m,k} \gamma \equiv^{n,m,k} (\alpha \cap_{n,m,k} \gamma) \cup (\beta \cap_{n,m,k} \gamma)$$

$$\alpha \cap_{n,m,k} \beta \equiv^{n,m,k} \beta \cap_{n,m,k} \alpha$$

In the last three equivalencies, α is of the form $[\varphi]\uparrow^n\downarrow^m[\psi]$ for some φ and ψ, and β is of the form $[\varphi']\uparrow^n\downarrow^m[\psi']$ for some φ' and ψ'.

In the first equivalence, the condition $n - n' = m - m'$ means that the navigation via $\uparrow^{n'}\downarrow^{m'}$ could actually connect the same nodes as $\uparrow^n\downarrow^m$, assuming the tree extends sufficiently upwards and a common ancestor is reached. The condition $n \leq n'$ assures that the upward portion of the navigation reaches at least the first common ancestor of the nodes, and the condition $n' \leq n + k$ means that $\uparrow^{n'}$ reaches at most up to the root of the tree, and not higher. Notice that if any of these conditions do not hold, the path expression $\uparrow^{n'}\downarrow^{m'}$ is

always false in the context of n, m, k-two-pointed data trees. If the three conditions hold simultaneously, then $\mathcal{T}, u, v \models \uparrow^{n'}\downarrow^{m'}$ for any n, m, k-two-pointed data tree $\mathcal{T}, u, v$.

It can be seen that, as expected, we arrive to results of definability and separation for XPath$_=(\uparrow\downarrow)$-path expressions, as in Theorems 80, 81, 83, and 84, but over the class of n, m, k-two-pointed data trees and using the notions of XPath$_=(\uparrow\downarrow)$-bisimulation and $\uparrow\downarrow$-quasi ultraproducts.

An example of use of these theorems is the following:

Example 86. **A class of two-pointed data trees definable in first-order (over data trees) but not definable by a set of XPath$_=(\uparrow\downarrow)$-path expressions.** Let K be the class of $1, 1, 0$-two-pointed data trees $\mathcal{T}, u, v$ such that u and v are both children of the root of $\mathcal{T}$, and they have the same data value. It is straightforward that this property is definable by a σ-first-order formula over the class of data trees. However, the autobisimulation between $\mathcal{T}$ and $\mathcal{T}'$ given in Figure 21 shows that K is not closed under binary bisimulations for XPath$_=(\uparrow\downarrow)$, as $\mathcal{T}, v, w$ is bisimilar to $\mathcal{T}', v', w'$ but $data(v') \neq data(w')$. Thus, by the corresponding theorem of definability, K is not definable by a set of XPath$_=(\uparrow\downarrow)$-path expressions.

Chapter 2

Axiomatizations

> En aquel Imperio, el Arte de la Cartografía logró tal Perfección que el Mapa de una sola Provincia ocupaba toda una Ciudad, y el mapa del Imperio, toda una Provincia. Con el tiempo, estos Mapas Desmesurados no satisficieron y los Colegios de Cartógrafos levantaron un Mapa del Imperio, que tenía el tamaño del Imperio y coincidía puntualmente con él.
>
> *Del rigor en la ciencia*
> Jorge Luis Borges

2.1 Introduction

In this chapter we develop the proof theory of $\mathrm{XPath}_{=}(\downarrow)$, designing an equational axiomatic system that only proves semantic truths of $\mathrm{XPath}_{=}(\downarrow)$, and such that all semantic truths of $\mathrm{XPath}_{=}(\downarrow)$ can be proved in this system. That is, we obtain a *sound* and *complete* axiomatization of $\mathrm{XPath}_{=}(\downarrow)$. Studying complete axiomatizations can give us an alternative method for solving the validity problem, which is undecidable for the full logic Core-Data-XPath [54], but it is decidable when the only axis present in the language is 'child', and in fact, also when adding 'descendant' [42] (and also for other fragments). Additionally, obtaining a complete axiomatization has applications in querying optimization through query rewriting. The idea here is to see equivalence axioms, which are of the form $\varphi \equiv \psi$, as (undirected) rules for the rewriting of queries; in this context, the completeness of the axiomatic system means that a semantic equivalence between two node or path expressions must have a corresponding chain of rewriting rules that transform the

first expression into the second one. Therefore, obtaining an axiomatization of XPath$_=(\downarrow)$, along with all the proofs of the theorems involved in the demonstration of its completeness, has the potential to be used as a first step in finding effective strategies for rewriting XPath-queries into equivalent but less complex forms.

To demonstrate soundness and completeness for Hilbert-style axiomatizations, one wants to see that a formula can be proved in the axiomatic system if and only it is valid (that is, it is satisfied in every possible model). More relevant to the framework we choose for this chapter, when using the rules of inference of equational logic one wants to see that an equivalence between two formulas can be proved (notated $\vdash \varphi \equiv \psi$) if and only if the truth value of both formulas coincides on all models (notated $\models \varphi \equiv \psi$). By the (finite) tree model property of BML, the validity of a formula with respect to the class of all Kripke models is equivalent to the validity in the class of (finite) tree-shaped Kripke models. Since there are truth-preserving translations to and from the data-oblivious Core-XPath, it is not surprising that there exist axiomatizations of the node expressions fragment of Core-XPath with 'child' as the only accessibility operator. Even more, there are also (equational) axiomatizations of all single axis fragments of Core-XPath (those where the only accessibility relation is the one of 'child', 'descendant', 'sibling', etc.), and also for the full Core-XPath language [102].

For the case of data-aware XPath$_=$, the resemblance with modal languages is now more distant, since, as we briefly indicated in §I.1.2, the models of XPath$_=$ cannot be represented by Kripke models. Indeed, finding an axiomatization in this case becomes more complex than for the navigational case. Our procedure involves devising a normal form theorem that shows that in our equational axiomatic system all consistent node expressions (resp. path expressions) can be proved equivalent to the disjunction (resp. union) of node (resp. path) expressions in normal form. Then we give a method for constructing, for every consistent node expression, a finite data tree where it is satisfied at the root. The construction of this tree is quite intricate, so we begin by giving an axiomatization for a simpler case, that of the syntactic fragment of XPath$_=(\downarrow)$ which we call XPath$_=(\downarrow)^-$, and is of independent interest.

2.1.1 Related work

As we mentioned before, there exist axiomatizations for *navigational* fragments of XPath with different axes [102]. Axiomatizations of other fragments of Core-XPath have been investigated in [13], and extensions with XPath 2.0 features have been addressed in [103]. We found only a few attempts of axiomatizing modal logics with some notion of data value.

A logical framework to reason about data organization is investigated in [9]. They introduce *reference structures* as the model to represent data storage, and a *propositional labeled modal language* to talk about such structures. Both together model memory configurations, i.e., they allow storing data files, and retrieving information about other cells' content and location of files. A sentence $[\![m]\!]A$ is read as "memory cell m stores sentence A". Then, data is represented by mean of sentences: for instance, if data c_i represents a

number N, c_i is the sentence "this is a number N" (same for other sorts of data). This representation is quite different from our approach. Nevertheless, according to our knowledge this is one of the first attempts on axiomatizing data-aware logics, by introducing a Hilbert-style axiomatization.

Tree Query Language (TQL) is a formalism based on *ambient logic* designed as a query language for semi-structured data. It allows checking schema properties, extracting tags satisfying a property and also recursive queries. The TQL data model is *information trees*, and the notation to talk about information trees is called *info-terms*. In [21] an axiomatization for info-terms is given in terms of a minimal congruence. This axiomatization is sound and complete with respect to the information tree semantics. This is more related with our approach in the sense that we consider data values as an equivalence relation.

The work most closely related to ours is [10], where an axiomatization was given for a very simple fragment of XPath$_=$, named *DataGL*. DataGL allows for constructions of the form $\langle \varepsilon = \downarrow_*[\varphi] \rangle$ and $\langle \varepsilon \neq \downarrow_*[\varphi] \rangle$, where φ is some node expression and $\downarrow_*$ denotes the navigational axis 'descendant', whose semantics is that of the reflexive transitive closure of the relation given by $\downarrow$. In particular, they introduce a sound and complete sequent calculus for this logic and derive PSPACE-completeness for the validity problem.

2.1.2 Contributions

We give sound and complete axiomatizations for XPath$_=(\downarrow)$ and for the subfragment XPath$_=(\downarrow)^-$. We extend the axiomatization of Core-XPath given in [102] with the needed axioms to obtain all validities of XPath$_=(\downarrow)$. Our axiomatizations are equational: inference rules will be the standard ones of equational logic, and all axioms are either of the form $\varphi \equiv \psi$ for node expressions φ and ψ or of the form $\alpha \equiv \beta$ for path expressions α and β. We show that an equivalence $\varphi \equiv \psi$ is derivable in the axiomatic system if and only if for any data tree, and any node x in it, either φ and ψ are true at x or both are false at x. We also present a similar result for path expressions: an equivalence $\alpha \equiv \beta$ is derivable if and only if for any data tree, and any pair of nodes (x, y) in it, either α and β are true at (x, y) or both are false at (x, y). Our completeness proof relies on a new normal form theorem for expressions of XPath$_=(\downarrow)$ (which, unlike those seen in Chapter 1, concerns axiomatically derivable equivalences rather than semantic equivalences), and a construction of a canonical model for any consistent formula in normal form.

We proceed gradually. To warm up, we first show an axiomatization for the fragment XPath$_=(\downarrow)^-$ of XPath$_=(\downarrow)$, which has all Boolean operators and data tests of the form $\langle \alpha = \beta \rangle$, but keeps out those of the form $\langle \alpha \neq \beta \rangle$. This fragment is still interesting since it allows us to express the *join* query constructor. Then we give the axiomatization for the full downward fragment XPath$_=(\downarrow)$, whose proof is more involved but uses some ideas from the simpler case.

2.1.3 Organization

In §2.2 we give the formal syntax and semantics of $\text{XPath}_=$ with 'child' axis, called $\text{XPath}_=(\downarrow)$. As we already mentioned, we also study a special syntactical fragment, called $\text{XPath}_=(\downarrow)^-$, whose data tests are all of the form $\langle \alpha = \beta \rangle$, keeping out those of the form $\langle \alpha \neq \beta \rangle$. In §2.3 we give a sound and complete axiomatic system for $\text{XPath}_=(\downarrow)^-$: in §2.3.1 we state the needed axioms, an extension of those introduced in [102]; in §2.3.2 we define the syntactic normal forms for $\text{XPath}_=(\downarrow)^-$ (these are not an extension of those defined in [102]) and state the corresponding normal form theorem; in §2.3.3 we show the completeness result, whose more complex part lies in proving that any node expression in normal form is satisfiable in a canonical model. In §2.4 we extend the previous axioms to get a sound and complete axiomatic system for $\text{XPath}_=(\downarrow)$. We follow the same route as for $\text{XPath}_=(\downarrow)^-$: axioms (§2.4.1), normal form (§2.4.2) and canonical model (§2.4.3). In §2.5 we close with some observations and a corollary of our axiomatizations. Some technical proofs were deferred to §2.6 to ease the readability of our main arguments.

2.2 Preliminaries

We define the node expressions TRUE and FALSE, and the path expression $\bot$, as follows:

$$\begin{array}{rcl} \text{TRUE} & \stackrel{\text{def}}{=} & \langle \varepsilon \rangle \\ \text{FALSE} & \stackrel{\text{def}}{=} & \neg\text{TRUE} \\ \bot & \stackrel{\text{def}}{=} & [\neg\langle \varepsilon \rangle] \end{array}$$

As we remark later, these expressions behave as expected in the axiomatic systems we design.

We notate $\text{XPath}_=(\downarrow)^-$ to the syntactic fragment which does not contain data tests of the form $\langle \alpha \neq \beta \rangle$. An **$\text{XPath}_=(\downarrow)^-$-formula** is either a node expression or a path expression of $\text{XPath}_=(\downarrow)^-$.

We remark that in this chapter we use, from §1.2.2, the definition of the **downward depth** of an $\text{XPath}_=(\downarrow)$-expression, notated $\mathtt{dd}$.

Data trees. We will work with an abstraction of the usual definition of data tree: instead of having data values in each node of the tree, we have an equivalence relation between the nodes or, equivalently, a partition. We identify two nodes with the same data value as being related by the equivalence relation, or belonging to the same equivalence class in the partition. This is more convenient for our purposes, and, as already mentioned before, it is equivalent to the data-domain formulation as far as the semantics of $\text{XPath}_=$ is concerned. With this perspective, we have:

Definition 87. Given $\mathbb{A}$ a finite set of *labels*, a *data tree* $\mathcal{T}$ is a pair (T, π), where T is a tree whose nodes are labeled with elements from $\mathbb{A}$, and π is a partition over the nodes of T. We denote with $[x]_\pi$ the class of x in the partition π.

Recall from §I.1.2 that we say that two node expressions φ, ψ of XPath$_=(\downarrow)$ are equivalent iff $[\![\varphi]\!]^{\mathcal{T}} = [\![\psi]\!]^{\mathcal{T}}$ for all data trees $\mathcal{T}$. In this chapter, we notate this as $\models \varphi \equiv \psi$ to emphasize the semantical nature of this equivalence and distinguish it from the purely syntactical equivalences that we introduce later. That is, we have:

$$\models \varphi \equiv \psi \overset{\text{def}}{\Leftrightarrow} [\![\varphi]\!]^{\mathcal{T}} = [\![\psi]\!]^{\mathcal{T}} \text{ for all data trees } \mathcal{T}.$$

Similarly, path expressions α, β of XPath$_=(\downarrow)$ are equivalent (notated $\models \alpha \equiv \beta$) iff for all data trees $\mathcal{T}$ the semantics of α and β coincide:

$$\models \alpha \equiv \beta \overset{\text{def}}{\Leftrightarrow} [\![\alpha]\!]^{\mathcal{T}} = [\![\beta]\!]^{\mathcal{T}} \text{ for all data trees } \mathcal{T}.$$

Let $\mathcal{T}, x, y$ and $\mathcal{T}', x', y'$ be two-pointed data trees, we say that $\mathcal{T}, x \equiv^{\downarrow -} \mathcal{T}', x'$ iff for all node expressions φ of XPath$_=(\downarrow)^-$ we have $\mathcal{T}, x \models \varphi$ iff $\mathcal{T}', x' \models \varphi$, and we say that $\mathcal{T}, x, y \equiv^{\downarrow -} \mathcal{T}', x', y'$ iff for all path expressions α of XPath$_=(\downarrow)^-$ $\mathcal{T}, x, y \models \alpha$ iff $\mathcal{T}', x', y' \models \alpha$.

Let $\mathcal{T} = (T, \pi)$ be a data tree. When T' is a subset of T, we write $\pi|_{T'}$ to denote the restriction of the partition π to T'. Let $x \in T$, and let X be the set of x and all its descendants in T, i.e. $X = \{x\} \cup \{y \in T \mid (\exists i \geq 1)\ x \overset{i}{\rightarrow} y\}$. In this context, $\boldsymbol{\mathcal{T}|_x}$ refers to the data tree $(T|_X, \pi|_X)$, that is, the data tree that consists of the subtree of T that is hanging from x, maintaining the partition of that portion.

We make the following observation:

Remark 88. *Let $(\mathcal{T}, \pi)$ be a data tree. Then:*

- $\mathcal{T}, x \equiv^{\downarrow} \mathcal{T}|_x, x$.
- *If y, z are descendants of x in $\mathcal{T}$, then $\mathcal{T}, y, z \equiv^{\downarrow} \mathcal{T}|_x, y, z$.*

Inference rules. An XPath$_=(\downarrow)$-**node equivalence** is an expression of the form $\varphi \equiv \psi$, where φ, ψ are node expressions of XPath$_=(\downarrow)$. An XPath$_=(\downarrow)$-**path equivalence** is an expression of the form $\alpha \equiv \beta$, where α, β are path expressions. An **axiom** is either a node equivalence or a path equivalence.

For P, Q both path expressions or both node expressions, we say that $P \equiv Q$ is *derivable* (or also that P is **provably equivalent** to Q) from a given set of axioms Σ (notated $\boldsymbol{\Sigma \vdash P \equiv Q}$) if it can be obtained from them using the standard rules of equational logic:

1. $P \equiv P$.
2. If $P \equiv Q$, then $Q \equiv P$.
3. If $P \equiv Q$ and $Q \equiv R$, then $P \equiv R$.
4. If $P \equiv Q$ and R' is obtained from R by replacing some occurrences of P by Q, then $R \equiv R'$.

We utilize the following abbreviations:

$$\varphi \leq \psi \overset{\text{def}}{\Leftrightarrow} \varphi \vee \psi \equiv \psi$$
$$\alpha \leq \beta \overset{\text{def}}{\Leftrightarrow} \alpha \cup \beta \equiv \beta.$$

Definition 89 (Consistent Node and Path Expressions)**.** Let Σ be a set of axioms. We say that a node expression [resp. path expression] P of $\mathrm{XPath}_{=}(\downarrow)$ is **Σ-consistent** if $\Sigma \nvdash P \equiv \textsc{false}$ [resp. $\Sigma \nvdash P \equiv \bot$]. We define $\mathbf{Con}_{\Sigma}$ as the set of **Σ-consistent node expressions**.

2.3 Axiomatic System for $\mathrm{XPath}_{=}(\downarrow)^{-}$

2.3.1 Axiomatization

The main theorems of this article are the ones about the completeness of the proposed axiomatizations. These theorems have two main ingredients: one is a normal form theorem that allows to rewrite any consistent node or path expression in terms of normal forms. The other one is the construction of a canonical model for any consistent node expression in normal form. As it is usually the case, at the same time, we give (through the set of axioms) the definition of consistency. So, an axiom (or an axiom scheme) could have been added either because it was needed to prove the normal form theorem or because it was needed to guarantee that every unsatisfiable formula is inconsistent —the key fact is that we have a much better intuition of what should be satisfiable than of what should be consistent. Of course we should be careful that the added axioms are sound but that is quite intuitive.

In Table 2.1 we list the axiom schemes for the fragment $\mathrm{XPath}_{=}(\downarrow)^{-}$. This list includes all the axiom schemes from [102] for the logic Core-XPath with single axis 'child' (second, third and fourth blocks) and adds the new axiom schemes for data tests of the form $\langle \alpha = \beta \rangle$ (last block). Also, remember that in our data trees each node satisfies exactly one label. We add two axiom schemes to handle this issue (first block). This is unessential for our development, and could be dropped to axiomatize $\mathrm{XPath}_{=}(\downarrow)$ over data trees whose every node is tagged with multiple labels, with minor changes to the definitions of normal forms.

Let XP^{-} be the set of all instantiations of the axiom schemes of Table 2.1 for a fixed alphabet $\mathbb{A}$. In the scope of this section we will often say that a node expression is *consistent* meaning that it is XP^{-}-consistent (as in Definition 89).

Observe that PrAx4 from [102, Table 3], defined by

PrAx4	$(\alpha\beta)[\varphi]$	$\equiv$	$\alpha(\beta[\varphi])$

is not present in our axiomatization because, due to our language design, it is a particular case of IsAx4.

The syntactic equivalences of Fact 90 will be useful for the next sections:

Axioms for labels			
LbAx1	TRUE	$\equiv$	$\bigvee_{a\in\mathbb{A}} a$
LbAx2	FALSE	$\equiv$	$a \wedge b$ (where $a \neq b$)
Path axiom schemes for predicates			
PrAx1	$(\alpha[\neg\langle\beta\rangle])\beta$	$\equiv$	$\bot$
PrAx2	$[\text{TRUE}]$	$\equiv$	ε
PrAx3	$[\varphi \vee \psi]$	$\equiv$	$[\varphi] \cup [\psi]$
Path axiom schemes for idempotent semirings			
IsAx1	$(\alpha \cup \beta) \cup \gamma$	$\equiv$	$\alpha \cup (\beta \cup \gamma)$
IsAx2	$\alpha \cup \beta$	$\equiv$	$\beta \cup \alpha$
IsAx3	$\alpha \cup \alpha$	$\equiv$	α
IsAx4	$\alpha(\beta\gamma)$	$\equiv$	$(\alpha\beta)\gamma$
IsAx5	$\varepsilon\alpha$	$\equiv$	α
	$\alpha\varepsilon$	$\equiv$	α
IsAx6	$\alpha(\beta \cup \gamma)$	$\equiv$	$(\alpha\beta) \cup (\alpha\gamma)$
	$(\alpha \cup \beta)\gamma$	$\equiv$	$(\alpha\gamma) \cup (\beta\gamma)$
IsAx7	$\bot \cup \alpha$	$\equiv$	α
Node axiom schemes			
NdAx1	φ	$\equiv$	$\neg(\neg\varphi \vee \psi) \vee \neg(\neg\varphi \vee \neg\psi)$
NdAx2	$\langle[\varphi]\rangle$	$\equiv$	φ
NdAx3	$\langle\alpha \cup \beta\rangle$	$\equiv$	$\langle\alpha\rangle \vee \langle\beta\rangle$
NdAx4	$\langle\alpha\beta\rangle$	$\equiv$	$\langle\alpha[\langle\beta\rangle]\rangle$
Node axiom schemes for equality			
EqAx1	$\langle\alpha = \beta\rangle$	$\equiv$	$\langle\beta = \alpha\rangle$
EqAx2	$\langle\alpha \cup \beta = \gamma\rangle$	$\equiv$	$\langle\alpha = \gamma\rangle \vee \langle\beta = \gamma\rangle$
EqAx3	$\varphi \wedge \langle\alpha = \beta\rangle$	$\equiv$	$\langle[\varphi]\alpha = \beta\rangle$
EqAx4	$\langle\alpha = \beta\rangle$	$\leq$	$\langle\alpha\rangle$
EqAx5	$\langle\gamma[\langle\alpha = \beta\rangle]\rangle$	$\leq$	$\langle\gamma\alpha = \gamma\beta\rangle$
EqAx6	$\langle\alpha = \alpha\rangle$	$\equiv$	$\langle\alpha\rangle$
EqAx7	$\langle\alpha = \varepsilon\rangle \wedge \langle\beta = \varepsilon\rangle$	$\leq$	$\langle\alpha = \beta\rangle$
EqAx8	$\langle\alpha = \beta[\langle\varepsilon = \gamma\rangle]\rangle$	$\leq$	$\langle\alpha = \beta\gamma\rangle$

Table 2.1: Axiomatic system XP^- for XPath$_=(\downarrow)^-$

Fact 90. *As seen in [102],* TRUE, FALSE, *and* $\bot$ *behave as expected:* $\mathsf{XP}^- \vdash \varphi \vee \text{TRUE} \equiv \text{TRUE}$, $\mathsf{XP}^- \vdash \alpha[\text{FALSE}] \equiv \bot$, *et cetera. Furthermore, we have the following from [102, Table 6]:*

Der1	$\mathsf{XP}^- \vdash \varphi \vee \psi \equiv \psi \vee \varphi$
Der2	$\mathsf{XP}^- \vdash \varphi \vee (\psi \vee \rho) \equiv (\varphi \vee \psi) \vee \rho$
Der12	$\mathsf{XP}^- \vdash \langle \alpha\beta \rangle \leq \langle \alpha \rangle$
Der13	$\mathsf{XP}^- \vdash \langle \alpha[\text{FALSE}] \rangle \equiv \text{FALSE}$
Der21	$\mathsf{XP}^- \vdash \alpha[\varphi][\psi] \equiv \alpha[\varphi \wedge \psi]$

We note that in order to prove the previous derivations one needs to use PrAx1, PrAx2, PrAx3, IsAx1, IsAx2, IsAx4, IsAx5, IsAx6, IsAx7, NdAx1, NdAx2, NdAx3 *and* NdAx4.[14]

As a consequence of **Der1**, **Der2** *and Huntington's equation* NdAx1, *we can derive all the axioms of Boolean algebras from the axioms in* XP^- *[64, 63]. In what follows, we will often use the Boolean properties without explicitly referencing them. In particular, we use the fact that* $\mathsf{XP}^- \vdash \psi \leq \text{FALSE}$ *implies that* ψ *is an inconsistent node expression, and that* $\mathsf{XP}^- \vdash \varphi \leq \psi$ *implies that* $\varphi \wedge \neg\psi$ *is inconsistent.*

Sometimes we use IsAx1, IsAx4, EqAx1, EqAx4, and EqAx6 without explicitly mentioning them. We omit such steps in order to make the proofs more readable.

It is not difficult to see that the axioms XP^- are sound for $\text{XPath}_=(\downarrow)^-$:

Proposition 91 (Soundness of $\text{XPath}_=(\downarrow)^-$)**.**

1. *Let* φ *and* ψ *be node expressions of* $\text{XPath}_=(\downarrow)^-$. *Then* $\mathsf{XP}^- \vdash \varphi \equiv \psi$ *implies* $\models \varphi \equiv \psi$.

2. *Let* α *and* β *be path expressions of* $\text{XPath}_=(\downarrow)^-$. *Then* $\mathsf{XP}^- \vdash \alpha \equiv \beta$ *implies* $\models \alpha \equiv \beta$.

Proof. Equational rules are valid because we have compositional semantics, and the proof that all the axioms schemes from Table 2.1 are sound is straightforward. □

2.3.2 Normal forms

When working in Core-XPath, the only expressions of the form $\langle\cdot\rangle$ (which we call 'diamonds') in the language are those of the type $\langle\alpha\rangle$. In the absence of data tests, any node expression $\langle[\varphi]\downarrow\beta\rangle$ is equivalent to $\varphi \wedge \langle\downarrow[\langle\beta\rangle]\rangle$. Hence, when the only axis is 'child', the only path expressions that are needed to write node expressions are those of the form $\downarrow[\langle\cdot\rangle]$, of length 1, and therefore the only 'diamonds' that we need are of the form $\langle\downarrow[\psi]\rangle$, which in the basic modal logic would be written simply as $\Diamond\psi$. This rewriting of path expressions is carried out in [102], and so normal forms have somewhat the same flavour as in the basic modal logic.

When data shows up, this rewriting is no longer possible: the node expression $\langle\alpha = \beta\rangle$ checks if there are nodes with equal data value *at the end of paths* α *and* β. So these

[14] **Der1** uses IsAx2, NdAx2, and NdAx3. **Der2** uses IsAx1 (and NdAx2 and NdAx3 again). We can now derive all the axioms of Boolean algebras by also using NdAx1. **Der12** also uses PrAx2, PrAx3, IsAx4, IsAx5, IsAx6, and NdAx4. **Der13** does not need further axioms. **Der21** also uses PrAx1 and IsAx7.

paths cannot be compressed as before. For an easy example, observe that the data-aware 'diamond' $\langle\downarrow[a]\downarrow[b] = \varepsilon\rangle$ is not equivalent to $\langle\downarrow[a \wedge \langle\downarrow[b]\rangle] = \varepsilon\rangle$.

The normal forms that we introduce are inspired by the classic *Disjunctive Normal Form* (DNF) for propositional logic, in which literals are formulas of smaller depth. Our normal forms take into account path expressions of arbitrary length, and this makes our definition more involved than the one in [102]. We introduce them in this subsection for the language XPath$_=(\downarrow)^-$. This definition is extended to the general logic XPath$_=(\downarrow)$ in §2.4.2.

We define the sets P_n^- and N_n^-, which contain the path and node expressions of XPath$_=(\downarrow)^-$, respectively, in normal form at level n:

Definition 92 (Normal form for XPath$_=(\downarrow)^-$)**.**

$$\begin{aligned}
P_0^- &= \{\varepsilon\} \\
N_0^- &= \{a \wedge \langle\varepsilon = \varepsilon\rangle \mid a \in \mathbb{A}\} \\
P_{n+1}^- &= \{\varepsilon\} \cup \left\{\downarrow[\psi]\beta \mid \psi \in N_n^-, \beta \in P_n^-\right\} \\
D_{n+1}^- &= \left\{\langle\alpha = \beta\rangle \mid \alpha, \beta \in P_{n+1}^-\right\} \\
N_{n+1}^- &= \left\{a \wedge \bigwedge_{\varphi \in C} \varphi \wedge \bigwedge_{\varphi \in D_{n+1}^- \setminus C} \neg\varphi \mid C \subseteq D_{n+1}^-, a \in \mathbb{A}\right\} \cap \mathbf{Con}_{\mathsf{XP}^-}.
\end{aligned}$$

Observe that we define normal forms by mutual recursion among three kinds of sets: P_n^-, D_n^- and N_n^- (for some n), which are sets of path expressions, data-aware diamonds, and node expressions, respectively. They consist of expressions that can look forward up to a certain downward depth. The index n indicates which maximum downward depth we are exploring, both in path and node expressions. Base cases are the simplest expressions of each kind (depth 0). New path expressions are constructed by using node and path expressions ψ and β from a previous level of their respective type, and exploring one more step using $\downarrow$. Data-aware diamond expressions are auxiliary expressions consisting of equalities between two path expressions of the same level. Finally, node expressions in normal form at some level n are formed of consistent conjunctions of positive and negative data-aware diamond expressions of level n. Notice that at each level i, each conjunction in N_i^- has one conjunct of the form a with $a \in \mathbb{A}$ which provides a label for the current node. Finally, let us remark that it would suffice that N_0^- contains formulas of the form a, for $a \in \mathbb{A}$. However, we include instead formulas of the form $a \wedge \langle\varepsilon = \varepsilon\rangle$ (containing the tautology $\langle\varepsilon = \varepsilon\rangle$) only for technical reasons.

Example 93. Let us see some examples of expressions in normal form. We consider only two labels a and b, and ignore redundancies (if we write $\langle\alpha = \beta\rangle$, we do not write $\langle\beta = \alpha\rangle$). The sets P_1^- and D_1^- are as follows:

$$\begin{aligned}
P_1^- &= \{\downarrow[a \wedge \langle\varepsilon = \varepsilon\rangle]\varepsilon, \downarrow[b \wedge \langle\varepsilon = \varepsilon\rangle]\varepsilon, \varepsilon\} \\
D_1^- &= \{\langle\downarrow[a \wedge \langle\varepsilon = \varepsilon\rangle]\varepsilon = \downarrow[b \wedge \langle\varepsilon = \varepsilon\rangle]\varepsilon\rangle, \langle\varepsilon = \downarrow[a \wedge \langle\varepsilon = \varepsilon\rangle]\varepsilon\rangle, \langle\varepsilon = \downarrow[b \wedge \langle\varepsilon = \varepsilon\rangle]\varepsilon\rangle,
\end{aligned}$$

$$\langle\downarrow[a \wedge \langle \varepsilon = \varepsilon\rangle]\varepsilon = \downarrow[a \wedge \langle \varepsilon = \varepsilon\rangle]\varepsilon\rangle, \langle\downarrow[b \wedge \langle \varepsilon = \varepsilon\rangle]\varepsilon = \downarrow[b \wedge \langle \varepsilon = \varepsilon\rangle]\varepsilon\rangle, \langle \varepsilon = \varepsilon\rangle\}$$

An example of a node expression in normal form at level 1, i.e. a node expression in N_1^-, is

$$\begin{aligned}\varphi \;=\; & a \wedge \langle \varepsilon = \varepsilon\rangle \wedge \langle\downarrow[a \wedge \langle \varepsilon = \varepsilon\rangle]\varepsilon = \downarrow[b \wedge \langle \varepsilon = \varepsilon\rangle]\varepsilon\rangle \wedge \langle\downarrow[a \wedge \langle \varepsilon = \varepsilon\rangle]\varepsilon = \downarrow[a \wedge \langle \varepsilon = \varepsilon\rangle]\varepsilon\rangle \\ & \wedge\langle\downarrow[b \wedge \langle \varepsilon = \varepsilon\rangle]\varepsilon = \downarrow[b \wedge \langle \varepsilon = \varepsilon\rangle]\varepsilon\rangle \wedge \neg\langle \varepsilon = \downarrow[a \wedge \langle \varepsilon = \varepsilon\rangle]\varepsilon\rangle \wedge \neg\langle \varepsilon = \downarrow[b \wedge \langle \varepsilon = \varepsilon\rangle]\varepsilon\rangle.\end{aligned}$$

The following lemmas (94, 95 and 96) are very intuitive and their proofs are straightforward.

Lemma 94. *Let $\psi \in N_n^-$ and $\alpha, \beta \in P_n^-$. Let $\mathcal{T}, x$ be a pointed data tree, such that $\mathcal{T}, x \models \psi$ and $\mathcal{T}, x \models \langle \alpha = \beta\rangle$. Then $\langle \alpha = \beta\rangle$ is a conjunct of ψ.*

Proof. The case when $n = 0$ follows from the definitions of P_0^- and N_0^-. If $n > 0$, since $\alpha, \beta \in P_n^-$, by definition of D_n^-, we have $\langle \alpha = \beta\rangle \in D_n^-$. Because $\psi \in N_n^-$, either $\langle \alpha = \beta\rangle$ or its negation is a conjunct of ψ. Suppose that the latter occurs, then $\mathcal{T}, x \models \neg\langle \alpha = \beta\rangle$, and, by hypothesis, $\mathcal{T}, x \models \langle \alpha = \beta\rangle$, which is a contradiction. □

Lemma 95. *Let $\psi \in N_n^-$ and $\alpha \in P_n^-$. If $[\psi]\alpha$ is consistent then $\langle \alpha = \alpha\rangle$ is a conjunct of ψ. As an immediate consequence, if $\langle\downarrow[\psi]\alpha\rangle$ is consistent then $\langle \alpha = \alpha\rangle$ is a conjunct of ψ.*

Proof. Since $\alpha \in P_n^-$, then either $\alpha = \varepsilon$ or α is of the form $\downarrow[\psi_1] \ldots \downarrow[\psi_k]\varepsilon$ for some $1 \leq k \leq n$, and $\psi_i \in N_{n-i}^-$. If $\alpha = \varepsilon$, we are done, as $\langle \varepsilon = \varepsilon\rangle$ is always a conjunct of ψ by consistency. Else, since $\langle \alpha = \alpha\rangle \in D_n^-$, $\langle \alpha = \alpha\rangle$ or its negation is a conjunct of ψ. By using **Der 21** from Fact 90, EqAx6 and PrAx1 consecutively, one can see that the latter case is not possible, because $[\psi]\alpha$ is consistent. Then $\langle \alpha = \alpha\rangle$ is a conjunct of ψ. □

Lemma 96. *For every pair of distinct elements $\varphi, \psi \in N_n^-$, $\varphi \wedge \psi$ is inconsistent.*

Proof. If $n = 0$, then $\varphi = a \wedge \langle \varepsilon = \varepsilon\rangle$ and $\psi = b \wedge \langle \varepsilon = \varepsilon\rangle$, with $a, b \in \mathbb{A}$ and $a \neq b$. Then by LbAx2, we have $\mathsf{XP}^- \vdash \varphi \wedge \psi \equiv$ FALSE, i.e., $\varphi \wedge \psi$ is inconsistent.

Let φ and ψ be distinct normal forms of degree $n > 0$, then we have two possibilities:

- If φ and ψ differ in the conjunct of the form a with $a \in \mathbb{A}$, then we use an argument similar to the one used for the base case.
- If not, then there is $\sigma \in D_n^-$ such that, without loss of generality, σ is a conjunct of φ and $\neg\sigma$ is a conjunct of ψ. Therefore, because $\varphi \wedge \psi$ contains $\sigma \wedge \neg\sigma$ as a sub-expression, we have $\mathsf{XP}^- \vdash \varphi \wedge \psi \equiv$ FALSE, i.e., it is inconsistent.

This concludes the proof. □

Lemma 97. *Let $\alpha, \beta \in P_n^-$. If there is a data tree $\mathcal{T}$ and nodes $x, y \in T$ such that $\mathcal{T}, x, y \models \alpha$ and $\mathcal{T}, x, y \models \beta$, then $\alpha = \beta$.*

Proof. Let $\alpha = \downarrow[\psi_1]...\downarrow[\psi_i]\varepsilon$ and $\beta = \downarrow[\rho_1]...\downarrow[\rho_j]\varepsilon$. By definition of the semantics of XPath$_=(\downarrow)^-$, $i = j$ since $\mathcal{T}, x, y \models \alpha$ and $\mathcal{T}, x, y \models \beta$. In particular, there are nodes $z_i \in T$, $1 \leq k \leq i$, such that $\mathcal{T}, z_i \models \psi_k, \mathcal{T}, z_i \models \rho_k$ for all $1 \leq k \leq i$. Using Proposition 91, we obtain that $\psi_k \wedge \rho_k$ is consistent for all $k = 1, \ldots, i$, and thus by Lemma 96 we have that $\psi_k = \rho_k$ for all $k = 1, \ldots, i$. Then we conclude that $\alpha = \beta$. □

The following lemma is a normal form result for the special case of data-aware 'diamond' node expressions in D_n^-:

Lemma 98. *Let $n > 0$ and $a \in \mathbb{A}$. If $\varphi \in D_n^-$ is consistent then there are $\psi_1, \ldots, \psi_k \in N_n^-$ such that $\mathsf{XP}^- \vdash a \wedge \varphi \equiv \bigvee_i \psi_i$*

Proof. Take

$$\psi = \bigvee \left(\left\{ a \wedge \bigwedge_{\psi \in C} \psi \wedge \bigwedge_{\psi \in D_n^- \setminus C} \neg\psi \mid \ C \subseteq D_n^-, \varphi \in C \right\} \cap \mathbf{Con}_{\mathsf{XP}^-} \right).$$

It can be seen that $\mathsf{XP}^- \vdash a \wedge \varphi \equiv \psi$. Notice that the above disjunction is not empty. Indeed, let $D_n^- \setminus \{\varphi\} = \{\psi_1, \ldots, \psi_k\}$, and define $\varphi_0 = a \wedge \varphi$ and $\varphi_{i+1} = \varphi_i \wedge \psi_{i+1}$ if $\varphi_i \wedge \psi_{i+1}$ is consistent and $\varphi_{i+1} = \varphi_i \wedge \neg\psi_{i+1}$ otherwise. By NdAx1 either $\varphi_i \wedge \psi_{i+1}$ or $\varphi_i \wedge \neg\psi_{i+1}$ is consistent, and hence φ_i is consistent for all i. This means that φ_k is consistent and hence it is one of the disjuncts of the above formula. □

The next lemma states that expressions in any P_n^- or N_n^- are provably equivalent to the union or disjunction, respectively, of expressions in higher levels of P_n^- or N_n^-.

Lemma 99. *Let $m > n$. If $\varphi \in N_n^-$ then there are $\varphi_1 \ldots \varphi_k \in N_m^-$ such that $\mathsf{XP}^- \vdash \varphi \equiv \bigvee_i \varphi_i$. If $\alpha \in P_n^-$ then there are $\alpha_1 \ldots \alpha_k \in P_m^-$ such that $\mathsf{XP}^- \vdash \alpha \equiv \bigcup_i \alpha_i$.*

Proof. Observe that it suffices to show this result for $m = n + 1$.

The basic idea is to proceed by induction over n, first proving the result for P_n^- and then using that for the case of N_n^-.

The base case for P_0^- is trivial, while the case for $\varphi \in N_0^-$ is easy by taking the disjunction of all node expressions in N_1^- which contain the same label as φ as a conjunct.

Now for the inductive case $\alpha = P_{n+1}^-$, if $\alpha = \varepsilon$ then the result is trivial, and otherwise $\alpha = \downarrow[\psi]\beta$ with $\psi \in N_n^-$ and $\beta \in P_n^-$. We now use the inductive hypothesis on ψ and β and distribute into a union in P_{n+2}^- using PrAx3 and the path axiom schemes for idempotent semirings. The case $\varphi \in N_{n+1}^-$ is solved similarly, using that we know the result holds for path expressions in P_{n+1}^-.

□

It is easier to prove that every consistent formula is satisfiable over expressions in normal form than over the general case, as we can rely on the particular structure of those expressions. However, these proofs would be of little use if expressions in normal form only represented a small subset of all possible expressions. That is not really the case: Theorem 101 below will show that all node expressions (and also all path expressions) are provably equivalent to a disjunction of expressions in normal form.

Example 100. As a simple example of these equivalences, take the language with only three labels a, b, and c, and consider the node expression $\varphi = \neg a$. Then $\mathsf{XP}^- \vdash \varphi \equiv (b \wedge \langle \varepsilon = \varepsilon \rangle) \vee (c \wedge \langle \varepsilon = \varepsilon \rangle)$, where $b \wedge \langle \varepsilon = \varepsilon \rangle$ and $c \wedge \langle \varepsilon = \varepsilon \rangle$ are node expressions in N_0^-.

For a slightly more complex example, related with Example 93, take the language with only the labels a and b, and consider the node expression

$$\varphi = \langle [a]{\downarrow}[a] = {\downarrow}[b] \rangle \wedge \neg \langle \varepsilon = {\downarrow}[a] \rangle.$$

Then $\mathsf{XP}^- \vdash \varphi \equiv \psi_1 \vee \psi_2$, where

$$\begin{aligned}
\psi_1 &= \psi \wedge \neg \langle \varepsilon = {\downarrow}[b \wedge \langle \varepsilon = \varepsilon \rangle] \varepsilon \rangle \\
\psi_2 &= \psi \wedge \langle \varepsilon = {\downarrow}[b \wedge \langle \varepsilon = \varepsilon \rangle] \varepsilon \rangle \\
\psi &= a \wedge \langle \varepsilon = \varepsilon \rangle \wedge \langle {\downarrow}[a \wedge \langle \varepsilon = \varepsilon \rangle] \varepsilon = {\downarrow}[b \wedge \langle \varepsilon = \varepsilon \rangle] \varepsilon \rangle \wedge \langle {\downarrow}[a \wedge \langle \varepsilon = \varepsilon \rangle] \varepsilon = {\downarrow}[a \wedge \langle \varepsilon = \varepsilon \rangle] \varepsilon \rangle \wedge \\
&\quad \wedge \langle {\downarrow}[b \wedge \langle \varepsilon = \varepsilon \rangle] \varepsilon = {\downarrow}[b \wedge \langle \varepsilon = \varepsilon \rangle] \varepsilon \rangle \wedge \neg \langle \varepsilon = {\downarrow}[a \wedge \langle \varepsilon = \varepsilon \rangle] \varepsilon \rangle
\end{aligned}$$

Theorem 101 (Normal form for $\mathrm{XPath}_=(\downarrow)^-$)**.** *Let φ be a consistent node expression of $\mathrm{XPath}_=(\downarrow)^-$ such that $\mathsf{dd}(\varphi) = n$. Then $\mathsf{XP}^- \vdash \varphi \equiv \bigvee_i \varphi_i$ for some $(\varphi_i)_{1 \le i \le k} \in N_n^-$. Let α be a consistent path expression of $\mathrm{XPath}_=(\downarrow)^-$ such that $\mathsf{dd}(\alpha) = n$. Then $\mathsf{XP}^- \vdash \alpha \equiv \bigcup_i [\varphi_i]\alpha_i$ for some $(\alpha_i)_{1 \le i \le k} \in P_n^-$ and $(\varphi_i)_{1 \le i \le k} \in N_n^-$. Furthermore, if α is ε or starting with $\downarrow$ then $\mathsf{XP}^- \vdash \alpha \equiv \bigcup_i \alpha_i$ for some $(\alpha_i)_{1 \le i \le k} \in P_n^-$.*

Proof. We show that if F is a consistent formula of $\mathrm{XPath}_=(\downarrow)^-$ such that $\mathsf{dd}(F) = n$, then

a) if F is a node expression then $\mathsf{XP}^- \vdash F \equiv \bigvee_i \psi_i$ for some $(\psi_i)_{1 \le i \le k} \in N_n^-$;

b) if F is a path expression then $\mathsf{XP}^- \vdash F \equiv \bigcup_i [\varphi_i]\alpha_i$ for some $(\alpha_i)_{1 \le i \le k} \in P_n^-$ and $(\varphi_i)_{1 \le i \le k} \in N_n^-$; furthermore, if F is ε or starts with $\downarrow$, then $\mathsf{XP}^- \vdash F \equiv \bigcup_i \alpha_i$ for some $(\alpha_i)_{1 \le i \le k} \in P_n^-$.

Because of EqAx6, it is enough to prove the lemma for the fragment of $\mathrm{XPath}_=(\downarrow)^-$ without diamonds of the form $\langle \alpha \rangle$. We proceed by induction on the complexity of F, denoted by c, and defined for the specific purpose of this proof as follows:

$$\begin{aligned}
\mathrm{c}(a) &= 1 & \mathrm{c}(\varepsilon) &= 0 \\
\mathrm{c}(\neg\varphi) &= 1 + \mathrm{c}(\varphi) & \mathrm{c}(\downarrow) &= 1 \\
\mathrm{c}(\varphi \wedge \psi) &= 1 + \mathrm{c}(\varphi) + \mathrm{c}(\psi) & \mathrm{c}(\alpha\beta) &= \mathrm{c}(\alpha) + \mathrm{c}(\beta) \\
\mathrm{c}(\langle \alpha = \beta \rangle) &= 1 + \mathrm{c}(\alpha) + \mathrm{c}(\beta) & \mathrm{c}(\alpha \cup \beta) &= 1 + \mathrm{c}(\alpha) + \mathrm{c}(\beta) \\
& & \mathrm{c}([\varphi]) &= 2 + \mathrm{c}(\varphi)
\end{aligned}$$

Observe that the only node expressions of least complexity (namely, 1) are the labels a or $\langle \varepsilon = \varepsilon \rangle$, that the only path expressions of least complexity (namely, 0) are those of the form $\varepsilon \ldots \varepsilon$, and that the only path expressions of complexity 1 consist of one $\downarrow$ symbol concatenated with any number of ε symbols at both sides (that number might be 0, leaving the path expression $\downarrow$). Observe also that $\mathrm{c}(\varphi \wedge \langle \alpha = \beta \rangle) < \mathrm{c}(\langle [\varphi]\alpha = \beta \rangle)$.

Base case. If the complexity of F is 0 then it is the path expression $\varepsilon \dots \varepsilon$, which is provably equivalent to ε by IsAx5. Since $\varepsilon \in P_0^-$, b) is immediate. If the complexity of F is 1 then F is either a node expression which consists of a single label or $\langle \varepsilon = \varepsilon \rangle$, or F is the path expression $\downarrow$ (eventually concatenated with ε but those path expressions are all provably equivalent to $\downarrow$ by EqAx6). If $F = a$ ($a \in \mathbb{A}$), then a) is immediate, since using EqAx6 and Boolean reasoning we have $\mathsf{XP}^- \vdash F \equiv a \wedge \langle \varepsilon = \varepsilon \rangle$, so $a \wedge \langle \varepsilon = \varepsilon \rangle \in N_0^-$, and we finish by applying Lemma 99. If $F = \langle \varepsilon = \varepsilon \rangle$, then a) follows from EqAx6, LbAx1, Boolean reasoning and Lemma 99. If $F = \downarrow$, by IsAx5, LbAx1 and PrAx2 we have $\mathsf{XP}^- \vdash F \equiv \downarrow[\bigvee_{a \in \mathbb{A}} a]\varepsilon$, and by PrAx3 and IsAx6, we conclude $\mathsf{XP}^- \vdash F \equiv \bigcup_{a \in \mathbb{A}} \downarrow[a]\varepsilon$ (observe that $\downarrow[a]\varepsilon \equiv \downarrow[a \wedge \langle \varepsilon = \varepsilon \rangle]\varepsilon \in P_1^-$).

Induction. If the complexity of F is greater than 1, then F involves some of the operators $\neg, \wedge, \langle\ \rangle, \cup, [\ ]$ or a concatenation different from the ones of complexity 1 mentioned above. We will perform the inductive step for each of these operators.

If $F = \varphi \wedge \psi$ or $\neg\varphi$, we reason as in the propositional case. If $F = \varphi \wedge \psi$, we use the inductive hypothesis on φ and ψ to obtain that $\mathsf{XP}^- \vdash F \equiv \bigvee_i \varphi_i \wedge \bigvee_j \psi_j$, where $\varphi_i \in N^-_{\mathsf{dd}(\varphi)}$ for all i and $\psi_j \in N^-_{\mathsf{dd}(\psi)}$ for all j. Actually, we can assume that $\varphi_i, \psi_j \in N_n^-$ for all i, j by Lemma 99. We now use Boolean distributive laws to prove that F is equivalent to $\bigvee_{i,j}(\varphi_i \wedge \psi_j)$. We then use Lemma 96 plus the consistency of F to remove from that expression redundant conjunctions (if $\varphi_i = \psi_j$, from $\varphi_i \wedge \psi_j$ we just keep φ_i) and inconsistent conjunctions (cases where $\varphi_i \neq \psi_j$).

If $F = \neg\varphi$, we have by inductive hypothesis that $\mathsf{XP}^- \vdash \neg\varphi \equiv \neg \bigvee_{1 \leq i \leq m} \varphi_i$, and we can again assume by Lemma 99 that $\varphi_i \in N_n^-$ for all i. Expanding each φ_i into $a_i \wedge \bigwedge_{\rho \in C_i} \rho \wedge \bigwedge_{\rho \in D_n^- \setminus C_i} \neg\rho$ (where $C_i \subseteq D_n^-$) and then using Boolean algebra, we have $\mathsf{XP}^- \vdash \neg\varphi \equiv \bigwedge_{1 \leq i \leq m}(\neg a_i \vee \bigvee_{\rho \in C_i} \neg\rho \vee \bigvee_{\rho \in D_n^- \setminus C_i} \rho)$. We now use Boolean distributive laws to get $\mathsf{XP}^- \vdash \neg\varphi \equiv \bigvee_{\omega \in \Omega} \bigwedge_{1 \leq i \leq m} \omega(i)$, where each $\omega(i)$ is either $\neg a_i$, some $\neg\rho$ for $\rho \in C_i$, or some $\rho \in D_n^- \setminus C_i$, and where Ω contains all possible strings ω of length m formed in that way. We now use LbAx1 to get $\mathsf{XP}^- \vdash \neg\varphi \equiv \bigvee_{\omega \in \Omega} \bigvee_{a \in \mathbb{A}} a \wedge \bigwedge_{1 \leq i \leq m} \omega(i)$. Then, we eliminate repetitions in conjunctions of node expressions, and use properties of Boolean algebra to eliminate inconsistencies; also, as each disjunct has some positive occurrences of some $a \in \mathbb{A}$, we can use LbAx2 and eliminate the (redundant) occurrences of negation of labels. So now we have that $\mathsf{XP}^- \vdash \neg\varphi \equiv \bigvee_{\omega \in \Omega} \bigvee_{a \in \mathbb{A}} \psi_{\omega,a}$, where each $\psi_{\omega,a}$ is of the form $a \wedge \bigwedge_{\rho \in C} \rho \wedge \bigwedge_{\rho \in D} \neg\rho$, with $C, D \subseteq D_n^-$ and $C \cap D = \emptyset$. However we do not necessarily have $D = D_n^- \setminus C$, so these conjunctions may not add up to be of the form of a node expression in N_n^-: to add the conjunctions needed in order to get normal forms, we proceed as in the proof of Lemma 98 to complete each $a \wedge \bigwedge_{\rho \in C} \rho \wedge \bigwedge_{\rho \in D} \neg\rho$ into $\bigvee_j (a \wedge \bigwedge_{\rho \in C_j} \wedge \bigwedge_{\rho \in D_n^- \setminus C_j} \neg\rho)$, where $C \subseteq C_j$ for all j. Finally, we have obtained a set $(C_k)_{k \in K}$ of subsets of D_n^- such that $\mathsf{XP}^- \vdash \neg\varphi \equiv \bigvee_{k \in K}(a_k \wedge \bigwedge_{\rho \in C_k} \rho \wedge \bigwedge_{\rho \in D_n^- \setminus C_k} \neg\rho)$.

If F is of the form $\langle \alpha = \beta \rangle$, we reason as follows. Since $c(\alpha) < c(\langle \alpha = \beta \rangle)$, by inductive hypothesis, we have $\mathsf{XP}^- \vdash \alpha \equiv \bigcup_i [\varphi_i]\alpha_i$ for some $\alpha_i \in P_n^-$ and $\varphi_i \in N_n^-$ (We may have to use also Lemma 99, PrAx3, IsAx6, and **Der21** of Fact 90 if $\mathsf{dd}(\alpha) < \mathsf{dd}(\langle \alpha = \beta \rangle)$). Similarly, we can turn β into $\bigcup_j [\psi_j]\beta_j$. Using EqAx2, EqAx3, and EqAx1, we obtain

$\mathsf{XP}^- \vdash F \equiv \bigvee_{i,j} \varphi_i \wedge \psi_j \wedge \langle \alpha_i = \beta_j \rangle$. We then use LbAx1 and Boolean reasoning to get $\mathsf{XP}^- \vdash F \equiv \bigvee_{i,j} \varphi_i \wedge \psi_j \wedge (\bigvee_{a \in \mathbb{A}} a \wedge \langle \alpha_i = \beta_j \rangle)$, and then distribute the $\wedge$, use Lemma 98 over each $a \wedge \langle \alpha_i = \beta_i \rangle$, and eliminate inconsistencies using Lemma 96 to obtain our desired result.

Suppose that F is a path expression. Without loss of generality, we can assume that $F \neq [\varphi]$ or $F \neq \alpha \cup \beta$ because in those cases, there exist an equivalent expression of the same complexity that is a concatenation ($[\varphi]\varepsilon$ or $(\alpha \cup \beta)\varepsilon$ respectively). Then we only need to prove the result for the concatenation in order to conclude the proof. Also without loss of generality we may assume that F does not start with ε, since in that case there exist an equivalent expression of the same complexity that doesn't start with ε. In case F is a concatenation $F = \alpha\beta$ that doesn't start with ε, we split the proof in three different cases according to the form of α (note that, by IsAx4, we can assume that α is not a concatenation itself).

If F is of the form $[\varphi]\beta$ then by IsAx5 we may suppose that β ends in ε and by **Der21** of Fact 90 we may suppose that β is either ε or starts with $\downarrow$ (notice that ε does not count in the complexity of a formula and that the expression in the left hand side of **Der21** of Fact 90 has a complexity greater than the one in the right hand side). By inductive hypothesis, φ is provably equivalent to $\bigvee_i \varphi_i$ for some $(\varphi_i)_i \in N_n^-$ (We may have to use Lemma 99 to increase the degree). Therefore, by PrAx3, $[\varphi]$ is provably equivalent to $\bigcup_i [\varphi_i]$. By inductive hypothesis, β is provably equivalent to $\bigcup_j \beta_j$ for some $(\beta_j)_j \in P_n^-$ (again, we may have to use Lemma 99). Hence F is provably equivalent to $(\bigcup_i [\varphi_i])(\bigcup_j \beta_j)$, and by IsAx6 we conclude that F is provably equivalent to $\bigcup_{i,j} [\varphi_i]\beta_j$ as we wanted to show.

If F is of the form $\downarrow\beta$, we use inductive hypothesis to show that β is provably equivalent to $\bigcup_i [\varphi_i]\beta_i$ for some $(\beta_i)_i \in P_{n-1}^-$ and $(\varphi_i)_i \in N_{n-1}^-$. By IsAx6, we conclude that F is provably equivalent to $\bigcup_i \downarrow[\varphi_i]\beta_i$, and $\downarrow[\varphi_i]\beta_i \in P_n^-$ as we wanted to show.

Finally, if F is of the form $(\gamma \cup \delta)\beta$, then, by IsAx6, $F \equiv (\gamma\beta) \cup (\delta\beta)$. The result follows from inductive hypothesis for $\gamma\beta$ and $\delta\beta$ (as usual, we may have to use Lemma 99, PrAx3, IsAx6, and **Der21** of Fact 90 to increase the degree). □

2.3.3 Completeness for node and path expressions

In this subsection we show that for node expressions φ and ψ of $\mathrm{XPath}_=(\downarrow)^-$, the equivalence $\varphi \equiv \psi$ is derivable from the axiom schemes of Table 2.1 if and only if φ is $\mathrm{XPath}_=(\downarrow)^-$-semantically equivalent to ψ. We also show the same result for path expressions of $\mathrm{XPath}_=(\downarrow)^-$.

We first introduce the main lemma of this section, and then continue to its consequences; as the proof of this lemma is very extensive, we postpone until we have proven Theorem 103.

Lemma 102. *Any node expression $\varphi \in N_n^-$ is satisfiable.*

Based on the above lemma, we arrive to the next theorem, which is the main result of this section:

Theorem 103 (Completeness of $\mathrm{XPath}_=(\downarrow)^-$)**.**

1. *Let φ and ψ be node expressions of* XPath$_=(\downarrow)^-$. *Then* $\mathsf{XP}^- \vdash \varphi \equiv \psi$ *iff* $\models \varphi \equiv \psi$.
2. *Let α and β be path expressions of* XPath$_=(\downarrow)^-$. *Then* $\mathsf{XP}^- \vdash \alpha \equiv \beta$ *iff* $\models \alpha \equiv \beta$.

Proof. Let us show item 1. Soundness follows from Proposition 91.

For completeness, suppose $\models \varphi \equiv \psi$. Now assume that φ is consistent and ψ is not. On the one hand, by Theorem 101, there is n such that φ is provably equivalent to $\bigvee_{1\leq i\leq k} \varphi_i$, for $\varphi_i \in N_n^-$. By Lemma 102, to be proved next, we have that in particular φ_1 (and hence φ) is satisfiable. On the other hand, by Proposition 91, ψ is unsatisfiable, and this contradicts the fact that $\models \varphi \equiv \psi$. This shows that if φ is consistent then so is ψ. Symmetrically, one can show that if ψ is consistent, then so is φ. Therefore, either φ and ψ are consistent or φ and ψ are inconsistent. In the latter case, we trivially have $\mathsf{XP}^- \vdash \varphi \equiv \psi$.

In case φ and ψ are consistent, by Theorem 101 and Lemma 99, there is n and node expressions φ' and ψ' which are disjunctions of node expressions in N_n^- such that $\mathsf{XP}^- \vdash \varphi \equiv \varphi'$ and $\mathsf{XP}^- \vdash \psi \equiv \psi'$.

Suppose that φ' contains a disjunct φ'' which is not a disjunct of ψ'. By Lemma 102, φ'' is satisfiable in some data tree $\mathcal{T}$. By Lemma 96, for any disjunct ψ'' of ψ' we have that $\varphi'' \wedge \psi''$ is inconsistent, and by Proposition 91, unsatisfiable. Hence ψ' is not satisfiable in $\mathcal{T}$, and so $\not\models \varphi \equiv \psi$, a contradiction. The case when ψ' contains a disjunct which is not a disjunct of φ' is analogous.

Then φ' and ψ' are identical, modulo reordering of disjunctions, and so $\mathsf{XP}^- \vdash \varphi' \equiv \psi'$ which implies $\mathsf{XP}^- \vdash \varphi \equiv \psi$.

For item 2, soundness follows from Proposition 91. For completeness, suppose $\models \alpha \equiv \beta$.

Suppose that α is consistent and β is not. On the one hand, by Theorem 101, there is n such that α is provably equivalent to $\bigcup_{1\leq i\leq k}[\varphi_i]\alpha_i$, with $\alpha_i \in P_n^-$ and $\varphi_i \in N_n^-$. Furthermore, we can assume that $[\varphi_1]\alpha_1$ is consistent (if it is not, we simply remove it from the disjunction) and so $\langle \alpha_1 = \alpha_1 \rangle$ is a conjunct of φ_1 by Lemma 95. By Lemma 102, the node expression φ_1 is satisfiable. Then, since $\langle \alpha_1 = \alpha_1 \rangle$ is a conjunct of φ_1, the path expression $[\varphi_1]\alpha_1$ is satisfiable, and so α is satisfiable. On the other hand, by Proposition 91, β is unsatisfiable, and this contradicts the fact that $\models \alpha \equiv \beta$. This shows that if α is consistent then so is β. Symmetrically, one can show that if β is consistent, then so is α. Therefore, either α and β are consistent or α and β are inconsistent. In the latter case, we trivially have $\mathsf{XP}^- \vdash \alpha \equiv \beta$.

Suppose both α and β are consistent. By Theorem 101 plus Lemma 99 we have that there is n and path expressions $\alpha_1 \ldots \alpha_k, \beta_1 \ldots \beta_\ell$ in P_n^- and node expressions $\varphi_1 \ldots \varphi_k$, $\psi_1 \ldots \psi_\ell \in N_n^-$ such that $\mathsf{XP}^- \vdash \alpha \equiv \bigcup_{1\leq i\leq k}[\varphi_i]\alpha_i$ and $\mathsf{XP}^- \vdash \beta \equiv \bigcup_{1\leq j\leq \ell}[\psi_j]\beta_j$. Furthermore, we can assume that $\langle \alpha_i = \alpha_i \rangle$ is a conjunct of φ_i for $i = 1 \ldots k$ and $\langle \beta_j = \beta_j \rangle$ is a conjunct of ψ_j for $j = 1 \ldots \ell$.

Now, suppose that

$$[\varphi_i]\alpha_i \notin \{[\psi_1]\beta_1, \ldots, [\psi_\ell]\beta_\ell\} \tag{18}$$

for some i. Since $\varphi_i \in N_n^-$, by Lemma 102, there is a data tree $\mathcal{T} = (T, \pi)$ with root r such that $\mathcal{T}, r \models \varphi_i$. Since $\langle \alpha_i = \alpha_i \rangle$ is a conjunct of φ_i, we have that there is $y \in T$ such that $\mathcal{T}, r, y \models \alpha_i$.

Let us show that $\mathcal{T}, r, y \not\models [\psi_j]\beta_j$ for any $j \leq \ell$. Fix any j. By (18), we have that $\varphi_i \neq \psi_j$ or $\alpha_i \neq \beta_j$. In the first case, $\mathcal{T}, r, y \not\models [\psi_j]\beta_j$ follows from Lemma 96 and Proposition 91 (in particular $\mathcal{T}, r \not\models \psi_j$). If $\varphi_i = \psi_j$, we have $\alpha_i \neq \beta_j$ and $\mathcal{T}, r, y \not\models [\psi_j]\beta_j$ follows from Lemma 97.

So we have that $\mathcal{T}, r, y \models \alpha$ but $\mathcal{T}, r, y \not\models \beta$, a contradiction with our hypothesis that $\models \alpha \equiv \beta$. Hence for any i there is j such that $[\varphi_i]\alpha_i = [\psi_j]\beta_j$. Analogously one can show that for any j there is i such that $[\psi_j]\beta_j = [\varphi_i]\alpha_i$. Then $\bigcup_{1\leq i\leq k}[\varphi_i]\alpha_i$ and $\bigcup_{1\leq j\leq \ell}[\psi_j]\beta_j$ are identical, modulo reordering of unions, and so $\mathsf{XP}^- \vdash \alpha \equiv \beta$. □

All we need to complete the argument is to prove Lemma 102. Doing this involves the rest of the section.

Canonical model

In order to prove Lemma 102, we construct, recursively in n and for every $\varphi \in N_n^-$, a data tree $\mathcal{T}^\varphi = (T^\varphi, \pi^\varphi)$ such that φ is satisfiable in $\mathcal{T}^\varphi$.

For the base case, if $\varphi \in N_0^-$ and $\varphi = a \wedge \langle \varepsilon = \varepsilon \rangle$ with $a \in \mathbb{A}$, simply define the data tree $\mathcal{T}^\varphi = (T^\varphi, \pi^\varphi)$ where T^φ is a tree which consists of the single node x with label a, and $\pi^\varphi = \{\{x\}\}$.

Now let $\varphi \in N_{n+1}^-$. Since φ is a conjunction as in Definition 92, it is enough to guarantee that the following conditions hold (we now observe that these conditions are enough because of EqAx1, but we usually avoid these observations of symmetry):

(C1) If $a \in \mathbb{A}$ is a conjunct of φ, then the root r^φ of $\mathcal{T}^\varphi$ has label a.

(C2) If $\langle \varepsilon = \downarrow[\psi]\alpha \rangle$ is a conjunct of φ, then there is a child $r^{\mathbf{v}}$ (where $\mathbf{v} = (\psi, \alpha)$; we will introduce this notation in time to formalize the construction) of the root r^φ of $\mathcal{T}^\varphi$ at which ψ is satisfied and a node $x^{\mathbf{v}}$ with the same data value as r^φ such that $\mathcal{T}^\varphi, r^{\mathbf{v}}, x^{\mathbf{v}} \models \alpha$.

(C3) If $\langle \downarrow[\psi]\alpha = \downarrow[\rho]\beta \rangle$ is a conjunct of φ, then there are two children $r_1^{\mathbf{u}}$, $r_2^{\mathbf{u}}$ of the root r^φ of $\mathcal{T}^\varphi$ at which ψ and ρ are satisfied respectively, and there are nodes $x^{\mathbf{u}}$ and $y^{\mathbf{u}}$ with the same data value such that $\mathcal{T}^\varphi, r_1^{\mathbf{u}}, x^{\mathbf{u}} \models \alpha$ and $\mathcal{T}^\varphi, r_2^{\mathbf{u}}, y^{\mathbf{u}} \models \beta$.

(C4) If $\neg\langle \varepsilon = \downarrow[\psi]\alpha \rangle$ is a conjunct of φ, then for each child z of the root r^φ of $\mathcal{T}^\varphi$ at which ψ is satisfied, if x is a node such that $\mathcal{T}^\varphi, z, x \models \alpha$, then the data value of x is different than the one of r^φ.

(C5) If $\neg\langle \downarrow[\psi]\alpha = \downarrow[\rho]\beta \rangle$ is a conjunct of φ, then for each children z_1, z_2 of the root r^φ of $\mathcal{T}^\varphi$ at which ψ and ρ are satisfied respectively, if w_1, w_2 are nodes such that $\mathcal{T}^\varphi, z_1, w_1 \models \alpha$ and $\mathcal{T}^\varphi, z_2, w_2 \models \beta$, then the data values of w_1 and w_2 are different.

Since the construction of the canonical model requires some technical notation that might hinder the understanding of the ideas behind it, we will begin with an intuitive description of the construction.

Insight into the construction

The idea to achieve all the previous conditions is to incrementally build a tree such that it satisfies at its root conditions (C1), (C4), and (C5), then also (C2) (without spoiling any previous conditions), and finally also (C3).

First we start with a root r^φ labeled a, where a is the label present in φ (so that condition (C1) is satisfied). At this point in the construction, as we only have one node, conditions (C4) and (C5) are trivially satisfied. On the contrary, (C2) and (C3) might not be satisfied, and require a positive action (i.e. changing the current model) to make them true. We want to add witnesses that guarantee the satisfaction of (C2) and (C3), and we will achieve this by the use of the inductive hypothesis to construct new trees that we will hang as children of r^φ. However, adding witnesses jeopardizes the satisfaction of (C4) and (C5), so we need to do it carefully enough.

First we add witnesses in order to satisfy condition (C2). If $\psi \in N_n^-$, by inductive hypothesis, there exists a tree $\mathcal{T}^\psi$ such that ψ is satisfied at $\mathcal{T}^\psi$. Also, if $\langle \varepsilon = \downarrow[\psi]\alpha\rangle$ is a conjunct of φ, by the consistency of φ, Lemma 95 and the inductive hypothesis, there is a pair of nodes satisfying α in $\mathcal{T}^\psi$ and starting at its root. In this case, we will hang a copy of $\mathcal{T}^\psi$ (or perhaps a slightly modified copy of it constructed in order not to spoil condition (C4)) as a child of r^φ and merge the equivalence class of r^φ to the equivalence class of the endpoint $x^{\mathbf{v}}$ of a specially chosen pair of nodes satisfying α and beginning at the root of $\mathcal{T}^\psi$ (see Figure 22(a)). This is the only merging required; other classes in $\mathcal{T}^\psi$ remain disjoint

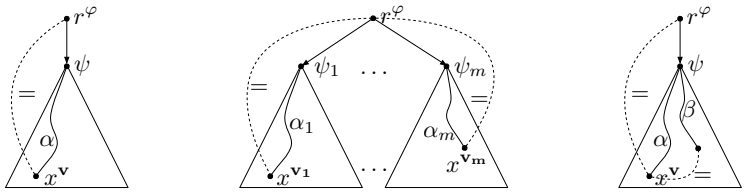


Figure 22: (a) A witness for $\langle \varepsilon = \downarrow[\psi]\alpha\rangle$; (b) we repeat the process of (a) for each conjunct $\langle \varepsilon = \downarrow[\psi_1]\alpha_1\rangle, \ldots, \langle \varepsilon = \downarrow[\psi_m]\alpha_m\rangle$ of φ; (c) by adding a witness for $\langle \varepsilon = \downarrow[\psi]\alpha\rangle$, we may be creating an unwanted witness for $\langle \varepsilon = \downarrow[\psi]\beta\rangle$.

from the previous constructed part of $\mathcal{T}^\varphi$. In this way, we will guarantee condition (C2) (see Figure 22(b)). Since the other equivalence classes of $\mathcal{T}^\psi$ will remain disjoint from the rest of the tree $\mathcal{T}^\varphi$ all along the construction and since two different normal forms cannot be satisfied at the same point (see Lemma 96 plus Proposition 91), the only way in which this process could spoil condition (C4) is that there is $\beta \in P_n^-$ such that $\neg\langle \varepsilon = \downarrow[\psi]\beta\rangle$ is a conjunct of φ and a pair of nodes satisfying β in $\mathcal{T}^\psi$, starting at its root and ending in a point with the same data value as $x^{\mathbf{v}}$ (see Figure 22(c)). But Lemma 104 ensures that (maybe with changes to $\mathcal{T}^\psi$) we can choose $x^{\mathbf{v}}$ to avoid this situation. Then, since we only add nodes to the equivalence class of the root r^φ by this process, the only way

in which we could spoil condition (C5) is if φ has conjuncts $\langle \varepsilon = \downarrow[\psi]\mu \rangle$, $\langle \varepsilon = \downarrow[\rho]\delta \rangle$ and $\neg\langle \downarrow[\psi]\mu = \downarrow[\rho]\delta \rangle$ for some $\psi, \rho \in N_n^-$, $\mu, \delta \in P_n^-$. But this is clearly unsatisfiable and so our axioms should tell us that it is inconsistent (see EqAx7).

We now proceed to add witnesses in order to satisfy condition (C3). By an argument similar to the one given for condition (C2), if $\langle \downarrow[\psi]\alpha = \downarrow[\rho]\beta \rangle$ is a conjunct of φ, there are, by inductive hypothesis, trees $\mathcal{T}^\psi$ and $\mathcal{T}^\rho$ at which ψ and ρ are satisfied and pairs of nodes satisfying α and β starting at their respective roots. We will hang a copy of each of those trees (or perhaps slightly modified copies of them) as children of r^φ and we will merge the equivalence classes (in $\mathcal{T}^\psi$ and $\mathcal{T}^\rho$) of the ending points $x^{\mathbf{u}}, y^{\mathbf{u}}$ of a specially chosen pair of nodes satisfying α (and starting at the root of $\mathcal{T}^\psi$) and a specially chosen pair of nodes satisfying β (and starting at the root of $\mathcal{T}^\rho$) as mentioned before (see Figure 23(a)). Note

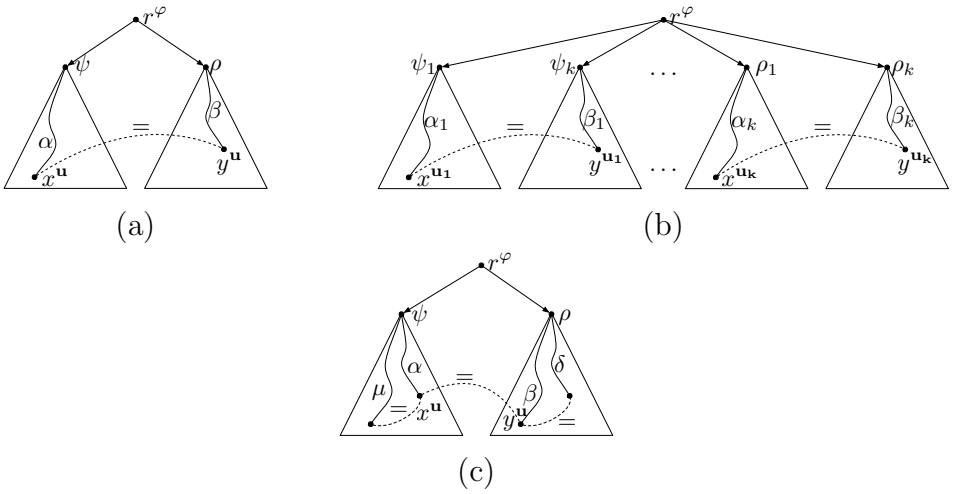


Figure 23: (a) A witness for $\langle \downarrow[\psi]\alpha = \downarrow[\rho]\beta \rangle$; (b) we repeat the process of (a) for each conjunct $\langle \downarrow[\psi_1]\alpha_1 = \downarrow[\rho_1]\beta_1 \rangle, \ldots, \langle \downarrow[\psi_k]\alpha_k = \downarrow[\rho_k]\beta_k \rangle$ of φ; (c) by adding a witness for $\langle \downarrow[\psi]\alpha = \downarrow[\rho]\beta \rangle$, we may be creating an unwanted witness for $\langle \downarrow[\psi]\mu = \downarrow[\rho]\delta \rangle$.

that all the classes in $\mathcal{T}^\psi$ and $\mathcal{T}^\rho$ remain disjoint from the previous constructed part of $\mathcal{T}^\varphi$. In this way, we guarantee condition (C3) (see Figure 23(b)). Since we are not adding any nodes to the class of r^φ, it is clear that we cannot spoil condition (C2) by performing this procedure. With a similar argument as the one given before, the only way in which we can spoil condition (C5) is that there are $\psi, \rho \in N_n^-$, $\alpha, \beta, \mu, \delta \in P_n^-$ such that $\langle \downarrow[\psi]\alpha = \downarrow[\rho]\beta \rangle$ and $\neg\langle \downarrow[\psi]\mu = \downarrow[\rho]\delta \rangle$ are conjuncts of φ, a pair of nodes satisfying μ beginning at the root of $\mathcal{T}^\psi$ and ending in a point with the same data value as $x^{\mathbf{u}}$, and a pair of nodes satisfying δ beginning at the root of $\mathcal{T}^\rho$ and ending in a point with the same data value as $y^{\mathbf{u}}$ (see Figure 23(c)). But Lemma 104 ensures that we can choose $x^{\mathbf{u}}$ and $y^{\mathbf{u}}$ to avoid this situation.

Formalization

In order to formalize the construction described above, we introduce the following key lemma:

Lemma 104. *Let $\psi_0 \in N_n^-$, $\alpha, \beta_1, \ldots, \beta_m \in P_n^-$. Suppose that there exists a tree $\mathcal{T}^{\psi_0} = (T^{\psi_0}, \pi^{\psi_0})$ with root r^{ψ_0} such that $\mathcal{T}^{\psi_0}, r^{\psi_0} \models \psi_0$ and for all $i = 1, \ldots, m$ there exists $\gamma_i \in P_{n+1}^-$ such that $\langle \gamma_i = \downarrow[\psi_0]\alpha\rangle \wedge \neg\langle \gamma_i = \downarrow[\psi_0]\beta_i\rangle$ is consistent. Then there exists a tree $\widetilde{\mathcal{T}^{\psi_0}} = (\widetilde{T^{\psi_0}}, \widetilde{\pi^{\psi_0}})$ with root $\widetilde{r^{\psi_0}}$ and a node x such that:*

- $\widetilde{\mathcal{T}^{\psi_0}}, \widetilde{r^{\psi_0}} \models \psi_0$,
- $\widetilde{\mathcal{T}^{\psi_0}}, \widetilde{r^{\psi_0}}, x \models \alpha$, *and*
- $[x]_{\widetilde{\pi^{\psi_0}}} \neq [y]_{\widetilde{\pi^{\psi_0}}}$ *for all y such that $\widetilde{\mathcal{T}^{\psi_0}}, \widetilde{r^{\psi_0}}, y \models \beta_i$ for some $i = 1, \ldots, m$.*

Proof. Suppose that $\alpha = \downarrow[\psi_1] \ldots \downarrow[\psi_{j_0}]\varepsilon$, where $\psi_k \in N_{n-k}^-$ for all $k = 1, \ldots, j_0$. If $j_0 = 0$ (that is, $\alpha = \varepsilon$), then it suffices to take $\widetilde{\mathcal{T}^{\psi_0}} = \mathcal{T}^{\psi_0}$ and $x = r^{\psi_0}$. We only need to show that then $\neg\langle \varepsilon = \beta_i\rangle$ is a conjunct of ψ_0 for all i. Indeed, assuming instead that $\langle \varepsilon = \beta_i\rangle$ is a conjunct of ψ_0 for some i, we have

$$\begin{aligned}
\langle \gamma_i = \downarrow[\psi_0]\alpha\rangle \wedge \neg\langle \gamma_i = \downarrow[\psi_0]\beta_i\rangle &\equiv \langle \gamma_i = \downarrow[\psi_0 \wedge \langle \varepsilon = \beta_i\rangle]\rangle \wedge \neg\langle \gamma_i = \downarrow[\psi_0]\beta_i\rangle \\
&\qquad \text{(Hypothesis } \langle \varepsilon = \beta_i\rangle \text{ is a conjunct of } \psi_0) \\
&\leq \langle \gamma_i = \downarrow[\psi_0]\beta_i\rangle \wedge \neg\langle \gamma_i = \downarrow[\psi_0]\beta_i\rangle \\
&\qquad \text{(\textbf{Der21} (Fact 90) \& EqAx8)} \\
&\equiv \text{FALSE} \qquad \text{(Boolean)}
\end{aligned}$$

which is a contradiction with our assumption that $\langle \gamma_i = \downarrow[\psi_0]\alpha\rangle \wedge \neg\langle \gamma_i = \downarrow[\psi_0]\beta_i\rangle$ is consistent, by standard propositional reasoning.

If $j_0 > 0$, to define $\widetilde{\mathcal{T}^{\psi_0}} = (\widetilde{T^{\psi_0}}, \widetilde{\pi^{\psi_0}})$ we modify the tree $\mathcal{T}^{\psi_0} = (T^{\psi_0}, \pi^{\psi_0})$. From the consistency of $\langle \gamma_i = \downarrow[\psi_0]\alpha\rangle$ for some i, by Lemma 95, we conclude that $\langle \alpha = \alpha\rangle$ is a conjunct of ψ_0. Hence there is $z \in T^{\psi_0}$, $z \neq r^{\psi_0}$, such that $\mathcal{T}^{\psi_0}, r^{\psi_0}, z \models \alpha$.

Before proceeding to complete the proof of this case, we give a sketch of it. We prove that we cannot have a witness for β_i with the same data value as z in the subtree $T^{\psi_0}|_z$. Intuitively this is because, in that case, α would be a prefix of β_i, say $\beta_i = \alpha\delta$, and $\langle \varepsilon = \delta\rangle$ would be a conjunct of ψ_{j_0}. Then $\langle \gamma_i = \downarrow[\psi_0]\alpha\rangle \wedge \neg\langle \gamma_i = \downarrow[\psi_0]\beta_i\rangle$ would be unsatisfiable (and thus it should be inconsistent) for any choice of γ_i which is a contradiction. But our hypotheses are not enough to avoid having a witness for β_i in the class of z outside $T^{\psi_0}|_z$; thus we need to change the tree in order to achieve the desired properties. We replicate the subtree $T^{\psi_0}|_z$ but using fresh data values (different from any other data value already present in $\mathcal{T}^{\psi_0}$), see Figure 24. It is clear that in this way, the second and the third conditions will be satisfied by the root of this new subtree. The first condition will also remain true because the positive conjuncts will remain valid since we are not suppressing

any nodes, and the negative ones will not be affected either because every node we add has a fresh data value.

Now we formalize the previous intuition. Call p the parent node of the aforementioned $z \in T^{\psi_0}$ and define $\widetilde{T^{\psi_0}}$ as $T^{\psi_0} \sqcup T^x$, where we define T^x as $T^{\psi_0}|_z$, and in $\widetilde{T^{\psi_0}}$ the root of T^x is a child x of p. Define $\widetilde{\pi^{\psi_0}}$ as $\pi^{\psi_0} \sqcup \pi^z$; observe that the data values of T^x differ from all those of the rest of $\widetilde{T^{\psi_0}}$ (see Figure 24).

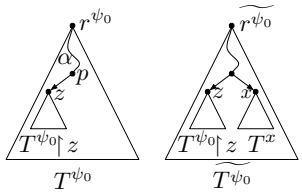


Figure 24: $T^x = T^{\psi_0}|_z$ is a new subtree with disjoint data values to the rest of $\widetilde{\mathcal{T}^{\psi_0}}$. The new node x satisfies $\widetilde{\mathcal{T}^{\psi_0}}, \widetilde{r^{\psi_0}}, x \models \alpha$.

We now check that this new tree $\widetilde{T^{\psi_0}}$ satisfies ψ_0 at its root $\widetilde{r^{\psi_0}}$. We prove by induction that x_j, the j-th ancestor of x (namely $x_j \xrightarrow{j} x$, and we let $x_0 := x$), satisfies $\widetilde{\mathcal{T}^{\psi_0}}, x_j \models \psi_{j_0-j}$. This proves both that $\widetilde{\mathcal{T}^{\psi_0}}, \widetilde{r^{\psi_0}} \models \psi_0$ and that $\widetilde{\mathcal{T}^{\psi_0}}, \widetilde{r^{\psi_0}}, x \models \alpha$. For the base case $j = 0$, the result is straightforward from Proposition 88: T^x is a copy of $T^{\psi_0}|_z$, with z satisfying ψ_{j_0}. For the inductive case, assume that the result holds for $x_0, \ldots, x_j$. We want to see that it holds for x_{j+1}. To do this, we verify that every conjunct of ψ_{j_0-j-1} is satisfied at x_{j+1}:

- If the conjunct is a label, it is clear that x_{j+1} still has that label in $\widetilde{\mathcal{T}^{\psi_0}}$, as it has not been changed by the construction.
- If the conjunct is of the form $\langle \mu_1 = \mu_2 \rangle$, then it must still hold in $\widetilde{\mathcal{T}^{\psi_0}}$ by inductive hypothesis plus the fact that our construction did not remove nodes.
- If the conjunct is of the form $\neg\langle \mu_1 = \mu_2 \rangle$, we observe that, by inductive hypothesis plus the fact that the data classes of nodes in T^x are disjoint with those of the rest of $\widetilde{\mathcal{T}^{\psi_0}}$, then $\langle \mu_1 = \mu_2 \rangle$ can only be true in x_{j+1} if there are witnesses y_1, y_2 in the same equivalence class in the new subtree T^x such that $\widetilde{\mathcal{T}^{\psi_0}}, x_{j+1}, y_1 \models \mu_1$ and $\widetilde{\mathcal{T}^{\psi_0}}, x_{j+1}, y_2 \models \mu_2$. In that case, we have that

  $$\mu_1 = \downarrow[\psi_{j_0-j}]\downarrow \ldots \downarrow[\psi_{j_0}]\hat{\mu}_1 \quad \text{and} \quad \mu_2 = \downarrow[\psi_{j_0-j}]\downarrow \ldots \downarrow[\psi_{j_0}]\hat{\mu}_2$$

  for some $\hat{\mu}_1, \hat{\mu}_2$, and that $\widetilde{\mathcal{T}^{\psi_0}}, x_0, y_1 \models \hat{\mu}_1$, $\widetilde{\mathcal{T}^{\psi_0}}, x_0, y_2 \models \hat{\mu}_2$. Therefore, by Lemma 94, $\langle \hat{\mu}_1 = \hat{\mu}_2 \rangle$ is a conjunct of ψ_{j_0}, and then $\mathcal{T}^{\psi_0}, z \models \langle \hat{\mu}_1 = \hat{\mu}_2 \rangle$, a contradiction with our assumption that $\neg\langle \downarrow[\psi_{j_0-j}]\downarrow \ldots \downarrow[\psi_{j_0}]\hat{\mu}_1 = \downarrow[\psi_{j_0-j}]\downarrow \ldots \downarrow[\psi_{j_0}]\hat{\mu}_2 \rangle$ is a conjunct of $\psi_{j_0-(j+1)}$ and $\mathcal{T}^{\psi_0}, x_{j+1} \models \psi_{j_0-(j+1)}$.

To conclude the proof, we only need to check that $[x]_{\widetilde{\pi^{\psi_0}}} \neq [y]_{\widetilde{\pi^{\psi_0}}}$ for all y such that $\widetilde{\mathcal{T}^{\psi_0}}, \widetilde{r^{\psi_0}}, y \models \beta_i$ for some $i = 1, \ldots, m$. Suppose that $\beta_i = \downarrow[\rho_1] \ldots \downarrow[\rho_{l_0}]\varepsilon$. If $l_0 < j_0$ or $\rho_l \neq \psi_l$ for some $l = 1, \ldots, j_0$, then the result follows immediately from the construction. Otherwise, $l_0 \geq j_0$ and $\rho_l = \psi_l$ for all $l = 1, \ldots, j_0$ and so, by hypothesis, there exists $\gamma_i \in P^-_{n+1}$ such that

$$\langle \gamma_i = \downarrow[\psi_0]\downarrow[\psi_1]\downarrow \ldots \downarrow[\psi_{j_0}]\varepsilon\rangle \wedge \neg\langle \gamma_i = \downarrow[\psi_0] \ldots \downarrow[\psi_{j_0}]\downarrow[\rho_{j_0+1}] \ldots \downarrow[\rho_{l_0}]\varepsilon\rangle$$

is consistent. We prove that $\neg\langle \varepsilon = \downarrow[\rho_{j_0+1}] \ldots \downarrow[\rho_{l_0}]\varepsilon\rangle$ is a conjunct of ψ_{j_0}, from which our desired property follows immediately since we have proved that $\widetilde{\mathcal{T}^{\psi_0}}, x \models \psi_{j_0}$. Aiming for a contradiction, suppose instead that $\langle \varepsilon = \downarrow[\rho_{j_0+1}] \ldots \downarrow[\rho_{l_0}]\varepsilon\rangle$ is a conjunct of ψ_{j_0}. Then, as $\alpha = \downarrow[\psi_1] \ldots \downarrow[\psi_{j_0}]\varepsilon$, we can derive that $\alpha \equiv \alpha[\langle \varepsilon = \downarrow[\rho_{j_0+1}] \ldots \downarrow[\rho_{l_0}]\varepsilon\rangle]$ (**Der21** (Fact 90)). Also observe that $\mathsf{XP}^- \vdash \beta_i \equiv \alpha\downarrow[\rho_{j_0+1}] \ldots \downarrow[\rho_{l_0}]\varepsilon$, and then we have

$$\begin{aligned}
\langle \gamma_i = \downarrow[\psi_0]\alpha\rangle &\equiv \langle \gamma_i = \downarrow[\psi_0]\alpha[\langle \varepsilon = \downarrow[\rho_{j_0+1}] \ldots \downarrow[\rho_{l_0}]\varepsilon\rangle]\rangle && (\textbf{Der21} \text{ (Fact 90)})\\
&\leq \langle \gamma_i = \downarrow[\psi_0]\alpha\downarrow[\rho_{j_0+1}] \ldots \downarrow[\rho_{l_0}]\varepsilon\rangle && (\mathsf{EqAx8})\\
&\equiv \langle \gamma_i = \downarrow[\psi_0]\beta_i\rangle
\end{aligned}$$

But using simple propositional reasoning, we have a contradiction with our hypothesis that $\langle \gamma_i = \downarrow[\psi_0]\alpha\rangle \wedge \neg\langle \gamma_i = \downarrow[\psi_0]\beta_i\rangle$ was consistent, a contradiction that came from assuming that $\langle \varepsilon = \downarrow[\rho_{j_0+1}] \ldots \downarrow[\rho_{l_0}]\varepsilon\rangle$ was a conjunct of ψ_{j_0}. □

Now that we have proved this lemma, we proceed to the formal construction of $\mathcal{T}^\varphi$, for $\varphi \in N^-_{n+1}$ (recall the base case at the beginning of our construction of the canonical model in §2.3.3).

Consider the following sets:

$$\begin{aligned}
\mathbf{V} &= \{(\psi, \alpha) \mid \langle \varepsilon = \downarrow[\psi]\alpha\rangle \text{ is a conjunct of } \varphi\}\\
\mathbf{U} &= \{(\psi, \alpha, \rho, \beta) \mid \langle \downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle \text{ is a conjunct of } \varphi\}
\end{aligned}$$

Rule 1. Witnesses for $\mathbf{v} = (\boldsymbol{\psi}, \boldsymbol{\alpha}) \in \boldsymbol{V}$. We define a data tree $\mathcal{T}^{\mathbf{v}} = (T^{\mathbf{v}}, \pi^{\mathbf{v}})$ with root $r^{\mathbf{v}}$. By inductive hypothesis, there exists a tree $\mathcal{T}^\psi$ such that ψ is satisfiable in that tree. In Lemma 104, consider

$$\begin{aligned}
\psi_0 &:= \psi\\
\mathcal{T}^{\psi_0} &:= \mathcal{T}^\psi\\
\alpha &:= \alpha\\
\{\beta_1, \ldots, \beta_m\} &:= \{\beta \in P^-_n \mid \neg\langle \varepsilon = \downarrow[\psi]\beta\rangle \text{ is a conjunct of } \varphi\}\\
\gamma_i &:= \varepsilon \text{ for all } i = 1, \ldots, m
\end{aligned}$$

Then there exists $\widetilde{\mathcal{T}^\psi} = (\widetilde{T^\psi}, \widetilde{\pi^\psi})$ with root $\widetilde{r^\psi}$ and a node x such that

- $\widetilde{\mathcal{T}^\psi}, \widetilde{r^\psi} \models \psi$,

- $\widetilde{\mathcal{T}^{\psi}}, \widetilde{r^{\psi}}, x \models \alpha$,
- $[x]_{\widetilde{\pi^{\psi}}} \neq [y]_{\widetilde{\pi^{\psi}}}$ for all y such that there is $\beta \in P_n^-$ with $(\psi, \beta) \notin \mathbf{V}$, and $\widetilde{\mathcal{T}^{\psi}}, \widetilde{r^{\psi}}, y \models \beta$

Define $\mathcal{T}^{\mathbf{v}}$ as $\widetilde{\mathcal{T}^{\psi}}$, and $x^{\mathbf{v}}$ as x. The root $r^{\mathbf{v}}$ and the partition $\pi^{\mathbf{v}}$ are the ones of $\widetilde{\mathcal{T}^{\psi}}$.

Rule 2. Witnesses for $\mathbf{u} = (\boldsymbol{\psi}, \boldsymbol{\alpha}, \boldsymbol{\rho}, \boldsymbol{\beta}) \in \boldsymbol{U}$. We define data trees $\mathcal{T}_1^{\mathbf{u}} = (T_1^{\mathbf{u}}, \pi_1^{\mathbf{u}})$ and $\mathcal{T}_2^{\mathbf{u}} = (T_2^{\mathbf{u}}, \pi_2^{\mathbf{u}})$ with roots $r_1^{\mathbf{u}}$, $r_2^{\mathbf{u}}$ respectively. By inductive hypothesis, there exist trees $\mathcal{T}^{\psi} = (T^{\psi}, \pi^{\psi})$ (with root r^{ψ}) and $\mathcal{T}^{\rho} = (T^{\rho}, \pi^{\rho})$ (with root r^{ρ}) such that ψ is satisfiable in $\mathcal{T}^{\psi}$ and ρ is satisfiable in $\mathcal{T}^{\rho}$. In Lemma 104, consider

$$\begin{aligned}
\psi_0 &:= \psi \\
\mathcal{T}^{\psi_0} &:= \mathcal{T}^{\psi} \\
\alpha &:= \alpha \\
\{\beta_1, \ldots, \beta_m\} &:= \{\gamma \in P_n^- \mid \neg\langle\downarrow[\rho]\beta = \downarrow[\psi]\gamma\rangle \text{ is a conjunct of } \varphi\} \\
\gamma_i &:= \downarrow[\rho]\beta \text{ for all } i = 1, \ldots, m
\end{aligned}$$

Then there exist $\widetilde{\mathcal{T}^{\psi}} = (\widetilde{T^{\psi}}, \widetilde{\pi^{\psi}})$ with root $\widetilde{r^{\psi}}$ and a node x such that:

- $\widetilde{\mathcal{T}^{\psi}}, \widetilde{r^{\psi}} \models \psi$,
- $\widetilde{\mathcal{T}^{\psi}}, \widetilde{r^{\psi}}, x \models \alpha$
- $[x]_{\widetilde{\pi^{\psi}}} \neq [y]_{\widetilde{\pi^{\psi}}}$ for all y such that there is $\gamma \in P_n^-$ with $\widetilde{\mathcal{T}^{\psi}}, \widetilde{r^{\psi}}, y \models \gamma$ and $\neg\langle\downarrow[\rho]\beta = \downarrow[\psi]\gamma\rangle$ is a conjunct of φ.

Define $T_1^{\mathbf{u}}$ as $\widetilde{T^{\psi}}$, $\pi_1^{\mathbf{u}}$ as $\widetilde{\pi^{\psi}}$, $r_1^{\mathbf{u}}$ as $\widetilde{r^{\psi}}$ and $x^{\mathbf{u}} = x \in T_1^{\mathbf{u}}$. Now let

$$\{\mu_1, \ldots, \mu_r\} = \left\{\mu \in P_n^- \mid \text{ there exists } \ y \in T_1^{\mathbf{u}} \text{ such that } \mathcal{T}_1^{\mathbf{u}}, r_1^{\mathbf{u}}, y \models \mu \text{ and } [y]_{\pi_1^{\mathbf{u}}} = [x^{\mathbf{u}}]_{\pi_1^{\mathbf{u}}}\right\}.$$

Then it follows that $\langle\downarrow[\rho]\beta = \downarrow[\psi]\mu_j\rangle$ is a conjunct of φ for all $j = 1, \ldots, r$. In Lemma 104, consider

$$\begin{aligned}
\psi_0 &:= \rho \\
\mathcal{T}^{\psi_0} &:= \mathcal{T}^{\rho} \\
\alpha &:= \beta \\
\{\beta_1, \ldots, \beta_m\} &:= \{\delta \in P_n^- \mid \exists j = 1, \ldots, r \text{ with } \neg\langle\downarrow[\rho]\delta = \downarrow[\psi]\mu_j\rangle \text{ is a conjunct of } \varphi\} \\
\gamma_i &:= \downarrow[\psi]\mu_j \text{ for } j = 1, \ldots r \text{ such that } \langle\downarrow[\rho]\beta_i = \downarrow[\psi]\mu_j\rangle \text{ is a conjunct of } \varphi.
\end{aligned}$$

Then there exist a tree $\widetilde{\mathcal{T}^{\rho}} = (\widetilde{T^{\rho}}, \widetilde{\pi^{\rho}})$ with root $\widetilde{r^{\rho}}$ and a node y such that

- $\widetilde{\mathcal{T}^{\rho}}, \widetilde{r^{\rho}} \models \rho$,

- $\widetilde{\mathcal{T}^\rho}, \widetilde{r^\rho}, y \models \beta$,
- $[y]_{\widetilde{\pi^\rho}} \neq [z]_{\widetilde{\pi^\rho}}$ for all z such that there is $\delta \in P_n^-$ and $j = 1, \ldots, r$ with $\widetilde{\mathcal{T}^\rho}, \widetilde{r^\rho}, z \models \delta$ and $\neg\langle\downarrow[\rho]\delta = \downarrow[\psi]\mu_j\rangle$ is a conjunct of φ.

Define $T_2^{\mathbf{u}}$ as $\widetilde{T^\rho}$, $\pi_2^{\mathbf{u}}$ as $\widetilde{\pi^\rho}$, $r_2^{\mathbf{u}}$ as $\widetilde{r^\rho}$ and $y^{\mathbf{u}} = y$. Without loss of generality, we assume that $T_1^{\mathbf{u}}$ and $T_2^{\mathbf{u}}$ are disjoint.

Now we define a partition $\pi^{\mathbf{u}}$ over $T_1^{\mathbf{u}} \cup T_2^{\mathbf{u}}$ as

$$\pi^{\mathbf{u}} = \pi_1^{\mathbf{u}} \cup \pi_2^{\mathbf{u}} \cup \{[x^{\mathbf{u}}]_{\pi_1^{\mathbf{u}}} \cup [y^{\mathbf{u}}]_{\pi_2^{\mathbf{u}}}\} \setminus \{[x^{\mathbf{u}}]_{\pi_1^{\mathbf{u}}}, [y^{\mathbf{u}}]_{\pi_2^{\mathbf{u}}}\}.$$

In other words, the rooted data tree $(T_1^{\mathbf{u}}, \pi^{\mathbf{u}}|_{T_1^{\mathbf{u}}}, r_1^{\mathbf{u}})$ is just a copy of $(\widetilde{T^\psi}, \widetilde{\pi^\psi}, \widetilde{r^\psi})$, with a special node named $x^{\mathbf{u}}$ which satisfies $T_1^{\mathbf{u}}, \pi^{\mathbf{u}}, r_1^{\mathbf{u}}, x^{\mathbf{u}} \models \alpha$. Analogously, the pointed data tree $(T_2^{\mathbf{u}}, \pi^{\mathbf{u}}|_{T_2^{\mathbf{u}}}, r_2^{\mathbf{u}})$ is a copy of $(\widetilde{T^\rho}, \widetilde{\pi^\rho}, \widetilde{r^\rho})$, with a special node named $y^{\mathbf{u}}$ which satisfies $T_2^{\mathbf{u}}, \pi^{\mathbf{u}}, r_2^{\mathbf{u}}, y^{\mathbf{u}} \models \beta$. Notice that the equivalence class $\sim$ induced by $\pi^{\mathbf{u}}$ (defined over the disjoint sets $T_1^{\mathbf{u}}$ and $T_2^{\mathbf{u}}$) is defined as $z \sim w$ iff $w \in [z]_{\widetilde{\pi^\psi}}$ or $w \in [z]_{\widetilde{\pi^\rho}}$, or both $w \in [x^{\mathbf{u}}]_{\widetilde{\pi^\psi}}$ and $z \in [y^{\mathbf{u}}]_{\widetilde{\pi^\rho}}$ or both $w \in [y^{\mathbf{u}}]_{\widetilde{\pi^\rho}}$ and $z \in [x^{\mathbf{u}}]_{\widetilde{\pi^\psi}}$. See Figure 25.

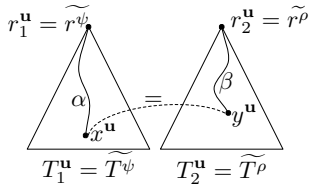


Figure 25: The data trees $\mathcal{T}_1^{\mathbf{u}} = (T_1^{\mathbf{u}}, \pi_1^{\mathbf{u}})$ and $\mathcal{T}_2^{\mathbf{u}} = (T_2^{\mathbf{u}}, \pi_2^{\mathbf{u}})$ for some $\mathbf{u} \in \mathbf{U}$. $\pi^{\mathbf{u_1}}$ and $\pi^{\mathbf{u_2}}$ are disjoint except that the equivalence class of $x^{\mathbf{u}}$ is merged with the equivalence class of $y^{\mathbf{u}}$.

The following remark will be used later to prove that Rule 2 does not spoil condition (C5) (cf. Figure 23(c)):

Remark 105. *Let $(\psi, \alpha, \rho, \beta) \in \mathbf{U}$. If $\neg\langle\downarrow[\psi]\mu = \downarrow[\rho]\delta\rangle$ is a conjunct of φ, then $[y^{\mathbf{u}}]_{\pi_2^{\mathbf{u}}} \neq [y]_{\pi_2^{\mathbf{u}}}$ for all y such that $\mathcal{T}_2^{\mathbf{u}}, r_2^{\mathbf{u}}, y \models \delta$ or $[x^{\mathbf{u}}]_{\pi_1^{\mathbf{u}}} \neq [x]_{\pi_1^{\mathbf{u}}}$ for all x such that $\mathcal{T}_1^{\mathbf{u}}, r_1^{\mathbf{u}}, x \models \mu$.*

Proof. The result is immediate from Rule 2: If neither of the conditions is satisfied, then $\mu = \mu_j$ for some $j = 1, \ldots, r$ and so $\langle\downarrow[\rho]\delta = \downarrow[\psi]\mu\rangle$ is a conjunct of φ which is a contradiction. □

The rooted data tree $(T^\varphi, \pi^\varphi, r^\varphi)$. Now, using Rule 1 and Rule 2, we define T^φ as the tree which consists of a root r^φ with label $a \in \mathbb{A}$ if a is a conjunct of φ, and with children

$$(T^{\mathbf{v}})_{\mathbf{v}\in\mathbf{V}} \ , \ (T_1^{\mathbf{u}})_{\mathbf{u}\in\mathbf{U}} \ , \ (T_2^{\mathbf{u}})_{\mathbf{u}\in\mathbf{U}}.$$

We assume that the nodes of all such trees are pairwise disjoint. Define π^φ over T^φ by

$$\pi^\varphi = \left(\bigcup_{\mathbf{v}\in\mathbf{V}} \pi^{\mathbf{v}} \setminus \{[x^{\mathbf{v}}]_{\pi^{\mathbf{v}}} \mid \mathbf{v} \in \mathbf{V}\} \right) \cup \left\{ \{r^\varphi\} \cup \bigcup_{\mathbf{v}\in\mathbf{V}} [x^{\mathbf{v}}]_{\pi^{\mathbf{v}}} \right\} \cup \bigcup_{\mathbf{u}\in\mathbf{U}} \pi^{\mathbf{u}}.$$

In other words, T^φ has a root, named r^φ, and children $(r^{\mathbf{v}})_{\mathbf{v}\in\mathbf{V}}$, $(r_1^{\mathbf{u}})_{\mathbf{u}\in\mathbf{U}}$, $(r_2^{\mathbf{u}})_{\mathbf{u}\in\mathbf{U}}$. Each of these children is the root of its corresponding tree inside T^φ as defined above, i.e. for each $\mathbf{v} \in \mathbf{V}$, $r^{\mathbf{v}}$ is the root of $T^{\mathbf{v}}$, and for each $\mathbf{u} \in \mathbf{U}$, $r_i^{\mathbf{u}}$ is the root of $T_i^{\mathbf{u}}$ $(i = 1, 2)$. All these subtrees are disjoint, and π^φ is defined as the disjoint union of partitions $\pi^{\mathbf{v}}$ for $\mathbf{v} \in \mathbf{V}$, and all $\pi^{\mathbf{u}}$ for $\mathbf{u} \in \mathbf{U}$, *with the exception* that we put into the same class the nodes r^φ and $(x^{\mathbf{v}})_{\mathbf{v}\in\mathbf{V}}$. See Figure 26.

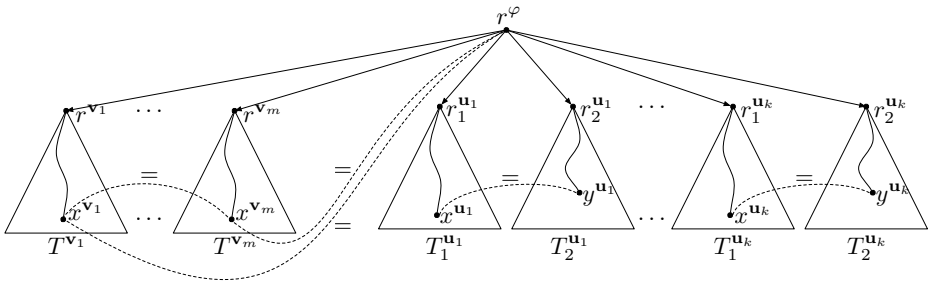


Figure 26: The data tree $\mathcal{T}^\varphi$, with root r^φ, when $\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$ and $\mathbf{U} = \{\mathbf{u}_1, \ldots, \mathbf{u}_k\}$. Nodes $r^\varphi, x^{\mathbf{v}_1}, \ldots, x^{\mathbf{v}_m}$ are in the same equivalence class, and for each i nodes $x^{\mathbf{u}_i}$ and $y^{\mathbf{u}_i}$ are in the same equivalence class.

The following fact follows easily by construction:

Fact 106. *The partition restricted to the trees $T^{\mathbf{v}}$ for $\mathbf{v} \in \mathbf{V}$ and the partition restricted to the trees $T_1^{\mathbf{u}}$ and $T_2^{\mathbf{u}}$ for $\mathbf{u} \in \mathbf{U}$ remains unchanged. More formally:*

- *For each $\mathbf{v} = (\psi, \alpha) \in \mathbf{V}$, we have $\pi^\varphi|_{T^{\mathbf{v}}} = \pi^{\mathbf{v}}$.*
- *For each $\mathbf{u} = (\psi, \alpha, \rho, \beta) \in \mathbf{U}$, we have $\pi^\varphi|_{T_1^{\mathbf{u}}} = \pi_1^{\mathbf{u}}$, and $\pi^\varphi|_{T_2^{\mathbf{u}}} = \pi_2^{\mathbf{u}}$.*

We conclude from Proposition 88 and the construction that:

Fact 107. *The validity of a formula in a child of r^φ is preserved in $\mathcal{T}^\varphi$. More formally:*

- *For each $\mathbf{v} \in \mathbf{V}$ and $x, y \in T^{\mathbf{v}}$ we have $\mathcal{T}^\varphi, x \equiv^{\downarrow -} \mathcal{T}^{\mathbf{v}}, x$ and $\mathcal{T}^\varphi, x, y \equiv^{\downarrow -} \mathcal{T}^{\mathbf{v}}, x, y$.*
- *For each $\mathbf{u} \in \mathbf{U}$, $i \in \{1, 2\}$ and $x, y \in T_i^{\mathbf{u}}$ we have $\mathcal{T}^\varphi, x \equiv^{\downarrow -} \mathcal{T}_i^{\mathbf{u}}, x$ and $\mathcal{T}^\varphi, x, y \equiv^{\downarrow -} \mathcal{T}_i^{\mathbf{u}}, x, y$.*

It only remains to check that conditions (C1)–(C5) are satisfied:

Verification of (C1). This condition is trivially satisfied.

Verification of (C2). Suppose $\langle \varepsilon = \downarrow[\psi]\alpha\rangle$ is a conjunct of φ. Then, by Rule 1, there is $x^{\mathbf{v}} \in T^{\varphi}$ such that $[r^{\varphi}]_{\pi^{\varphi}} = [x^{\mathbf{v}}]_{\pi^{\varphi}}$, with $\mathbf{v} = (\psi, \alpha)$. We also know by construction that $\mathcal{T}^{\mathbf{v}}, r^{\mathbf{v}} \models \psi$ and $\mathcal{T}^{\mathbf{v}}, r^{\mathbf{v}}, x^{\mathbf{v}} \models \alpha$. By Fact 107 we conclude $\mathcal{T}^{\varphi}, r^{\varphi} \models \langle \varepsilon = \downarrow[\psi]\alpha\rangle$.

Verification of (C3). Suppose $\langle \downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$ is a conjunct of φ. Then, by Rule 2, there are $x^{\mathbf{u}}, y^{\mathbf{u}} \in \mathcal{T}^{\varphi}$ such that $[x^{\mathbf{u}}]_{\pi^{\varphi}} = [y^{\mathbf{u}}]_{\pi^{\varphi}}$, with $\mathbf{u} = (\psi, \alpha, \rho, \beta)$. We also know on the one hand that $\mathcal{T}_1^{\mathbf{u}}, r_1^{\mathbf{u}} \models \psi$ and $\mathcal{T}_2^{\mathbf{u}}, r_2^{\mathbf{u}} \models \rho$, and on the other hand that $\mathcal{T}_1^{\mathbf{u}}, r_1^{\mathbf{u}}, x^{\mathbf{u}} \models \alpha$ and $\mathcal{T}_2^{\mathbf{u}}, r_2^{\mathbf{u}}, y^{\mathbf{u}} \models \beta$. By Fact 107 we conclude $\mathcal{T}^{\varphi}, r^{\varphi} \models \langle \downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$.

Verification of (C4). Suppose $\neg\langle \varepsilon = \downarrow[\psi]\alpha\rangle$ is a conjunct of φ. Aiming for a contradiction, suppose that $\mathcal{T}^{\varphi}, r^{\varphi} \models \langle \varepsilon = \downarrow[\psi]\alpha\rangle$. Then there is a successor z of r^{φ} in which ψ holds, and by construction plus Lemma 96, z is the root of some copy of a data tree $\widetilde{\mathcal{T}^{\psi}}$. Moreover, there is $x \in \widetilde{T^{\psi}}$ such that $\mathcal{T}^{\varphi}, z, x \models \alpha$, with $[x]_{\pi^{\varphi}} = [r^{\varphi}]_{\pi^{\varphi}}$. In addition to this, $(\psi, \alpha) \notin \mathbf{V}$ and so, by Rule 1, $[x]_{\pi^{\varphi}} \neq [x^{\mathbf{v}}]_{\pi^{\varphi}}$ for all $\mathbf{v} \in \mathbf{V}$. Then, by construction, $[x]_{\pi^{\varphi}} \neq [r^{\varphi}]_{\pi^{\varphi}}$ which is a contradiction.

Verification of (C5). Suppose $\neg\langle \downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$ is a conjunct of φ. Aiming for a contradiction, suppose that $\mathcal{T}^{\varphi}, r^{\varphi} \models \langle \downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$. Then there are successors z_1 and z_2 of r^{φ} in which ψ and ρ holds, respectively. Also, by construction and Lemma 96, z_1 and z_2 are the roots of some copies of data trees $\widetilde{\mathcal{T}^{\psi}}$ and $\widetilde{\mathcal{T}^{\rho}}$ (note that we are using the notation $\widetilde{\mathcal{T}^{\psi}}$ and $\widetilde{\mathcal{T}^{\rho}}$ either if the tree is the one obtained by inductive hypothesis or a modified version of it). Moreover, there are descendants w_1 and w_2 such that $\mathcal{T}^{\varphi}, z_1, w_1 \models \alpha$, $\mathcal{T}^{\varphi}, z_2, w_2 \models \beta$ and $[w_1]_{\pi^{\varphi}} = [w_2]_{\pi^{\varphi}}$. We now have two cases to analyze:

- $\widetilde{\mathcal{T}^{\psi}} = \widetilde{\mathcal{T}^{\rho}}$: In this case, because of Lemma 96, $\psi = \rho$. And we have $\widetilde{\mathcal{T}^{\psi}}, \widetilde{r^{\psi}} \models \langle \alpha = \beta\rangle$, and as a consequence $\langle \alpha = \beta\rangle$ has to be a conjunct of ψ (Lemma 94). We prove that in this case $\langle \downarrow[\psi]\alpha' = \downarrow[\psi]\alpha'\rangle$ can not be a conjunct of φ for any $\alpha' \in P_n^-$: If this were the case, $\langle \downarrow[\psi]\alpha' = \downarrow[\psi]\alpha'\rangle \wedge \neg\langle \downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$ would be consistent, but:

$$
\begin{aligned}
&\langle \downarrow[\psi]\alpha' = \downarrow[\psi]\alpha'\rangle \wedge \neg\langle \downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle \\
&\quad \leq \langle \downarrow[\psi]\rangle \wedge \neg\langle \downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle && (\textbf{Der12} \text{ (Fact 90)}) \\
&\quad \equiv \langle \downarrow[\psi \wedge \langle \alpha = \beta\rangle]\rangle \wedge \neg\langle \downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle && (\langle \alpha = \beta\rangle \text{ is a conjunct of } \psi) \\
&\quad \equiv \langle \downarrow[\psi][\langle \alpha = \beta\rangle]\rangle \wedge \neg\langle \downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle && (\textbf{Der21} \text{ (Fact 90)}) \\
&\quad \leq \langle \downarrow[\psi]\alpha = \downarrow[\psi]\beta\rangle \wedge \neg\langle \downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle && (\text{EqAx5}) \\
&\quad \equiv \text{FALSE} && (\text{Boolean})
\end{aligned}
$$

  which is a contradiction.

  Then, $\langle \downarrow[\psi]\alpha' = \downarrow[\psi]\alpha'\rangle$ is not a conjunct of φ and so, it follows easily from the consistency of φ that $(\psi, \alpha') \notin \mathbf{V}$ for all $\alpha' \in P_n^-$. And also $(\psi, \alpha', \rho', \beta') \notin \mathbf{U}$ for all

$\alpha', \beta' \in P_n^-$, $\rho' \in N_n^-$. This gives a contradiction by construction because in this case it would not be a copy of a tree $\widetilde{\mathcal{T}^\psi}$.

- $\widetilde{\mathcal{T}^\psi} \neq \widetilde{\mathcal{T}^\rho}$: In this case, there are two possibilities to consider:
  - One possibility is that $[w_1]_{\pi^\varphi} = [w_2]_{\pi^\varphi}$ because Rule 2 was applied (see Figure 27 (a)). Then there is $\mathbf{u} = (\psi, \alpha', \rho, \beta') \in \mathbf{U}$ (the symmetric case is analogous). In this case we have $\mathcal{T}_1^{\mathbf{u}}, r_1^{\mathbf{u}}, x^{\mathbf{u}} \models \alpha'$ and $\mathcal{T}_2^{\mathbf{u}}, r_2^{\mathbf{u}}, y^{\mathbf{u}} \models \beta'$. Furthermore, since $[w_1]_{\pi^\varphi} = [w_2]_{\pi^\varphi}$, we have that $[w_1]_{\pi_1^{\mathbf{u}}} = [x^{\mathbf{u}}]_{\pi_1^{\mathbf{u}}}$ and $[w_2]_{\pi_2^{\mathbf{u}}} = [y^{\mathbf{u}}]_{\pi_2^{\mathbf{u}}}$ which is a contradiction by Remark 105.
  - The other possibility is that $[w_1]_{\pi^\varphi} = [w_2]_{\pi^\varphi}$ because Rule 1 was applied twice (see Figure 27(b)). Then there exist $\mathbf{v}_1 = (\psi, \alpha')$, $\mathbf{v}_2 = (\rho, \beta') \in \mathbf{V}$. In this case we have $\mathcal{T}^{\mathbf{v}_1}, r^{\mathbf{v}_1}, x^{\mathbf{v}_1} \models \alpha'$ and $\mathcal{T}^{\mathbf{v}_2}, r^{\mathbf{v}_2}, x^{\mathbf{v}_2} \models \beta'$. Furthermore, since $[w_1]_{\pi^\varphi} = [w_2]_{\pi^\varphi}$, we have that $[w_1]_{\pi^{\mathbf{v}_1}} = [x^{\mathbf{v}_1}]_{\pi^{\mathbf{v}_1}}$ and $[w_2]_{\pi^{\mathbf{v}_2}} = [y^{\mathbf{v}_2}]_{\pi^{\mathbf{v}_2}}$. Then, by Rule 1, (ψ, α) and (ρ, β) belong to $\mathbf{V}$ which gives a contradiction because of the consistency of φ plus EqAx7.

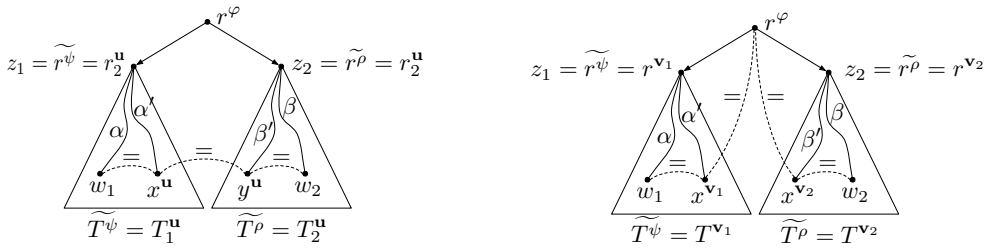


Figure 27: Nodes w_1 and w_2 are in the same equivalence class because (a) Rule 2 was applied via $\mathbf{u} = (\psi, \alpha', \rho, \beta') \in \mathbf{U}$, or (b) Rule 1 was applied twice via $\mathbf{v}_1 = (\psi, \alpha')$, $\mathbf{v}_2 = (\rho, \beta') \in \mathbf{V}$.

2.4 Axiomatic System for XPath$_=(\downarrow)$

2.4.1 Axiomatization

In this subsection we introduce additional axiom schemes to handle inequalities. Axioms schemes in Table 2.2 extend those from Table 2.1 to form a complete axiomatic system for the full logic XPath$_=(\downarrow)$. Observe that NeqAx1 – NeqAx5 are analogous to EqAx1 – EqAx5.

Let XP be the set of all instantiations of the axiom schemes from Table 2.1 plus the ones from Table 2.2. In the scope of this section we will often say that a node expression is *consistent* meaning that it is XP-consistent (as in Definition 89).

Node axiom schemes for inequality

NeqAx1	$\langle \alpha \neq \beta \rangle$	$\equiv$	$\langle \beta \neq \alpha \rangle$	Analogous to EqAx1 – EqAx5 but with symbol $\neq$ instead of $=$
NeqAx2	$\langle \alpha \cup \beta \neq \gamma \rangle$	$\equiv$	$\langle \alpha \neq \gamma \rangle \vee \langle \beta \neq \gamma \rangle$	
NeqAx3	$\varphi \wedge \langle \alpha \neq \beta \rangle$	$\equiv$	$\langle [\varphi]\alpha \neq \beta \rangle$	
NeqAx4	$\langle \alpha \neq \beta \rangle$	$\leq$	$\langle \alpha \rangle$	
NeqAx5	$\langle \gamma[\langle \alpha \neq \beta \rangle] \rangle$	$\leq$	$\langle \gamma\alpha \neq \gamma\beta \rangle$	
NeqAx6	$\langle \alpha = \gamma \rangle \wedge \langle \beta = \eta \rangle$	$\leq$	$\langle \alpha = \beta \rangle \vee \langle \gamma \neq \eta \rangle$	
NeqAx7	$\langle \alpha \neq \gamma \rangle \wedge \langle \beta = \eta \rangle$	$\leq$	$\langle \alpha \neq \beta \rangle \vee \langle \gamma \neq \eta \rangle$	
NeqAx8	$\langle \gamma = \eta[\neg\langle \alpha = \beta \rangle \wedge \langle \alpha \rangle]\beta \rangle$	$\leq$	$\langle \gamma \neq \eta\alpha \rangle$	
NeqAx9	$\langle \gamma \neq \eta[\neg\langle \alpha \neq \beta \rangle \wedge \langle \alpha \rangle]\beta \rangle$	$\leq$	$\langle \gamma \neq \eta\alpha \rangle$	
NeqAx10	$\langle \gamma = \eta[\neg\langle \alpha \neq \alpha \rangle \wedge \langle \alpha = \beta \rangle]\alpha \rangle$	$\leq$	$\langle \gamma = \eta\beta \rangle$	

Table 2.2: Additional axiom schemes to allow for data inequality tests. The axiomatic system XP consists of all the instantiations of this table, plus the ones of Table 2.1.

Sometimes we use NeqAx1 and NeqAx4 without explicitly mentioning them. We omit such steps in order to make the proofs more readable. We also note that NeqAx2 and NeqAx3 are necessary for the proof of Theorem 113, which is omitted; they have to be used in the same way as EqAx2 and EqAx3 in the proof of Theorem 101.

It is not difficult to see that the axioms XP are sound for XPath$_=(\downarrow)$:

Proposition 108 (Soundness of XPath$_=(\downarrow)$)**.**

1. *Let φ and ψ be node expressions of* XPath$_=(\downarrow)$*. Then* $\mathsf{XP} \vdash \varphi \equiv \psi$ *implies* $\models \varphi \equiv \psi$*.*
2. *Let α and β be path expressions of* XPath$_=(\downarrow)$*. Then* $\mathsf{XP} \vdash \alpha \equiv \beta$ *implies* $\models \alpha \equiv \beta$*.*

2.4.2 Normal forms

We define the sets P_n and N_n, that contain the path and node expressions of XPath$_=(\downarrow)$, respectively, in normal form at level n:

Definition 109 (Normal form for XPath$_=(\downarrow)$)**.**

$$
\begin{aligned}
P_0 &= \{\varepsilon\} \\
N_0 &= \{a \wedge \langle \varepsilon = \varepsilon \rangle \wedge \neg\langle \varepsilon \neq \varepsilon \rangle \mid a \in \mathbb{A}\} \\
P_{n+1} &= \{\varepsilon\} \cup \{\downarrow[\psi]\beta \mid \psi \in N_n, \beta \in P_n\} \\
D_{n+1} &= \{\langle \alpha = \beta \rangle \mid \alpha, \beta \in P_{n+1}\} \cup \{\langle \alpha \neq \beta \rangle \mid \alpha, \beta \in P_{n+1}\} \\
N_{n+1} &= \left\{ a \wedge \bigwedge_{\varphi \in C} \varphi \wedge \bigwedge_{\varphi \in D_{n+1} \setminus C} \neg\varphi \mid C \subseteq D_{n+1}, a \in \mathbb{A} \right\} \cap \mathbf{Con}_{\mathsf{XP}}.
\end{aligned}
$$

Normal forms are built using the same idea from §2.3.2, but considering also data-aware diamonds with inequalities. Again, let us remark that it would suffice that N_0 contains formulas of the form a, for $a \in \mathbb{A}$, but we include instead formulas of the form $a \wedge \langle \varepsilon = \varepsilon \rangle \wedge \neg\langle \varepsilon \neq \varepsilon \rangle$ (containing the tautologies $\langle \varepsilon = \varepsilon \rangle$ and $\neg\langle \varepsilon \neq \varepsilon \rangle$) for technical reasons. For instance, considering again two labels a and b, the node expressions of N_0 are

$$\psi = a \wedge \langle \varepsilon = \varepsilon \rangle \wedge \neg\langle \varepsilon \neq \varepsilon \rangle \quad \text{and} \quad \theta = b \wedge \langle \varepsilon = \varepsilon \rangle \wedge \neg\langle \varepsilon \neq \varepsilon \rangle.$$

The sets P_1 and D_1 are as follows:

$$\begin{aligned} P_1 &= \{\downarrow[\psi]\varepsilon, \downarrow[\theta]\varepsilon, \varepsilon\} \\ D_1 &= \{\langle \varepsilon = \varepsilon \rangle, \langle \downarrow[\psi]\varepsilon = \downarrow[\theta]\varepsilon \rangle, \langle \varepsilon = \downarrow[\psi]\varepsilon \rangle, \langle \varepsilon = \downarrow[\theta]\varepsilon \rangle, \langle \downarrow[\psi]\varepsilon = \downarrow[\psi]\varepsilon \rangle, \langle \downarrow[\theta]\varepsilon = \downarrow[\theta]\varepsilon \rangle, \\ &\quad \langle \varepsilon \neq \varepsilon \rangle, \langle \downarrow[\psi]\varepsilon \neq \downarrow[\theta]\varepsilon \rangle, \langle \varepsilon \neq \downarrow[\psi]\varepsilon \rangle, \langle \varepsilon \neq \downarrow[\theta]\varepsilon \rangle, \langle \downarrow[\psi]\varepsilon \neq \downarrow[\psi]\varepsilon \rangle, \langle \downarrow[\theta]\varepsilon \neq \downarrow[\theta]\varepsilon \rangle\} \end{aligned}$$

An example of a node expression in normal form at level 1, i.e. a node expression in N_1, is

$$\begin{aligned} \varphi = {} & a \wedge \langle \varepsilon = \varepsilon \rangle \wedge \neg\langle \varepsilon \neq \varepsilon \rangle \wedge \langle \downarrow[\psi]\varepsilon = \downarrow[\theta]\varepsilon \rangle \wedge \langle \downarrow[\psi]\varepsilon = \downarrow[\psi]\varepsilon \rangle \wedge \langle \downarrow[\theta]\varepsilon = \downarrow[\theta]\varepsilon \rangle \wedge \\ & \wedge \langle \varepsilon \neq \downarrow[\psi]\varepsilon \rangle \wedge \langle \varepsilon \neq \downarrow[\theta]\varepsilon \rangle \wedge \langle \downarrow[\psi]\varepsilon \neq \downarrow[\theta]\varepsilon \rangle \wedge \neg\langle \varepsilon = \downarrow[\psi]\varepsilon \rangle \wedge \neg\langle \varepsilon = \downarrow[\theta]\varepsilon \rangle \wedge \\ & \wedge \langle \downarrow[\theta]\varepsilon \neq \downarrow[\theta]\varepsilon \rangle \wedge \langle \downarrow[\psi]\varepsilon \neq \downarrow[\psi]\varepsilon \rangle. \end{aligned}$$

Analogs of Lemmas 94, 95 and 96 hold in this case, with the same proofs as those given for the case of $\mathrm{XPath}_{=}(\downarrow)^{-}$:

Lemma 110. *Let $* \in \{=, \neq\}$, $\psi \in N_n$ and $\alpha, \alpha' \in P_n$. Let $\mathcal{T}, u$ be a pointed data tree, such that $\mathcal{T}, u \models \psi$ and $\mathcal{T}, u \models \langle \alpha * \alpha' \rangle$. Then $\langle \alpha * \alpha' \rangle$ is a conjunct of ψ*

Lemma 111. *Let $\psi \in N_n$ and $\alpha \in P_n$. If $[\psi]\alpha$ is consistent then $\langle \alpha = \alpha \rangle$ is a conjunct of ψ. As an immediate consequence, if $\langle \downarrow[\psi]\alpha \rangle$ is consistent then $\langle \alpha = \alpha \rangle$ is a conjunct of ψ.*

Lemma 112. *For every pair of distinct elements $\varphi, \psi \in N_n$, $\varphi \wedge \psi$ is inconsistent.*

We omit the proof of the following theorem, since it is analogous to the one for XP^{-} (Theorem 101):

Theorem 113 (Normal form for $\mathrm{XPath}_{=}(\downarrow)$)**.** *Let φ be a consistent node expression of $\mathrm{XPath}_{=}(\downarrow)$ such that $\mathsf{dd}(\varphi) = n$. Then $\mathsf{XP} \vdash \varphi \equiv \bigvee_i \varphi_i$ for some $(\varphi_i)_{1 \leq i \leq k} \in N_n$. Let α be a consistent path expression of $\mathrm{XPath}_{=}(\downarrow)$ such that $\mathsf{dd}(\alpha) = n$. Then $\mathsf{XP} \vdash \alpha \equiv \bigcup_i [\varphi_i]\alpha_i$ for some $(\alpha_i)_{1 \leq i \leq k} \in P_n$ and $(\varphi_i)_{1 \leq i \leq k} \in N_n$. Furthermore, if α is ε or starting with $\downarrow$ then $\mathsf{XP} \vdash \alpha \equiv \bigcup_i \alpha_i$ for some $(\alpha_i)_{1 \leq i \leq k} \in P_n$.*

The following two technical lemmas, whose proofs are deferred to §2.6, will be needed for the construction of the canonical model:

Lemma 114. *Let $* \in \{=, \neq\}$, $\gamma \in P_n$, $\psi_i \in N_{n-i}$ for $i = 1, \ldots, i_0$, $\alpha, \beta \in P_{n-i_0}$ such that*

$$\langle \gamma * \downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\alpha \rangle \wedge \neg\langle \gamma * \downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\beta \rangle$$

is consistent and $\neg\langle \alpha \neq \alpha \rangle$ is a conjunct of ψ_{i_0}. Then $\neg\langle \alpha = \beta \rangle$ is a conjunct of ψ_{i_0}.

Lemma 115. *Let $\psi \in N_n$, $\alpha, \beta \in P_n$ such that $\langle \downarrow[\psi]\alpha \neq \downarrow[\psi]\alpha \rangle \wedge \neg\langle \downarrow[\psi]\gamma \neq \downarrow[\psi]\gamma \rangle$ is consistent and $\neg\langle \alpha \neq \alpha \rangle$ is a conjunct of ψ. Then $\neg\langle \alpha = \gamma \rangle$ is a conjunct of ψ.*

2.4.3 Completeness for node and path expressions

In this subsection we show that for node expressions φ and ψ of XPath$_=(\downarrow)$, the equivalence $\varphi \equiv \psi$ is derivable from the axiom schemes of Table 2.1 plus Table 2.2 if and only if φ is XPath$_=(\downarrow)$-semantically equivalent to ψ. We also show the corresponding result for path expressions of XPath$_=(\downarrow)$.

Theorem 116 (Completeness of XPath$_=(\downarrow)$)**.**

1. *Let φ and ψ be node expressions of* XPath$_=(\downarrow)$*. Then* $\mathsf{XP} \vdash \varphi \equiv \psi$ *iff* $\models \varphi \equiv \psi$*.*
2. *Let α and β be path expressions of* XPath$_=(\downarrow)$*. Then* $\mathsf{XP} \vdash \alpha \equiv \beta$ *iff* $\models \alpha \equiv \beta$*.*

The proof of the above theorem is analogous to that of Theorem 103. The critical part of the argumentation is the analog of Lemma 102 for the more expressive logic XPath$_=(\downarrow)$:

Lemma 117. *Any node expression $\varphi \in N_n$ is satisfiable.*

The rest of this section is devoted to the proof of Lemma 117.

Canonical model

We construct, recursively in n and for every $\varphi \in N_n$, a data tree $\mathcal{T}^\varphi = (T^\varphi, \pi^\varphi)$ such that φ is satisfiable in $\mathcal{T}^\varphi$.

For the base case, if $\varphi \in N_0$ and $\varphi = a \wedge \langle \varepsilon = \varepsilon \rangle \wedge \neg \langle \varepsilon \neq \varepsilon \rangle$ with $a \in \mathbb{A}$, we define the data tree $\mathcal{T}^\varphi = (T^\varphi, \pi^\varphi)$ where T^φ is a tree which consists of the single node x with label a, and $\pi^\varphi = \{\{x\}\}$.

Now, let $\varphi \in N_{n+1}$. Since φ is a conjunction as in Definition 109, it is enough to guarantee that the following conditions hold (observe that we are using EqAx1 and NeqAx1 but we usually avoid these observations of symmetry):

(C1) If $a \in \mathbb{A}$ is a conjunct of φ, then the root r^φ of $\mathcal{T}^\varphi$ has label a.

(C2) If $\langle \varepsilon = \downarrow[\psi]\alpha \rangle$ is a conjunct of φ, then there is a child $r^{\mathbf{v}}$ of the root r^φ of $\mathcal{T}^\varphi$ at which ψ is satisfied, and a node $x^{\mathbf{v}}$ with the same data value as r^φ such that $\mathcal{T}^\varphi, r^{\mathbf{v}}, x^{\mathbf{v}} \models \alpha$.

(C3) If $\langle \varepsilon \neq \downarrow[\psi]\alpha \rangle$ is a conjunct of φ, then there is a child $r^{\mathbf{v}}$ of the root r^φ of $\mathcal{T}^\varphi$ at which ψ is satisfied, and a node $x^{\mathbf{v}}$ with different data value than r^φ such that $\mathcal{T}^\varphi, r^{\mathbf{v}}, x^{\mathbf{v}} \models \alpha$.

(C4) If $\langle \downarrow[\psi]\alpha = \downarrow[\rho]\beta \rangle$ is a conjunct of φ, then there are two children $r_1^{\mathbf{u}}$, $r_2^{\mathbf{u}}$ of the root r^φ of $\mathcal{T}^\varphi$ at which ψ and ρ are satisfied respectively, and there are nodes $x^{\mathbf{u}}$ and $y^{\mathbf{u}}$ with the same data value such that $\mathcal{T}^\varphi, r_1^{\mathbf{u}}, x^{\mathbf{u}} \models \alpha$ and $\mathcal{T}^\varphi, r_2^{\mathbf{u}}, y^{\mathbf{u}} \models \beta$.

(C5) If $\langle \downarrow[\psi]\alpha \neq \downarrow[\rho]\beta \rangle$ is a conjunct of φ, then there are two children $r_1^{\mathbf{u}}$, $r_2^{\mathbf{u}}$ of the root r^φ of $\mathcal{T}^\varphi$ at which ψ and ρ are satisfied respectively, and there are nodes $x^{\mathbf{u}}$ and $y^{\mathbf{u}}$ with different data value such that $\mathcal{T}^\varphi, r_1^{\mathbf{u}}, x^{\mathbf{u}} \models \alpha$ and $\mathcal{T}^\varphi, r_2^{\mathbf{u}}, y^{\mathbf{u}} \models \beta$.

(C6) If $\neg\langle\varepsilon = \downarrow[\psi]\alpha\rangle$ is a conjunct of φ, then for each child z of the root r^φ of $\mathcal{T}^\varphi$ at which ψ is satisfied, if x is a node such that $\mathcal{T}^\varphi, z, x \models \alpha$, then the data value of x is different than the one of r^φ.

(C7) If $\neg\langle\varepsilon \neq \downarrow[\psi]\alpha\rangle$ is a conjunct of φ, then for each child z of the root r^φ of $\mathcal{T}^\varphi$ at which ψ is satisfied, if x is a node such that $\mathcal{T}^\varphi, z, x \models \alpha$, then the data value of x is the same as the one of r^φ.

(C8) If $\neg\langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$ is a conjunct of φ, then for each children z_1, z_2 of the root r^φ of $\mathcal{T}^\varphi$ at which ψ and ρ are satisfied respectively, if w_1, w_2 are nodes such that $\mathcal{T}^\varphi, z_1, w_1 \models \alpha$ and $\mathcal{T}^\varphi, z_2, w_2 \models \beta$, then the data values of w_1 and w_2 are different.

(C9) If $\neg\langle\downarrow[\psi]\alpha \neq \downarrow[\rho]\beta\rangle$ is a conjunct of φ, then for each children z_1, z_2 of the root r^φ of $\mathcal{T}^\varphi$ at which ψ and ρ are satisfied respectively, if w_1, w_2 are nodes such that $\mathcal{T}^\varphi, z_1, w_1 \models \alpha$ and $\mathcal{T}^\varphi, z_2, w_2 \models \beta$, then w_1 and w_2 have the same data value.

As in §2.3.3, we first give an intuitive description of the construction of the model, and then proceed to formalize it:

Insight into the construction

The construction given in §2.3.3 has some similarities with the one we are about to present. As before, we will hang, from a common root, copies of trees given by inductive hypothesis to guarantee the satisfaction of some conjuncts of φ. Like in the previous case, we may need to introduce some changes on those trees in order to avoid spoiling the satisfaction of other conjuncts.

However, this construction is far more complex than the one for $\mathrm{XPath}_{=}(\downarrow)^{-}$. In the previous case, when adding new witnesses with fresh data values, one only needed to be careful enough to avoid putting in the same class nodes that should be in different classes. Now, in addition to that (which is also harder to achieve, as witnessed by the differences between Lemmas 104 and 126 explained at the end of the latter), one also needs to guarantee conditions of the form $\neg\langle\mu \neq \delta\rangle$ with $\mu, \delta \in P_{n+1}$ which force the merging of classes of *every* witness of the kind of paths involved that could appear along the construction.

Unlike the case of $\mathrm{XPath}_{=}(\downarrow)^{-}$, each pair of path expressions μ, δ in P_{n+1} will occur in two conjuncts of φ instead of one (we do not care about symmetric repetitions). Indeed, in the case of $\mathrm{XPath}_{=}(\downarrow)^{-}$, for μ, δ in P^{-}_{n+1}, we either have $\langle\mu = \delta\rangle$ or $\neg\langle\mu = \delta\rangle$ as a conjunct of a node expression in N^{-}_{n+1}. Now we have four choices because we also have either $\langle\mu \neq \delta\rangle$ or $\neg\langle\mu \neq \delta\rangle$, and hence *two* conjuncts containing μ and δ will occur in node expressions of N_{n+1}. We cannot treat as separate from each other those two conjuncts in which the same pair μ, δ appear, so we first split P_{n+1} into four subsets to deal with diamonds that compare against the constant empty path:

$$\mathbf{V}_{=,\neq} = \{(\psi, \alpha) \mid \psi \in N_n, \alpha \in P_n, \langle\varepsilon = \downarrow[\psi]\alpha\rangle \text{ and } \langle\varepsilon \neq \downarrow[\psi]\alpha\rangle \text{ are conjuncts of } \varphi\}$$

$$\mathbf{V}_{=,\neg\neq} = \{(\psi,\alpha) \mid \psi \in N_n, \alpha \in P_n, \langle \varepsilon = \downarrow[\psi]\alpha\rangle \text{ and } \neg\langle \varepsilon \neq \downarrow[\psi]\alpha\rangle \text{ are conjuncts of } \varphi\}$$
$$\mathbf{V}_{\neg=,\neq} = \{(\psi,\alpha) \mid \psi \in N_n, \alpha \in P_n, \neg\langle \varepsilon = \downarrow[\psi]\alpha\rangle \text{ and } \langle \varepsilon \neq \downarrow[\psi]\alpha\rangle \text{ are conjuncts of } \varphi\}$$
$$\mathbf{V}_{\neg=,\neg\neq} = \{(\psi,\alpha) \mid \psi \in N_n, \alpha \in P_n, \neg\langle \varepsilon = \downarrow[\psi]\alpha\rangle \text{ and } \neg\langle \varepsilon \neq \downarrow[\psi]\alpha\rangle \text{ are conjuncts of } \varphi\}$$

We make the following observations regarding the above definitions:

Observation 118. *For $(\psi,\alpha) \in \mathbf{V}_{\neg=,\neg\neq}$, our axioms should tell us that either $\langle\alpha\rangle$ is not a conjunct of ψ or $\downarrow[\psi]\beta$ does not appear in any other positive conjunct of φ. If this is not the case, then φ would be clearly unsatisfiable and thus our axiomatic system would not be complete. This assertion is a consequence of the following lemma plus* **Der12** *of Fact 90. It is important to remark that the axioms required for the proof can be easily proven sound.*

Lemma 119. *Let $\psi \in N_n$, $\alpha \in P_n$, $\gamma \in P_{n+1}$. If $\neg\langle\gamma = \downarrow[\psi]\alpha\rangle \wedge \neg\langle\gamma \neq \downarrow[\psi]\alpha\rangle \wedge \langle\gamma\rangle \wedge \langle\downarrow[\psi]\rangle$ is consistent, then $\neg\langle\alpha = \alpha\rangle$ is a conjunct of ψ.*

Proof. See §2.6. □

Then, by Lemma 112 plus the fact that we will construct our model by hanging from the root *the trees given by inductive hypothesis, we should not be worried about the satisfaction of either $\neg\langle\varepsilon = \downarrow[\psi]\alpha\rangle$ nor $\neg\langle\varepsilon \neq \downarrow[\psi]\alpha\rangle$ because we will never create a pair of nodes witnessing the path $\downarrow[\psi]\alpha$.*

Observation 120. *For $(\psi,\alpha) \in \mathbf{V}_{=,\neg\neq}$, our axioms should tell us that in a tree $\mathcal{T}^\psi$, any pair of nodes satisfying α ends in a node in the same equivalence class, since we want to put any such node in the class of the root r^φ. The following lemma has this property as an immediate consequence.*

Lemma 121. *Let $* \in \{=,\neq\}$, $\psi \in N_n$, $\alpha,\beta \in P_n$, $\gamma \in P_{n+1}$. If $\langle\gamma = \downarrow[\psi]\alpha\rangle \wedge \neg\langle\gamma \neq \downarrow[\psi]\alpha\rangle \wedge \neg\langle\gamma * \downarrow[\psi]\beta\rangle$ is consistent, then $\neg\langle\alpha * \beta\rangle$ is a conjunct of ψ.*

Proof. See §2.6. □

Observation 122. *For $(\psi,\alpha) \in \mathbf{V}_{=,\neg\neq}$ and $(\psi,\beta) \in \mathbf{V}_{\neg=,\neq}$, Lemma 121 also tells us that in a tree $\mathcal{T}^\psi$, any pairs of nodes satisfying α and β end in points in different equivalence classes; which is also necessary to be able to satisfy φ.*

Observation 123. *For $(\psi,\alpha) \in \mathbf{V}_{=,\neq}$ and $(\psi,\beta) \in \mathbf{V}_{=,\neg\neq}$, in order to obtain a witness for $\langle\varepsilon \neq \downarrow[\psi]\alpha\rangle$, our axioms should tell us that in a tree $\mathcal{T}^\psi$ we can find a pair of nodes satisfying α starting from the root, and such that its ending node is in a different class from that of the ending node of any pair of nodes satisfying β and beginning at the root of that tree. The following lemma combined with Observation 120 has this as an immediate consequence.*

Lemma 124. *Let $* \in \{=,\neq\}$, $\psi \in N_n$, $\alpha,\beta \in P_n$, $\gamma \in P_{n+1}$. If $\langle\gamma = \downarrow[\psi]\alpha\rangle \wedge \neg\langle\gamma \neq \downarrow[\psi]\alpha\rangle \wedge \langle\gamma * \downarrow[\psi]\beta\rangle$ is consistent, then $\langle\alpha * \beta\rangle$ is a conjunct of ψ.*

Proof. See §2.6. □

Observation 125. *For $(\psi, \alpha) \in \mathbf{V}_{=,\neq}$, in order to obtain a witness for $\langle \varepsilon = \downarrow[\psi]\alpha\rangle$, we need a tree in which ψ is satisfied and a pair of nodes (beginning at the root of that tree) satisfying α and ending in a node such that: it is in the class of the ending nodes of pairs of nodes satisfying β for $(\psi, \beta) \in \mathbf{V}_{=,\neg\neq}$, but it is not in the class of any ending node of a pair of nodes satisfying γ for $(\psi, \gamma) \in \mathbf{V}_{\neg=,\neq}$. In case there exists $\beta \in P_n$ such that $(\psi, \beta) \in \mathbf{V}_{=,\neg\neq}$, any tree at which ψ is satisfied will work by the previous observations and lemmas. But in case $(\psi, \beta) \notin \mathbf{V}_{=,\neg\neq}$ for all $\beta \in P_n$, we will have to make use of Lemma 126 (the analogous of Lemma 104 for this case).*

Processing data-aware diamonds of the form $(\neg)\langle \varepsilon * \downarrow[\psi]\alpha\rangle$. Having all these observations at hand, we begin by analyzing the following (non-disjoint) cases to construct our tree $\mathcal{T}^{\varphi}$:

(Case 1) For $(\psi, \alpha) \in \mathbf{V}_{=,\neq}$, we add two witnesses. One for $\langle \varepsilon = \downarrow[\psi]\alpha\rangle$ from which we merge the class of the ending point $x^{\mathbf{v_1}}$ of a pair of nodes satisfying α as in Observation 125 with the class of r^{φ}. We add another witness for $\langle \varepsilon \neq \downarrow[\psi]\alpha\rangle$ (remember Observation 123). See Figure 28(a).

(Case 2) For $(\psi, \alpha) \in \mathbf{V}_{=,\neg\neq}$, we add one witness for $\langle \varepsilon = \downarrow[\psi]\alpha\rangle$ (see Figure 28(b)) and, at the end of the construction, we will merge the class of any node x such that $r^{\varphi}, x \models \downarrow[\psi]\alpha$ with the class of r^{φ} (remember Observation 120).

(Case 3) For $(\psi, \alpha) \in \mathbf{V}_{\neg=,\neq}$, we add one witness for $\langle \varepsilon \neq \downarrow[\psi]\alpha\rangle$ (See Figure 28(c)). Note that $\langle \varepsilon \neq \downarrow[\psi]\alpha\rangle \wedge \neg\langle \varepsilon = \downarrow[\psi]\alpha\rangle$ will be satisfied by Observations 122 and 125.

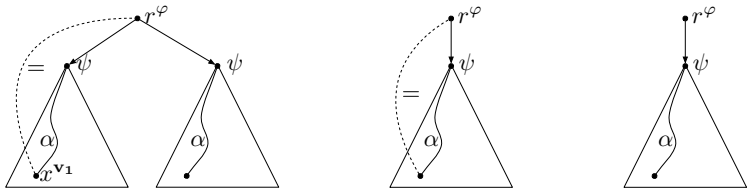


Figure 28: (a) Witnesses for $\langle \varepsilon = \downarrow[\psi]\alpha\rangle$ and $\langle \varepsilon \neq \downarrow[\psi]\alpha\rangle$ for $(\psi, \alpha) \in \mathbf{V}_{=,\neq}$; (b) A witness for $\langle \varepsilon = \downarrow[\psi]\alpha\rangle$ for $(\psi, \alpha) \in \mathbf{V}_{=,\neg\neq}$; (c) A witness for $\langle \varepsilon \neq \downarrow[\psi]\alpha\rangle$ for $(\psi, \alpha) \in \mathbf{V}_{\neg=,\neq}$.

Processing data-aware diamonds of the form $(\neg)\langle \downarrow[\psi]\alpha * \downarrow[\rho]\beta\rangle$. For conjuncts of φ the form $(\neg)\langle \downarrow[\psi]\alpha * \downarrow[\rho]\beta\rangle$ that do not involve comparison with the constant path ε, we have that, depending on which of the sets $\mathbf{V}_{=,\neq}, \mathbf{V}_{=,\neg\neq}, \mathbf{V}_{\neg=,\neq}, \mathbf{V}_{\neg=,\neg\neq}$ do (ψ, α) and (ρ, β) belong to, many of the four possible combinations ($\langle \downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$ and $\langle \downarrow[\psi]\alpha \neq \downarrow[\rho]\beta\rangle$, $\langle \downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$ and $\neg\langle \downarrow[\psi]\alpha \neq \downarrow[\rho]\beta\rangle$, etc.) are not possible as conjuncts for a consistent φ. More specifically:

(Case 4) If we have $\langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$ and $\neg\langle\downarrow[\psi]\alpha \neq \downarrow[\rho]\beta\rangle$ as conjuncts of φ, then all the following cases should be impossible since, in that case, φ would be clearly unsatisfiable and thus it should be inconsistent: (ψ, α) or (ρ, β) in $\mathbf{V}_{=,\neq}$, (ψ, α) or (ρ, β) in $\mathbf{V}_{\neg=,\neg\neq}$, one in $\mathbf{V}_{=,\neg\neq}$ and the other in $\mathbf{V}_{\neg=,\neq}$. Besides, if both belong to $\mathbf{V}_{=,\neg\neq}$, since we merge the class of any node x such that $r^\varphi, x \models \downarrow[\psi]\alpha$ or $r^\varphi, x \models \downarrow[\rho]\beta$, those conjuncts $\langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$ and $\neg\langle\downarrow[\psi]\alpha \neq \downarrow[\rho]\beta\rangle$ will be satisfied. If both belong to $\mathbf{V}_{\neg=,\neq}$, we need to force these conjuncts by merging the class of any node x such that $r^\varphi, x \models \downarrow[\psi]\alpha$ or $r^\varphi, x \models \downarrow[\rho]\beta$ (note that we have such nodes by (Case 3)). It is important to notice that this process does not add nodes to the class of the root since such nodes x are never in the same equivalence class than any $x^{\mathbf{v_1}}$ from (Case 1) nor in the same equivalence class of a witness of $\langle\downarrow[\mu]\delta\rangle$ for $(\mu, \delta) \in \mathbf{V}_{=,\neg\neq}$.

(Case 5) If we have $\langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$ and $\langle\downarrow[\psi]\alpha \neq \downarrow[\rho]\beta\rangle$ as conjuncts of φ, then it cannot be the case that (ψ, α) or (ρ, β) belong to $\mathbf{V}_{\neg=,\neg\neq}$. Neither is possible that both of them belong to $\mathbf{V}_{=,\neg\neq}$ or one to $\mathbf{V}_{=,\neg\neq}$ and the other to $\mathbf{V}_{\neg=,\neq}$. Besides, if $(\psi, \alpha), (\rho, \beta)$ belong to $\mathbf{V}_{=,\neq}$, then $\langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$ and $\langle\downarrow[\psi]\alpha \neq \downarrow[\rho]\beta\rangle$ are already satisfied: $\langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$ by the witnesses for $\langle\varepsilon = \downarrow[\psi]\alpha\rangle$ and $\langle\varepsilon = \downarrow[\rho]\beta\rangle$ (see Figure 29(a)), $\langle\downarrow[\psi]\alpha \neq \downarrow[\rho]\beta\rangle$ by the witnesses for $\langle\varepsilon = \downarrow[\psi]\alpha\rangle$ and $\langle\varepsilon \neq \downarrow[\rho]\beta\rangle$ (see Figure 29(b) and remember Observation 123). In case one belongs to $\mathbf{V}_{=,\neq}$ and the other to $\mathbf{V}_{=,\neg\neq}$, the argument is similar. If one belongs to $\mathbf{V}_{=,\neq}$ and the other to $\mathbf{V}_{\neg=,\neq}$ or both to $\mathbf{V}_{\neg=,\neq}$, $\langle\downarrow[\psi]\alpha \neq \downarrow[\rho]\beta\rangle$ will be satisfied using arguments similar to the previous ones; but we need to add witnesses to guarantee the satisfaction of $\langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$ (see Figure 29(c)). In some cases, the merging performed in (Case 4), would have already merged the classes of a witness for $\langle\downarrow[\psi]\alpha\rangle$ and a witness for $\langle\downarrow[\rho]\beta\rangle$, in the remaining cases, we will need to force that merging carefully enough not to spoil conditions (C6) and (C8) (we will use Lemma 126 to achieve that).

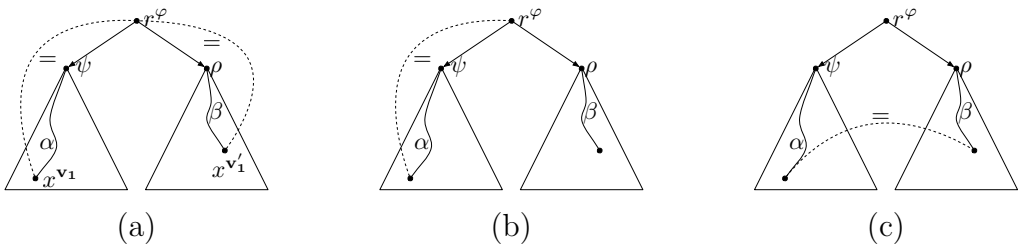


Figure 29: (a) Witnesses for $\langle\varepsilon = \downarrow[\psi]\alpha\rangle$ and $\langle\varepsilon = \downarrow[\rho]\beta\rangle$ for $(\psi, \alpha) \in \mathbf{V}_{=,\neq}, (\rho, \beta) \in \mathbf{V}_{=,\neg\neq}$ end up in the same equivalence class; (b) Witnesses for $\langle\varepsilon = \downarrow[\psi]\alpha\rangle$ for $(\psi, \alpha) \in \mathbf{V}_{=,\neg\neq}$ and $\langle\varepsilon \neq \downarrow[\rho]\beta\rangle$ for $(\rho, \beta) \in \mathbf{V}_{\neg=,\neq}$ end up in different equivalence classes; (c) Witnesses for $\langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$ for $(\psi, \alpha) \in \mathbf{V}_{=,\neq}, (\rho, \beta) \in \mathbf{V}_{\neg=,\neq}$ or $(\psi, \alpha), (\rho, \beta) \in \mathbf{V}_{\neg=,\neq}$.

Finally, these last two cases are satisfied automatically:

(Case 6) If we have $\neg\langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$ and $\langle\downarrow[\psi]\alpha \neq \downarrow[\rho]\beta\rangle$ as conjuncts of φ, then all the following cases should be impossible: (ψ, α) or (ρ, β) in $\mathbf{V}_{\neg=,\neg\neq}$, both in $\mathbf{V}_{=,\neq}$ or $\mathbf{V}_{=,\neg\neq}$. Besides, if one belongs to $\mathbf{V}_{=,\neg\neq}$ and the other to $\mathbf{V}_{\neg=,\neq}$ or if one belongs to $\mathbf{V}_{=,\neq}$ and the other to $\mathbf{V}_{\neg=,\neq}$ or if they both belong to $\mathbf{V}_{\neg=,\neq}$, $\neg\langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$ and $\langle\downarrow[\psi]\alpha \neq \downarrow[\rho]\beta\rangle$ will be satisfied automatically —the last two cases may not be as intuitive as others but are also true and we will give a detailed proof in time.

(Case 7) If we have $\neg\langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$ and $\neg\langle\downarrow[\psi]\alpha \neq \downarrow[\rho]\beta\rangle$ as conjuncts of φ, then the only case that should not lead to an inconsistency is when at least one of (ψ, α) and (ρ, β) is in $\mathbf{V}_{\neg=,\neg\neq}$ and, in this case, $\neg\langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$ and $\neg\langle\downarrow[\psi]\alpha \neq \downarrow[\rho]\beta\rangle$ will be satisfied automatically.

Formalization

In order to formalize the construction described above, we introduce the following lemma, which is key to guarantee conditions (C2) and (C4) without spoiling conditions (C6) and (C8):

Lemma 126. *Let $\psi_0 \in N_n$, $\alpha, \beta_1, \ldots, \beta_m \in P_n$. Suppose that there exists a tree $\mathcal{T}^{\psi_0} = (T^{\psi_0}, \pi^{\psi_0})$ with* root *r^{ψ_0} such that $\mathcal{T}^{\psi_0}, r^{\psi_0} \models \psi_0$ and for all $i = 1, \ldots, m$ there exists $\gamma_i \in P_{n+1}$ such that $\langle\gamma_i = \downarrow[\psi_0]\alpha\rangle \wedge \neg\langle\gamma_i = \downarrow[\psi_0]\beta_i\rangle$ is consistent. Then there exists a tree $\widetilde{\mathcal{T}^{\psi_0}} = (\widetilde{T^{\psi_0}}, \widetilde{\pi^{\psi_0}})$ with* root *r^{ψ_0} and a node x such that:*

- $\widetilde{\mathcal{T}^{\psi_0}}, \widetilde{r^{\psi_0}} \models \psi_0$,
- $\widetilde{\mathcal{T}^{\psi_0}}, \widetilde{r^{\psi_0}}, x \models \alpha$, *and*
- $[x]_{\widetilde{\pi^{\psi_0}}} \neq [y]_{\widetilde{\pi^{\psi_0}}}$ *for all y such that $\widetilde{\mathcal{T}^{\psi_0}}, \widetilde{r^{\psi_0}}, y \models \beta_i$ for some $i = 1, \ldots, m$.*

Proof. Suppose $\alpha = \downarrow[\psi_1] \ldots \downarrow[\psi_{j_0}]\varepsilon$ where $\psi_k \in N_{n-k}$ for all $k = 1, \ldots, j_0$ and let

$$k_0 = \min_{0 \leq k \leq j_0} \{k \mid \neg\langle\downarrow[\psi_{k+1}] \ldots \downarrow[\psi_{j_0}]\varepsilon \neq \downarrow[\psi_{k+1}] \ldots \downarrow[\psi_{j_0}]\varepsilon\rangle \text{ is a conjunct of } \psi_k\}.$$

In case $k_0 = 0$ (i.e. $\neg\langle\alpha \neq \alpha\rangle$), by Lemma 114 , $\neg\langle\alpha = \beta_i\rangle$ is a conjunct of ψ_0 for all $i = 1, \ldots, m$. Then $\widetilde{\mathcal{T}^{\psi_0}} = (T^{\psi_0}, \pi^{\psi_0})$ satisfies the desired properties. The intuitive idea behind this application of Lemma 114 is that in case every ending point of a pair of nodes satisfying α is in the same equivalence class, then there cannot be pairs of nodes satisfying α and β_i ending in points with the same data value, because in that case $\langle\gamma_i = \downarrow[\psi_0]\alpha\rangle \wedge \neg\langle\gamma_i = \downarrow[\psi_0]\beta_i\rangle$ would be unsatisfiable and thus inconsistent, which is a contradiction with our hypothesis.

In case $k_0 \neq 0$, by consistency, there are $z', x' \in T^{\psi_0}$ such that $\mathcal{T}^{\psi_0}, r^{\psi_0}, z' \models \downarrow[\psi_1] \ldots \downarrow[\psi_{k_0}]$ and $\mathcal{T}^{\psi_0}, z', x' \models \downarrow[\psi_{k_0+1}] \ldots \downarrow[\psi_{j_0}]$. Before proceeding to complete the proof of this case, we give an intuitive idea. We prove that we cannot have a witness for β_i with the same

data value than x' in the subtree $T^{\psi_0}|_{z'}$. Intuitively this is because, in that case, α and β_i would have a common prefix. Let us say that

$$\beta = \downarrow[\psi_1]\ldots\downarrow[\psi_{k_0}]\downarrow[\rho_{k_0+1}]\ldots\downarrow[\rho_{l_0}]\varepsilon \quad \text{and} \quad \langle\downarrow[\psi_{k_0+1}]\ldots\downarrow[\psi_{j_0}]\varepsilon = \downarrow[\rho_{k_0+1}]\ldots\downarrow[\rho_{l_0}]\varepsilon\rangle$$

is a conjunct of ψ_{k_0}. Then, since $\neg\langle\downarrow[\psi_{k_0+1}]\ldots\downarrow[\psi_{j_0}]\varepsilon \neq \downarrow[\psi_{k_0+1}]\ldots\downarrow[\psi_{j_0}]\varepsilon\rangle$ is also a conjunct of ψ_{k_0}, $\langle\gamma_i = \downarrow[\psi_0]\alpha\rangle \wedge \neg\langle\gamma_i = \downarrow[\psi_0]\beta_i\rangle$ would be unsatisfiable (and thus inconsistent) for any choice of γ_i, which is a contradiction. But our hypotheses do not guarantee that we would not have a witness for β_i in the class of x' outside $T^{\psi_0}|_{z'}$, and therefore we need to change the tree in order to achieve the desired properties. We replicate the subtree $T^{\psi_0}|_{z'}$ but using a fresh data value (different from any other data value already present in $\mathcal{T}^{\psi_0}$) for the class of the companion of x' that we call x; see Figure 30. It is clear that in this way, the second and the third conditions will be satisfied by x. The first condition will also remain true because, intuitively, the positive conjuncts will remain valid since we are not suppressing any nodes, and the negative ones that compare by equality will not be affected because every new node has either the same data value than its companion or a fresh data value. The argument for negative conjuncts that compare by inequality is based on the way in which we have chosen k_0 (see a detailed proof below).

Now we formalize the previous intuition. Let p be the parent of z' ($k_0 > 0$). As we did in the proof of Lemma 104, we define $\widetilde{\mathcal{T}^{\psi_0}}$ by adding a new child z of p and a data tree $\mathcal{T} = (T, \pi)$ hanging from z. This tree T is a copy of $T^{\psi_0}|_z$, and we call x to the companion of x'. $\widetilde{\pi^{\psi_0}}$ is defined as π^{ψ_0} with the exception that the class of x is new (the classes of the other nodes of T are merged with the classes of their companions) (see Figure 30).

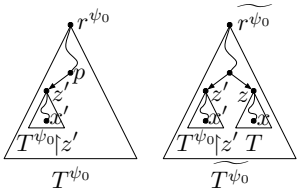


Figure 30: $T = T^{\psi_0}|_z$ is a new subtree with a special node x such that its class of data values is disjoint to the rest of $\widetilde{\mathcal{T}^{\psi_0}}$ and $\widetilde{\mathcal{T}^{\psi_0}}, \widetilde{r^{\psi_0}}, x \models \alpha$.

We first prove by induction that z_j, the j-th ancestor of z (namely $z_j \xrightarrow{j} z$, and we let $z_0 := z$), satisfies $\widetilde{\mathcal{T}^{\psi_0}}, z_j \models \psi_{k_0-j}$. This will prove both that $\widetilde{\mathcal{T}^{\psi_0}}, \widetilde{r^{\psi_0}} \models \psi_0$ and that $\widetilde{\mathcal{T}^{\psi_0}}, \widetilde{r^{\psi_0}}, x \models \alpha$. By Proposition 88, it is straightforward from the construction that ψ_{k_0} is satisfied at z (the companion of z') which proves the base case. For the inductive case, assume the result holds for $z_0, \ldots, z_j$. We want to see that it holds for z_{j+1}. To do this, we check that every conjunct of ψ_{k_0-j-1} is satisfied at z_{j+1}:

- If the conjunct is a label, it is clear that z_{j+1} has that label in $\widetilde{\mathcal{T}^{\psi_0}}$, as it has not been changed by the construction.

- If the conjunct is of the form $\langle \mu_1 = \mu_2 \rangle$ or $\langle \mu_1 \neq \mu_2 \rangle$, then it must still hold in $\widetilde{\mathcal{T}^{\psi_0}}$ by inductive hypothesis and the fact that our construction did not remove nodes.

- If the conjunct is of the form $\neg\langle \mu_1 = \mu_2 \rangle$, we observe that, by inductive hypothesis plus the way in which we have constructed $\widetilde{\mathcal{T}^{\psi_0}}$, we have that: If $\widetilde{\mathcal{T}^{\psi_0}}, z_{j+1} \models \langle \mu_1 = \mu_2 \rangle$ then $\mathcal{T}^{\psi_0}, z_{j+1} \models \langle \mu_1 = \mu_2 \rangle$ (for a complete proof of this assertion, one can use arguments similar to the ones used in Lemma 104) which is a contradiction with the fact that $\mathcal{T}^{\psi_0}, z_{j+1} \models \psi_{k_0-(j+1)}$. Then $\widetilde{\mathcal{T}^{\psi_0}}, z_{j+1} \models \neg\langle \mu_1 = \mu_2 \rangle$.

- If the conjunct is of the form $\neg\langle \mu_1 \neq \mu_2 \rangle$, by inductive hypothesis plus the way in which we have constructed $\widetilde{\mathcal{T}^{\psi_0}}$, $\langle \mu_1 \neq \mu_2 \rangle$ can only be true in z_{j+1} if there are witnesses y_1, y_2 in distinct equivalence classes such that $\widetilde{\mathcal{T}^{\psi_0}}, z_{j+1}, y_1 \models \mu_1$, $\widetilde{\mathcal{T}^{\psi_0}}, z_{j+1}, y_2 \models \mu_2$ and at least one of them is in the new subtree T. In that case, without loss of generality, we have that $\mu_1 = \downarrow[\psi_{k_0-j}] \dots \downarrow[\psi_{k_0}]\hat{\mu_1}$. Then, by definition of k_0, $\langle \downarrow[\psi_{k_0-j}] \dots \downarrow[\psi_{j_0}]\varepsilon \neq \downarrow[\psi_{k_0-j}] \dots \downarrow[\psi_{j_0}]\varepsilon \rangle$ is a conjunct of ψ_{k_0-j-1}. Therefore, by consistency and NeqAx7, $\langle \downarrow[\psi_{k_0-j}] \dots \downarrow[\psi_{j_0}]\varepsilon \neq \mu_2 \rangle$ or $\neg\langle \mu_2 = \mu_2 \rangle$ is also a conjunct of ψ_{k_0-j-1}. If the latter occurs, we have a contradiction by the previous item. If $\langle \downarrow[\psi_{k_0-j}] \dots \downarrow[\psi_{j_0}]\varepsilon \neq \mu_2 \rangle$ is a conjunct of ψ_{k_0-j-1}, by Lemma 114 $\neg\langle \downarrow[\psi_{k_0+1}] \dots \downarrow[\psi_{j_0}]\varepsilon = \hat{\mu_1} \rangle$ is a conjunct of ψ_{k_0}. Then, by construction, the class of y_1 in $\widetilde{\mathcal{T}^{\psi_0}}$ is equal to the class of its companion and so we can assume that $y_1 \notin T$. Analogously we can assume that $y_2 \notin T$ but, as we have already said, by inductive hypothesis plus the way in which we have constructed $\widetilde{\mathcal{T}^{\psi_0}}$, $\langle \mu_1 \neq \mu_2 \rangle$ cannot be satisfied at z_{j+1} by witnesses y_1, y_2 if neither of them is in the new subtree T.

To conclude the proof, we only need to check that $[x]_{\widetilde{\pi^{\psi_0}}} \neq [y]_{\widetilde{\pi^{\psi_0}}}$ for all y such that $\widetilde{\mathcal{T}^{\psi_0}}, \widetilde{r^{\psi_0}}, y \models \beta_i$ for some $i = 1, \dots, m$. Suppose that $\beta_i = \downarrow[\rho_1] \dots \downarrow[\rho_{l_0}]\varepsilon$. If $l_0 < k_0$ or $\rho_l \neq \psi_l$ for some $l = 1, \dots, k_0$, then the result follows immediately from construction. If not, by hypothesis, there exists $\gamma_i \in P_{n+1}$ such that $\langle \gamma_i = \downarrow[\psi_0] \dots \downarrow[\psi_{j_0}]\varepsilon \rangle \wedge \neg\langle \gamma_i = \downarrow[\psi_0] \dots \downarrow[\psi_{k_0}]\downarrow[\rho_{k_0+1}] \dots \downarrow[\rho_{l_0}]\varepsilon \rangle$ is consistent and $\neg\langle \downarrow[\psi_{k_0+1}] \dots \downarrow[\psi_{j_0}]\varepsilon \neq \downarrow[\psi_{k_0+1}] \dots \downarrow[\psi_{j_0}]\varepsilon \rangle$ is a conjunct of ψ_{k_0}. Then, by Lemma 114, $\neg\langle \downarrow[\psi_{k_0+1}] \dots \downarrow[\psi_{j_0}]\varepsilon = \downarrow[\rho_{k_0+1}] \dots \downarrow[\rho_{l_0}]\ \varepsilon \rangle$ is a conjunct of ψ_{k_0}. This together with the fact that the class of x is disjoint with the part of $\widetilde{T^{\psi_0}}$ outside of T, shows that $[x]_{\widetilde{\pi^{\psi_0}}} \neq [y]_{\widetilde{\pi^{\psi_0}}}$ if y is such that $\widetilde{\mathcal{T}^{\psi_0}}, \widetilde{r^{\psi_0}}, y \models \beta_i$, which concludes the proof. □

It might be useful to remark on the differences between Lemmas 126 and 104, as they are one of the reasons why the completeness result for $\mathrm{XPath}_{=}(\downarrow)$ is more complicated than for $\mathrm{XPath}_{=}(\downarrow)^{-}$. The main differences between those two lemmas are:

- In Lemma 126, if we would replicate the subtree hanging from a witness of $\langle \alpha \rangle$ then, due to the fact that we are working with the complete fragment (with inequality tests also), we would not be able to prove that each ancestor of that node satisfies the desired formulas. So we are forced to find that minimum k_0 that tells us which subtree we should replicate.

- In Lemma 104, we can use new data for every new node since, again, we are not working with inequality tests. But when it comes to the complete fragment, we need to be more careful in the way we define the partition in $\widetilde{\mathcal{T}^{\psi_0}}$ changing only the class of the new witness of $\langle\alpha\rangle$.

Now that we have this key lemma, we proceed to the formal construction of $\mathcal{T}^\varphi$. We define some special sets of quadruples $(\psi, \alpha, \rho, \beta)$ with $\psi, \rho \in N_n$, $\alpha, \beta \in P_n$:

- **U** is the set of quadruples $(\psi, \alpha, \rho, \beta)$ such that one of the following holds:
  - $(\psi, \alpha), (\rho, \beta) \in \mathbf{V}_{\neg=,\neq}$, and $\langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$, $\langle\downarrow[\psi]\alpha \neq \downarrow[\rho]\beta\rangle$ are conjuncts of φ, or
  - $(\psi, \alpha) \in \mathbf{V}_{=,\neq}$, $(\rho, \beta) \in \mathbf{V}_{\neg=,\neq}$, and $\langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$, $\langle\downarrow[\psi]\alpha \neq \downarrow[\rho]\beta\rangle$ are conjuncts of φ.

  cf. (Case 5).

- **Z** is the set of all quadruples $(\psi, \alpha, \rho, \beta)$ such that $(\psi, \alpha), (\rho, \beta) \in \mathbf{V}_{\neg=,\neq}$, and $\langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$, $\neg\langle\downarrow[\psi]\alpha \neq \downarrow[\rho]\beta\rangle$ are conjuncts of φ.

  cf. (Case 4).

The following lemma states that the relation between the elements of $\mathbf{V}_{\neg=,\neq}$ defined by the set **Z** is transitive, a fact which will be needed to prove that φ is indeed satisfied in the constructed tree:

Lemma 127. *If* $(\psi, \alpha, \rho, \beta), (\rho, \beta, \theta, \gamma) \in \mathbf{Z}$*, then* $(\psi, \alpha, \theta, \gamma) \in \mathbf{Z}$*.*

Proof. By NeqAx7 and the consistency of φ, $\neg\langle\downarrow[\psi]\alpha \neq \downarrow[\theta]\gamma\rangle$ is a conjunct of φ. Then, by consistency of φ plus NeqAx6, $\langle\downarrow[\psi]\alpha = \downarrow[\theta]\gamma\rangle$ is also a conjunct of φ which concludes the proof. □

Now that we have these lemmas, we proceed to construct $\mathcal{T}^\varphi$ as follows:

Rule 1. Witnesses for $\mathbf{v_1} = (\psi, \alpha) \in \mathbf{V}_{=,\neq}$. (cf. (Case 1)) We define data trees $\mathcal{T}_1^{\mathbf{v_1}} = (T_1^{\mathbf{v_1}}, \pi_1^{\mathbf{v_1}})$ and $\mathcal{T}_2^{\mathbf{v_1}} = (T_2^{\mathbf{v_1}}, \pi_2^{\mathbf{v_1}})$ with roots $r_1^{\mathbf{v_1}}$ and $r_2^{\mathbf{v_1}}$ respectively. In order to choose appropriate witnesses for $\langle\varepsilon = \downarrow[\psi]\alpha\rangle$ and $\langle\varepsilon \neq \downarrow[\psi]\alpha\rangle$, we need the following lemma:

Lemma 128. *Let* $\mathbf{v_1} = (\psi, \alpha) \in \mathbf{V}_{=,\neq}$*. Then there exist* $\widetilde{\mathcal{T}^\psi} = (\widetilde{T^\psi}, \widetilde{\pi^\psi})$ *with* root $\widetilde{r^\psi}$ *and a node* x *such that:*

- $\widetilde{\mathcal{T}^\psi}, \widetilde{r^\psi} \models \psi$,
- $\widetilde{\mathcal{T}^\psi}, \widetilde{r^\psi}, x \models \alpha$,

- $[x]_{\widetilde{\pi^\psi}} = [y]_{\widetilde{\pi^\psi}}$ *for all* y *such that there is* $\beta \in P_n$ *with* $(\psi, \beta) \in \mathbf{V}_{=,\neg\neq}$ *and* $\widetilde{\mathcal{T}^\psi}, \widetilde{r^\psi}, y \models \beta$,
- $[x]_{\widetilde{\pi^\psi}} \neq [z]_{\widetilde{\pi^\psi}}$ *for all* z *such that there is* $\gamma \in P_n$ *with* $(\psi, \gamma) \in \mathbf{V}_{\neg=,\neq}$ *and* $\widetilde{\mathcal{T}^\psi}, \widetilde{r^\psi}, z \models \gamma$.

Proof. We first analyze the case that there exists $\beta \in P_n$ such that $(\psi, \beta) \in \mathbf{V}_{=,\neg\neq}$. Then, by Lemmas 121 and 124, the result is immediate from the fact that we are assuming there is a tree $\mathcal{T}^\psi$ satisfying ψ at its root. The idea is that by inductive hypothesis, there exists $\mathcal{T}^\psi = (T^\psi, \pi^\psi)$ satisfying ψ at is root. Then, Lemma 121 guarantees that every witness of some β as described before belongs to the same class in π^ψ and that every witness of some γ as described before does not belong to this class. Finally, Lemma 124 shows the existence of the desired node x.

To conclude the proof, suppose that $(\psi, \beta) \notin \mathbf{V}_{=,\neg\neq}$ for all $\beta \in P_n$. Then the result follows from Lemma 126. □

Using Lemma 128, define $T_1^{\mathbf{v_1}}$ as $\widetilde{T^\psi}$, $\pi_1^{\mathbf{v_1}}$ as $\widetilde{\pi^\psi}$, $r_1^{\mathbf{v_1}}$ as $\widetilde{r^\psi}$ and $x^{\mathbf{v_1}} = x \in T_1^{\mathbf{v_1}}$. Also, by inductive hypothesis, there exists a tree $\mathcal{T}^\psi = (T^\psi, \pi^\psi)$ with root r^ψ such that $\mathcal{T}^\psi, r^\psi \models \psi$. Define $T_2^{\mathbf{v_1}}$ as T^ψ, $\pi_2^{\mathbf{v_1}}$ as π^ψ and $r_2^{\mathbf{v_1}}$ as r^ψ. Without loss of generality, we assume that $T_1^{\mathbf{v_1}}$ and $T_2^{\mathbf{v_1}}$ are disjoint. In other words, the rooted data tree $(T_1^{\mathbf{v_1}}, \pi_1^{\mathbf{v_1}}, r_1^{\mathbf{v_1}})$ is just a copy of $(\widetilde{T^\psi}, \widetilde{\pi^\psi}, \widetilde{r^\psi})$ with a special node named $x^{\mathbf{v_1}}$ and $(T_2^{\mathbf{v_1}}, \pi_2^{\mathbf{v_1}}, r_2^{\mathbf{v_1}})$ is just a copy of (T^ψ, π^ψ) disjoint with $(T_1^{\mathbf{v_1}}, \pi_1^{\mathbf{v_1}}, r_1^{\mathbf{v_1}})$. See Figure 31(a).

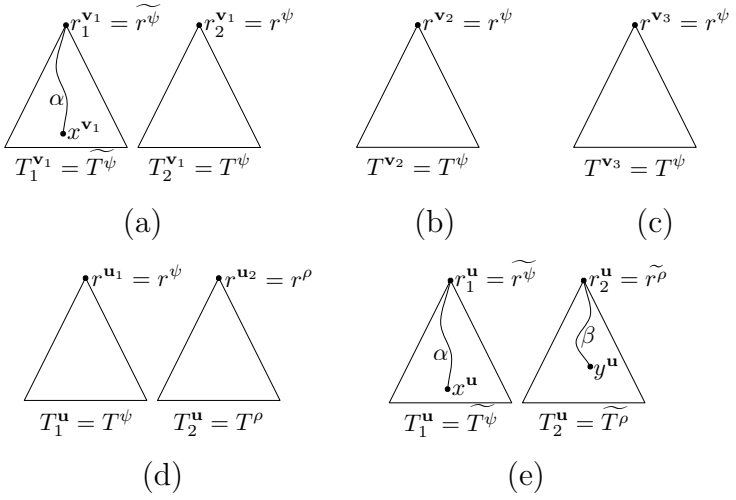


Figure 31: Witnesses for (a) $\mathbf{v_1} = (\psi, \alpha) \in \mathbf{V}_{=,\neq}$; (b) $\mathbf{v_2} = (\psi, \alpha) \in \mathbf{V}_{=,\neg\neq}$; (c) $\mathbf{v_3} = (\psi, \alpha) \in \mathbf{V}_{\neg=,\neq}$; (d) $\mathbf{u} = (\psi, \alpha, \rho, \beta) \in \mathbf{U_1}$; (e) $\mathbf{u} = (\psi, \alpha, \rho, \beta) \in \mathbf{U_2}$.

Rule 2. Witnesses for $\mathbf{v_2} = (\psi, \alpha) \in \mathbf{V}_{=,\neg\neq}$. (cf. (Case 2)) We define a data tree $\mathcal{T}^{\mathbf{v_2}} = (T^{\mathbf{v_2}}, \pi^{\mathbf{v_2}})$ with root $r^{\mathbf{v_2}}$. By inductive hypothesis, there exists $\mathcal{T}^\psi = (T^\psi, \pi^\psi)$, with

root r^ψ such that $\mathcal{T}^\psi, r^\psi \models \psi$. Define $T^{\mathbf{v_2}}$ as T^ψ, $\pi^{\mathbf{v_2}}$ as π^ψ, and $r^{\mathbf{v_2}}$ as r^ψ. In other words, the rooted data tree $(T^{\mathbf{v_2}}, \pi^{\mathbf{v_2}}, r^{\mathbf{v_2}})$ is just a copy of $(T^\psi, \pi^\psi, r^\psi)$. See Figure 31(b).

Rule 3. Witnesses for $\mathbf{v_3} = (\boldsymbol{\psi}, \boldsymbol{\alpha}) \in \mathbf{V}_{\neg=,\neq}$. (cf. (Case 3)) We define a data tree $\mathcal{T}^{\mathbf{v_3}} = (T^{\mathbf{v_3}}, \pi^{\mathbf{v_3}})$ with root $r^{\mathbf{v_3}}$. By inductive hypothesis, there exists $\mathcal{T}^\psi = (T^\psi, \pi^\psi)$, with root r^ψ such that $\mathcal{T}^\psi, r^\psi \models \psi$. Define $T^{\mathbf{v_3}}$ as T^ψ, $\pi^{\mathbf{v_3}}$ as π^ψ, and $r^{\mathbf{v_3}}$ as r^ψ. In other words, the rooted data tree $(T^{\mathbf{v_3}}, \pi^{\mathbf{v_3}}, r^{\mathbf{v_3}})$ is just a copy of $(T^\psi, \pi^\psi, r^\psi)$. See Figure 31(c).

Rule 4. Witnesses for $\boldsymbol{u} = (\boldsymbol{\psi}, \boldsymbol{\alpha}, \boldsymbol{\rho}, \boldsymbol{\beta}) \in \mathbf{U}$. (cf. (Case 5)) We define data trees $\mathcal{T}_1^{\mathbf{u}} = (T_1^{\mathbf{u}}, \pi_1^{\mathbf{u}})$ and $\mathcal{T}_2^{\mathbf{u}} = (T_2^{\mathbf{u}}, \pi_2^{\mathbf{u}})$ with roots $r_1^{\mathbf{u}}$, $r_2^{\mathbf{u}}$ respectively.

By inductive hypothesis, there exist trees $\mathcal{T}^\psi = (T^\psi, \pi^\psi)$ (with root r^ψ) and $\mathcal{T}^\rho = (T^\rho, \pi^\rho)$ (with root r^ρ) such that $\mathcal{T}^\psi, r^\psi \models \psi$ and $\mathcal{T}^\rho, r^\rho \models \rho$.

Now, in order to consider the information given by $\mathbf{U}$ and its interaction with $\mathbf{Z}$, we split $\mathbf{U}$ into two different subsets:

- $\mathbf{U_1}$ is the set of $(\psi, \alpha, \rho, \beta) \in \mathbf{U}$ for which there are $\gamma, \delta \in P_n$ such that:
  - $(\psi, \gamma, \rho, \delta) \in \mathbf{Z}$,
  - $\langle \gamma = \alpha \rangle$ is a conjunct of ψ,
  - $\langle \delta = \beta \rangle$ is a conjunct of ρ.
- $\mathbf{U_2} = \mathbf{U} \setminus \mathbf{U_1}$

For $\mathbf{u} = (\psi, \alpha, \rho, \beta) \in \mathbf{U_1}$, define $T_1^{\mathbf{u}}$ as T^ψ, $\pi_1^{\mathbf{u}}$ as π^ψ, $r_1^{\mathbf{u}}$ as r^ψ and define $T_2^{\mathbf{u}}$ as T^ρ, $\pi_2^{\mathbf{u}}$ as π^ρ, $r_2^{\mathbf{u}}$ as r^ρ. Without loss of generality, we assume that $T_1^{\mathbf{u}}$ and $T_2^{\mathbf{u}}$ are disjoint.

In other words, the rooted data tree $(T_1^{\mathbf{u}}, \pi_1^{\mathbf{u}}, r_1^{\mathbf{u}})$ is just a copy of $(T^\psi, \pi^\psi, r^\psi)$ and the pointed data tree $(T_2^{\mathbf{u}}, \pi_2^{\mathbf{u}}, r_2^{\mathbf{u}})$ is a copy of $(T^\rho, \pi^\rho, r^\rho)$. See Figure 31(d). Note that these are the cases in which the satisfaction of $\langle \downarrow[\psi]\alpha = \downarrow[\rho]\beta \rangle$ will be guaranteed by the merging described in (Case 4).

For $\mathbf{u} = (\psi, \alpha, \rho, \beta) \in \mathbf{U_2}$, in Lemma 126 consider

$$\begin{aligned} \psi_0 &:= \psi \\ \mathcal{T}^{\psi_0} &:= \mathcal{T}^\psi \\ \alpha &:= \alpha \\ \{\beta_1, \ldots, \beta_m\} &:= \{\gamma \in P_n \mid \neg\langle \downarrow[\rho]\beta = \downarrow[\psi]\gamma \rangle \text{ is a conjunct of } \varphi\} \\ \gamma_i &:= \downarrow[\rho]\beta \text{ for all } i = 1, \ldots, m \end{aligned}$$

Then there exist $\widetilde{\mathcal{T}^\psi} = (\widetilde{T^\psi}, \widetilde{\pi^\psi})$ with root $\widetilde{r^\psi}$ and a node x such that:

- $\widetilde{\mathcal{T}^\psi}, \widetilde{r^\psi} \models \psi$,
- $\widetilde{\mathcal{T}^\psi}, \widetilde{r^\psi}, x \models \alpha$

- $[x]_{\widetilde{\pi^\psi}} \neq [y]_{\widetilde{\pi^\psi}}$ for all y such that there is $\gamma \in P_n$ with $\widetilde{\mathcal{T}^\psi}, \widetilde{r^\psi}, y \models \gamma$ and $\neg\langle\downarrow[\rho]\beta = \downarrow[\psi]\gamma\rangle$ is a conjunct of φ.

Define $T_1^{\mathbf{u}}$ as $\widetilde{T^\psi}$, $\pi_1^{\mathbf{u}}$ as $\widetilde{\pi^\psi}$, $r_1^{\mathbf{u}}$ as $\widetilde{r^\psi}$ and $x^{\mathbf{u}} = x \in T_1^{\mathbf{u}}$. Now let

$$\{\mu_1, \ldots, \mu_r\} = \left\{\mu \in P_n \mid \text{ there exists } \ y \in T_1^{\mathbf{u}} \text{ such that } \mathcal{T}_1^{\mathbf{u}}, r_1^{\mathbf{u}}, y \models \mu \text{ and } [y]_{\pi_1^{\mathbf{u}}} = [x^{\mathbf{u}}]_{\pi_1^{\mathbf{u}}}\right\}.$$

Then it follows that $\langle\downarrow[\rho]\beta = \downarrow[\psi]\mu_j\rangle$ is a conjunct of φ for all $j = 1, \ldots, r$.

In Lemma 126, consider

$$\begin{aligned}
\psi_0 &:= \rho \\
\mathcal{T}^{\psi_0} &:= \mathcal{T}^\rho \\
\alpha &:= \beta \\
\{\beta_1, \ldots, \beta_m\} &:= \{\delta \in P_n \mid \exists j = 1, \ldots, r \text{ with } \neg\langle\downarrow[\rho]\delta = \downarrow[\psi]\mu_j\rangle \text{ is a conjunct of } \varphi\} \\
\gamma_i &:= \downarrow[\psi]\mu_j \text{ for } j = 1, \ldots r \text{ such that } \langle\downarrow[\rho]\beta_i = \downarrow[\psi]\mu_j\rangle \text{ is a conjunct of } \varphi
\end{aligned}$$

Then there exist a tree $\widetilde{\mathcal{T}^\rho} = (\widetilde{T^\rho}, \widetilde{\pi^\rho})$ with root $\widetilde{r^\rho}$ and a node y such that

- $\widetilde{\mathcal{T}^\rho}, \widetilde{r^\rho} \models \rho$,
- $\widetilde{\mathcal{T}^\rho}, \widetilde{r^\rho}, y \models \beta$,
- $[y]_{\widetilde{\pi^\rho}} \neq [z]_{\widetilde{\pi^\rho}}$ for all z such that there is $\delta \in P_n$ and $j = 1, \ldots, r$ with $\widetilde{\mathcal{T}^\rho}, \widetilde{r^\rho}, z \models \delta$ and $\neg\langle\downarrow[\rho]\delta = \downarrow[\psi]\mu_j\rangle$ is a conjunct of φ.

Define $T_2^{\mathbf{u}}$ as $\widetilde{T^\rho}$, $\pi_2^{\mathbf{u}}$ as $\widetilde{\pi^\rho}$, $r_2^{\mathbf{u}}$ as $\widetilde{r^\rho}$ and $y^{\mathbf{u}} = y$. Without loss of generality, we assume that $T_1^{\mathbf{u}}$ and $T_2^{\mathbf{u}}$ are disjoint.

In other words, the rooted data tree $(T_1^{\mathbf{u}}, \pi^{\mathbf{u}}|_{T_1^{\mathbf{u}}}, r_1^{\mathbf{u}})$ is just a copy of $(\widetilde{T^\psi}, \widetilde{\pi^\psi}, \widetilde{r^\psi})$, with a special node named $x^{\mathbf{u}}$ which satisfies $\mathcal{T}_1^{\mathbf{u}}, r_1^{\mathbf{u}}, x^{\mathbf{u}} \models \alpha$. Analogously, the pointed data tree $(T_2^{\mathbf{u}}, \pi^{\mathbf{u}}|_{T_2^{\mathbf{u}}}, r_2^{\mathbf{u}})$ is a copy of $(\widetilde{T^\rho}, \widetilde{\pi^\rho}, \widetilde{r^\rho})$, with a special node named $y^{\mathbf{u}}$ which satisfies $\mathcal{T}_2^{\mathbf{u}}, r_2^{\mathbf{u}}, y^{\mathbf{u}} \models \beta$. See Figure 31(e).

Notice that this rule differs from Rule 2 of §2.3.3 in the fact that we do not merge the classes of $x^{\mathbf{u}}$ and $y^{\mathbf{u}}$ yet. We will perform that merging only at the end of the construction. This is not really important and we could have merged the classes at this step; the reason for doing it at the end is only a technical issue. The proof of Fact 134 will be easier to understand this way.

The following remark will be used later to prove that φ is indeed satisfied in the constructed tree. Its proof is omitted since it is analogous to the proof of Remark 105:

Remark 129. *Let $(\psi, \alpha, \rho, \beta) \in \mathbf{U_2}$. If $\neg\langle\downarrow[\psi]\mu = \downarrow[\rho]\delta\rangle$ is a conjunct of φ, then $[y^{\mathbf{u}}]_{\pi_2^{\mathbf{u}}} \neq [y]_{\pi_2^{\mathbf{u}}}$ for all y such that $\mathcal{T}_2^{\mathbf{u}}, r_2^{\mathbf{u}}, y \models \delta$ or $[x^{\mathbf{u}}]_{\pi_1^{\mathbf{u}}} \neq [x]_{\pi_1^{\mathbf{u}}}$ for all x such that $\mathcal{T}_1^{\mathbf{u}}, r_1^{\mathbf{u}}, x \models \mu$.* □

The rooted data tree $(T^\varphi, \pi^\varphi, r^\varphi)$. As shown in Figure 32, now we define T^φ, using our Rules, as the tree which consists of a root r^φ with label $a \in \mathbb{A}$ if a is a conjunct of φ, and with children

$$(T_1^{\mathbf{v_1}})_{\mathbf{v_1}\in\mathbf{V}_{=,\neq}}, (T_2^{\mathbf{v_1}})_{\mathbf{v_1}\in\mathbf{V}_{=,\neq}}, (T^{\mathbf{v_2}})_{\mathbf{v_2}\in\mathbf{V}_{=,\neg\neq}}, (T^{\mathbf{v_3}})_{\mathbf{v_3}\in\mathbf{V}_{\neg=,\neq}}, (T_1^{\mathbf{u}})_{\mathbf{u}\in\mathbf{U}}, (T_2^{\mathbf{u}})_{\mathbf{u}\in\mathbf{U}}.$$

As a first step we provisionally define $\widetilde{\pi^\varphi}$ over T^φ by

$$\widetilde{\pi^\varphi} = \{\{r^\varphi\}\} \cup \bigcup_{\mathbf{v_1}\in\mathbf{V}_{=,\neq}} (\pi_1^{\mathbf{v_1}} \cup \pi_2^{\mathbf{v_1}}) \cup \bigcup_{\mathbf{v_2}\in\mathbf{V}_{=,\neg\neq}} \pi^{\mathbf{v_2}} \cup \bigcup_{\mathbf{v_3}\in\mathbf{V}_{\neg=,\neq}} \pi^{\mathbf{v_3}} \cup \bigcup_{\mathbf{u}\in\mathbf{U}} (\pi_1^{\mathbf{u}} \cup \pi_2^{\mathbf{u}})$$

It is important to notice that, up to this point in the construction, the tree hanging from

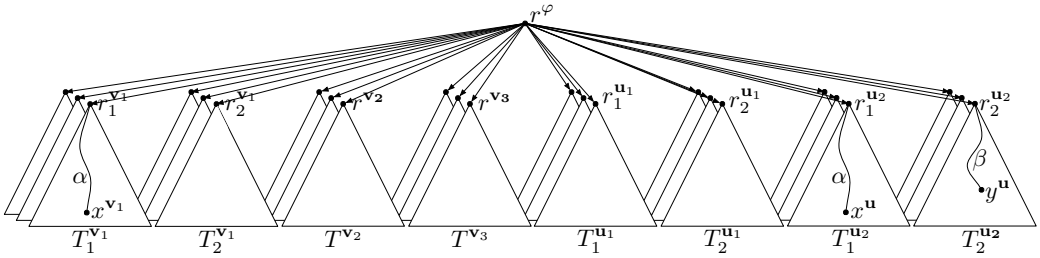


Figure 32: The tree T^φ (without any partition yet).

each child of the root preserves its original partition.

In order to consider the information given by $\mathbf{Z}$ (cf. (Case 4)), we split $\mathbf{V}_{\neg=,\neq}$ into two subsets:

$$\mathbf{V}'_{\neg=,\neq} = \{(\psi,\alpha) \in \mathbf{V}_{\neg=,\neq} \mid \text{ for all } (\rho,\beta) \in \mathbf{V}_{\neg=,\neq},\ (\psi,\alpha,\rho,\beta) \notin \mathbf{Z}\},$$
$$\mathbf{V}''_{\neg=,\neq} = \mathbf{V}_{\neg=,\neq} \setminus \mathbf{V}'_{\neg=,\neq}.$$

The following property of the set $\mathbf{V}''_{\neg=,\neq}$ will be used to prove that φ is indeed satisfied at the constructed tree:

Lemma 130. *Let $(\theta,\delta),(\theta,\delta') \in \mathbf{V}_{\neg=,\neq}$. Suppose that $(\theta,\delta) \in \mathbf{V}''_{\neg=,\neq}$ and $\neg\langle\delta' \neq \delta'\rangle$ and $\langle\delta = \delta'\rangle$ are conjuncts of θ. Then $(\theta,\delta,\theta,\delta') \in \mathbf{Z}$.*

Proof. By NeqAx7, $\neg\langle\downarrow[\theta]\delta \neq \downarrow[\theta]\delta\rangle$ is a conjunct of φ. By EqAx5 plus **Der21** of Fact 90, $\langle\downarrow[\theta]\delta = \downarrow[\theta]\delta'\rangle$ is a conjunct of φ. If we suppose that $\langle\downarrow[\theta]\delta' \neq \downarrow[\theta]\delta'\rangle$ is a conjunct of φ, by Lemma 115 , we have that $\neg\langle\delta = \delta'\rangle$ is a conjunct of θ which is a contradiction. Then we can assume that $\neg\langle\downarrow[\theta]\delta' \neq \downarrow[\theta]\delta'\rangle$ is a conjunct of φ and so we can conclude from NeqAx7 that $\neg\langle\downarrow[\theta]\delta \neq \downarrow[\theta]\delta'\rangle$ is a conjunct of φ. Then we have that $(\theta,\delta,\theta,\delta') \in \mathbf{Z}$. □

As a particular case of Lemma 130, we have:

Remark 131. *Let* $(\theta,\delta),(\theta,\delta') \in \mathbf{V}_{\neg=,\neq}$. *Suppose that* $(\theta,\delta),(\theta,\delta') \in \mathbf{V}''_{\neg=,\neq}$ *and* $\langle \delta = \delta' \rangle$ *is a conjunct of* θ. *Then* $(\theta,\delta,\theta,\delta') \in \mathbf{Z}$.

Proof. Use NeqAx5 plus **Der21** of Fact 90 and NeqAx7. □

We classify the elements of $\mathbf{V}''_{\neg=,\neq}$ according to the following equivalence relation:

$$[(\psi,\alpha)] = [(\rho,\beta)] \quad \text{iff} \quad (\psi,\alpha,\rho,\beta) \in \mathbf{Z}.$$

Observe that this relation is reflexive by NeqAx7, it is clearly symmetric and it is transitive by Lemma 127. We name the equivalence classes $A_1,\ldots,A_m$. We define $\widehat{\pi^\varphi}$ over T^φ taking into account the information given by $\mathbf{V}_{=,\neq}$, $\mathbf{V}_{=,\neg\neq}$ and $\mathbf{Z}$. $\widehat{\pi^\varphi}$ is the smallest equivalence relation containing $\widetilde{\pi^\varphi}$ such that:

- $[x^{\mathbf{v_1}}]_{\widehat{\pi^\varphi}} = [r^\varphi]_{\widehat{\pi^\varphi}}$ for all $\mathbf{v_1} \in \mathbf{V}_{=,\neq}$,
- $[x]_{\widehat{\pi^\varphi}} = [r^\varphi]_{\widehat{\pi^\varphi}}$ for all $x \in M$,
- For all $i = 1,\ldots,m$ $[x]_{\widehat{\pi^\varphi}} = [y]_{\widehat{\pi^\varphi}}$ for all $x,y \in L_i$

where

$$\begin{aligned} M = \{x \mid\ & \text{there exists } (\psi,\alpha) \in \mathbf{V}_{=,\neg\neq} \text{ and a child } z \text{ of } r^\varphi \text{ such that} \\ & T^\varphi, \widetilde{\pi^\varphi}, z \models \psi \text{ and } T^\varphi, \widetilde{\pi^\varphi}, z, x \models \alpha\} \\ L_i = \{x \mid\ & \text{there exists } (\psi,\alpha) \in A_i \text{ and a child } z \text{ of } r^\varphi \text{ such that} \\ & T^\varphi, \widetilde{\pi^\varphi}, z \models \psi \text{ and } T^\varphi, \widetilde{\pi^\varphi}, z, x \models \alpha\} \end{aligned}$$

for all $i = 1,\ldots,m$.

In the previous "gluing", we forced our model to satisfy all diamonds of the form $\langle \varepsilon = \downarrow[\psi]\alpha\rangle$, $\neg\langle \varepsilon \neq \downarrow[\psi]\alpha\rangle$ and $\neg\langle \downarrow[\psi]\alpha \neq \downarrow[\rho]\beta\rangle$ that need to be forced.

It is important to notice that, up to here, the tree hanging from each child of the root still preserves its partition:

Fact 132. *The partition restricted to the trees* $T_1^{\mathbf{v_1}}$, $T_2^{\mathbf{v_1}}$ *for* $\mathbf{v_1} \in \mathbf{V}_{=,\neq}$, *the partition restricted to the trees* $T^{\mathbf{v_2}}$ *for* $\mathbf{v_2} \in \mathbf{V}_{=,\neg\neq}$, *the partition restricted to the trees* $T^{\mathbf{v_3}}$ *for* $\mathbf{v_3} \in \mathbf{V}_{\neg=,\neq}$ *and the partition restricted to the trees* $T_1^{\mathbf{u}}$ *and* $T_2^{\mathbf{u}}$ *for* $\mathbf{u} \in \mathbf{U}$ *remain unchanged. More formally:*

- *For each* $\mathbf{v_1} = (\psi,\alpha) \in \mathbf{V}_{=,\neq}$ *and* $i \in \{1,2\}$, *we have* $\widehat{\pi^\varphi}|_{T_i^{\mathbf{v_1}}} = \pi_i^{\mathbf{v_1}}$.
- *For each* $\mathbf{v_2} = (\psi,\alpha) \in \mathbf{V}_{=,\neg\neq}$, *we have* $\widehat{\pi^\varphi}|_{T^{\mathbf{v_2}}} = \pi^{\mathbf{v_2}}$.
- *For each* $\mathbf{v_3} = (\psi,\alpha) \in \mathbf{V}_{\neg=,\neq}$, *we have* $\widehat{\pi^\varphi}|_{T^{\mathbf{v_3}}} = \pi^{\mathbf{v_3}}$.
- *For each* $\mathbf{u} = (\psi,\alpha,\rho,\beta) \in \mathbf{U}$ *and* $i \in \{1,2\}$, *we have* $\widehat{\pi^\varphi}|_{T_i^{\mathbf{u}}} = \pi_i^{\mathbf{u}}$.

Proof. We give a sketch of the proof, omitting the details.If we think we have three kinds of "gluings", root$_{=,\neq}$-kind, root$_{=,\neg\neq}$-kind and Z-kind, then the way in which two equivalence classes in the same subtree can (hypothetically) be glued together is by a sequence of these gluings. The examples displayed in Figure 33 shows that in (a), the classes of nodes x and y were glued together by a sequence of the form root$_{=,\neq}$-root$_{=,\neg\neq}$; in (b), the classes of nodes x and y were glued together by a sequence of the form root$_{=,\neq}$-root$_{=,\neg\neq}$-Z.

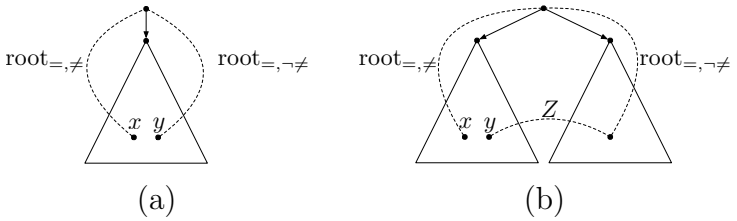


Figure 33: Examples of (hypothetical) "gluings".

We give a list of the ingredients for the complete proof.

- By Rule 1, every witness for $\langle\downarrow[\psi]\alpha\rangle$ with $(\psi,\alpha)\in\mathbf{V}_{\neg=,\neq}$ in $\mathcal{T}_1^{\mathbf{v_1}}$ is in a different class (according to $\widetilde{\pi^{\varphi}}$) than $x^{\mathbf{v_1}}$ for all $\mathbf{v_1}\in\mathbf{V}_{=,\neq}$. Thus we do not have sequences containing root$_{=,\neq}$-Z or Z-root$_{=,\neq}$.

- Lemma 121 implies that every witness for $\langle\downarrow[\psi]\alpha\rangle$ with $(\psi,\alpha)\in\mathbf{V}_{=,\neg\neq}$ and every witness for $\langle\downarrow[\psi]\beta\rangle$ with $(\psi,\beta)\in\mathbf{V}_{\neg=,\neq}$ in the same subtree belong to different classes in that subtree. As a particular case, every $x\in M$ and $y\in L_i$ in the same subtree belong to different classes. Thus we do not have sequences containing root$_{=,\neg\neq}$-Z or Z-root$_{=,\neg\neq}$.

- Since we use a different copy at each application of Rule 1, we do not have sequences starting and ending with root$_{=,\neq}$.

- Lemma 121 implies that every witness for $\langle\downarrow[\psi]\alpha\rangle$ with $(\psi,\alpha)\in\mathbf{V}_{=,\neg\neq}$ in the same subtree belong to the same equivalence class in that subtree. Thus we do not have to worry about sequences starting and ending with root$_{=,\neg\neq}$ because this kind of sequences do not glue different classes.

- By Rule 1, every $x^{\mathbf{v_1}}$ is in the same class that every witness in the same subtree of $\langle\downarrow[\psi]\alpha\rangle$ with $(\psi,\alpha)\in\mathbf{V}_{=,\neg\neq}$. Thus we do not have to worry about sequences starting with root$_{=,\neq}$ and ending with root$_{=,\neg\neq}$ (or vice versa) because this kind of sequences do not glue different classes.

  Combining the previous items, it only remains to consider sequences of only Z-kind gluings.

- If $x, x' \in L_i$ in the same subtree, a very simple derivation involving NeqAx5, shows that $[x]_{\widetilde{\pi^\varphi}} = [x']_{\widetilde{\pi^\varphi}}$. Thus we do not have to worry about sequences of the form Z (just one Z-kind gluing).
- By Lemma 133 below, we do not have to worry about longer sequences of all Z-kind gluings.

This concludes the proof of the Fact. □

Lemma 133. *Let* $\psi, \theta_0, \ldots, \theta_m \in N_n$, $\alpha, \beta, \delta_0, \delta'_0, \ldots, \delta_m, \delta'_m \in P_n$, $x, x', y, y' \in T^\psi, x_0, y_0 \in T^{\theta_0}, \ldots, x_m, y_m \in T^{\theta_m}$ *such that (see Figure 34):*

- $[x]_{\pi^\psi} = [x']_{\pi^\psi}$, $[y]_{\pi^\psi} = [y']_{\pi^\psi}$, $T^\psi, r^\psi, x' \models \alpha$, $T^\psi, r^\psi, y' \models \beta$,
- $[x_i]_{\pi^{\theta_i}} = [y_i]_{\pi^{\theta_i}}$, $T^{\theta_i}, r^{\theta_i}, x_i \models \delta_i$, $T^{\theta_i}, r^{\theta_i}, y_i \models \delta'_i$ *for* $i = 0 \ldots m$, *and*
- $(\theta_0, \delta_0, \psi, \alpha) \in \mathbf{Z}$, $(\theta_i, \delta_i, \theta_{i-1}, \delta'_{i-1}) \in \mathbf{Z}$ *for* $i = 1 \ldots m$, $(\theta_m, \delta'_m, \psi, \beta) \in \mathbf{Z}$.

Then $[x]_{\pi^\psi} = [y]_{\pi^\psi}$.

(Notation: For $\rho \in N_n$, *we use* $\mathcal{T}^\rho = (T^\rho, \pi^\rho)$ *with root* r^ρ *to denote* any *tree in which* ρ *is satisfiable, namely the one given by inductive hypothesis, or the modified one* $\widetilde{\mathcal{T}^\rho}$.*)*

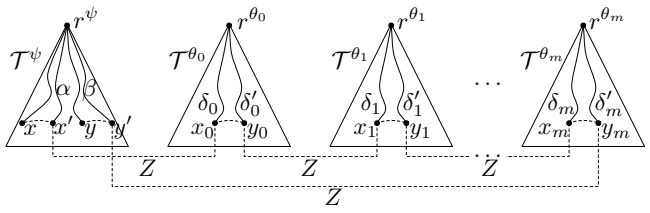


Figure 34: The hypothesis of Lemma 133.

Proof. Observe that, by Lemma 130 plus Lemma 127, $(\psi, \alpha, \psi, \beta) \in \mathbf{Z}$. Then, by NeqAx7 plus Lemma 121, $\neg\langle \alpha \neq \beta \rangle$ is a conjunct of ψ and so $[x]_{\pi^\psi} = [y]_{\pi^\psi}$. □

Finally, define π^φ over T^φ by

$$\pi^\varphi \;=\; \big(\widehat{\pi^\varphi} \setminus (\{[x^{\mathbf{u}}]_{\widehat{\pi^\varphi}}\}_{\mathbf{u}\in\mathbf{U_2}} \cup \{[y^{\mathbf{u}}]_{\widehat{\pi^\varphi}}\}_{\mathbf{u}\in\mathbf{U_2}})\big) \cup \bigcup_{\mathbf{u}\in\mathbf{U_2}} \{[x^{\mathbf{u}}]_{\widehat{\pi^\varphi}} \cup [y^{\mathbf{u}}]_{\widehat{\pi^\varphi}}\}.$$

In other words, T^φ has a root, named r^φ, and children

$$(T_1^{\mathbf{v_1}})_{\mathbf{v_1}\in\mathbf{V}_{=,\neq}}, (T_2^{\mathbf{v_1}})_{\mathbf{v_1}\in\mathbf{V}_{=,\neq}}, (T^{\mathbf{v_2}})_{\mathbf{v_2}\in\mathbf{V}_{=,\neg\neq}}, (T^{\mathbf{v_3}})_{\mathbf{v_3}\in\mathbf{V}_{\neg=,\neq}}, (T_1^{\mathbf{u}})_{\mathbf{u}\in\mathbf{U}}, (T_2^{\mathbf{u}})_{\mathbf{u}\in\mathbf{U}}.$$

Each of these children is the root of its corresponding tree inside T^φ as defined above. All these subtrees are disjoint, and π^φ is defined as the disjoint union of the partitions *with the exception* that we put into the same class:

- the nodes r^φ, $(x^{\mathbf{v_1}})_{\mathbf{v_1}\in\mathbf{V}_{=,\neq}}$ and every witness of $\langle\downarrow[\psi]\alpha\rangle$ with $(\psi,\alpha)\in\mathbf{V}_{=,\neg\neq}$,
- a witness for $\langle\downarrow[\psi]\alpha\rangle$ and a witness for $\langle\downarrow[\rho]\beta\rangle$ if $(\psi,\alpha,\rho,\beta)\in\mathbf{U_2}$,
- every pair of witnesses of $\langle\downarrow[\psi]\alpha\rangle$ and $\langle\downarrow[\rho]\beta\rangle$ respectively with $(\psi,\alpha,\rho,\beta)\in\mathbf{Z}$.

In the previous gluing, we forced our model to satisfy all diamonds of the form $\langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$ that need to be forced.

The following Fact is key to prove that φ is satisfied in $\mathcal{T}^\varphi$:

Fact 134. *The partition restricted to the trees $T_1^{\mathbf{v_1}}$, $T_2^{\mathbf{v_1}}$ for $\mathbf{v_1}\in\mathbf{V}_{=,\neq}$, the partition restricted to the trees $T^{\mathbf{v_2}}$ for $\mathbf{v_2}\in\mathbf{V}_{=,\neg\neq}$, the partition restricted to the trees $T^{\mathbf{v_3}}$ for $\mathbf{v_3}\in\mathbf{V}_{\neg=,\neq}$ and the partition restricted to the trees $T_1^{\mathbf{u}}$ and $T_2^{\mathbf{u}}$ for $\mathbf{u}\in\mathbf{U}$ remain unchanged. More formally:*

- *For each $\mathbf{v_1}=(\psi,\alpha)\in\mathbf{V}_{=,\neq}$ and $i\in\{1,2\}$, we have $\pi^\varphi|_{T_i^{\mathbf{v_1}}}=\pi_i^{\mathbf{v_1}}$.*
- *For each $\mathbf{v_2}=(\psi,\alpha)\in\mathbf{V}_{=,\neg\neq}$, we have $\pi^\varphi|_{T^{\mathbf{v_2}}}=\pi^{\mathbf{v_2}}$.*
- *For each $\mathbf{v_3}=(\psi,\alpha)\in\mathbf{V}_{\neg=,\neq}$, we have $\pi^\varphi|_{T^{\mathbf{v_3}}}=\pi^{\mathbf{v_3}}$.*
- *For each $\mathbf{u}=(\psi,\alpha,\rho,\beta)\in\mathbf{U}$ and $i\in\{1,2\}$, we have $\pi^\varphi|_{T_i^{\mathbf{u}}}=\pi_i^{\mathbf{u}}$.*

Proof. We give a guide for the proof, omitting the details.

Now think that we have four kinds of "gluings", root$_{=,\neq}$-kind, root$_{=,\neg\neq}$-kind, Z-kind and U_2-kind, then the way in which two equivalence classes in the same subtree can (hypothetically) be glued together is by a sequence of these gluings. In the example displayed in Figure 35 the classes of nodes x and y were glued together by a sequence of the form Z-Z-U_2.

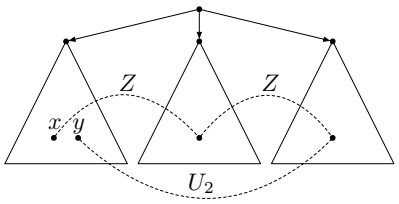


Figure 35: Example of (hypothetical) "gluing".

We give a list of the ingredients for the complete proof.

- We have already observed that the same assertions hold if we change π^φ for $\widehat{\pi^\varphi}$ so we are only interested in sequences that involve some gluing of kind U_2. Moreover, we can assume all the observations made in the proof of Fact 132.

- The fact that $x^{\mathbf{v_1}}$ and $x^{\mathbf{u}}$ (or $y^{\mathbf{u}}$) are always in different subtrees tells us that we do not have sequences containing root$_{=,\neq}$-U_2 or U_2-root$_{=,\neq}$.
- Lemma 121 implies that every witness for $\langle\downarrow[\psi]\alpha\rangle$ with $(\psi,\alpha)\in\mathbf{V}_{=,\neg\neq}$ and every witness for $\langle\downarrow[\psi]\beta\rangle$ with $(\psi,\beta)\in\mathbf{V}_{\neg=,\neq}$ in the same subtree belong to different classes in that subtree. Thus we do not have sequences containing root$_{=,\neg\neq}$-U_2 or U_2-root$_{=,\neg\neq}$ coming from $\mathbf{u}=(\psi,\alpha,\rho,\beta)\in U_2$ with $(\psi,\alpha),(\rho,\beta)\in\mathbf{V}_{\neg=,\neq}$. Besides, suppose that we have one of those sequences coming from $\mathbf{u}=(\psi,\alpha,\rho,\beta)\in U_2$ with $(\psi,\alpha)\in\mathbf{V}_{=,\neq},(\rho,\beta)\in\mathbf{V}_{\neg=,\neq}$ (the symmetric case is analogous) and $(\psi,\mu)\in\mathbf{V}_{=,\neg\neq}$. Then, by the consistency of φ plus NeqAx6, we can conclude that $\neg\langle\downarrow[\psi]\mu=\downarrow[\rho]\beta\rangle$ is a conjunct of φ. This gives us a contradiction by Remark 129. Thus we do not have sequences containing root$_{=,\neg\neq}$-U_2 or U_2- emphroot$_{=,\neg\neq}$ at all.
- By Lemma 130 plus Lemma 127, we can reduce sequences with two consecutive Z-kind gluings to sequences not having two consecutive Z-kind gluings.
- Since we use new subtrees for each $\mathbf{u}\in\mathbf{U_2}$, we cannot have sequences containing U_2-U_2 neither sequences starting and ending with U_2.
- By Lemma 135 below, we cannot have sequences that alternate Z-kind gluings with U_2-kind gluings.
- One can think that the gluing of the classes $[x^{\mathbf{u}}]_{\widehat{\pi^\varphi}}$ and $[y^{\mathbf{u}}]_{\widehat{\pi^\varphi}}$ is made one at a time since they are finite.

This concludes the proof of the Fact. □

Lemma 135. *Let $\psi,\theta_0,\ldots,\theta_m\in N_n$, $\alpha,\beta,\delta_0,\delta_0',\ldots,\delta_m,\delta_m'\in P_n$, $x,x',y,y'\in T^\psi, x_0,y_0\in T^{\theta_0},\ldots,\ x_m,y_m\in T^{\theta_m}$. The following conditions (see Figure 36) cannot be satisfied all at the same time:*

- $[x]_{\pi^\psi}=[x']_{\pi^\psi}$, $[y]_{\pi^\psi}=[y']_{\pi^\psi}$, $T^\psi,r^\psi,x'\models\alpha$, $T^\psi,r^\psi,y'\models\beta$,
- $[x_i]_{\pi^{\theta_i}}=[y_i]_{\pi^{\theta_i}}$, $T^{\theta_i},r^{\theta_i},x_i\models\delta_i$, $T^{\theta_i},r^{\theta_i},y_i\models\delta_i'$ *for* $i=0\ldots m$,
- $(\theta_0,\delta_0,\psi,\alpha)\in\mathbf{Z}$,
- *for* $i=1\ldots m$, $(\theta_i,\delta_i,\theta_{i-1},\delta_{i-1}')\in\begin{cases}\mathbf{U_2} & \text{if } i \text{ is odd}\\ \mathbf{Z} & \text{otherwise}\end{cases}$
- $(\theta_m,\delta_m',\psi,\beta)\in\begin{cases}\mathbf{Z} & \text{if } m \text{ is odd}\\ \mathbf{U_2} & \text{otherwise}\end{cases}$

(Notation: For $\rho\in N_n$, we use $\mathcal{T}^\rho=(T^\rho,\pi^\rho)$ with root r^ρ to denote any *tree in which ρ is satisfiable, namely the one given by inductive hypothesis, or the modified one $\widetilde{\mathcal{T}^\rho}$.)*

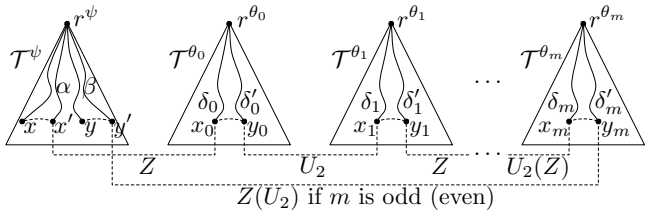


Figure 36: The hypothesis of Lemma 135.

Proof. We proceed by induction on m:

- Case $m = 0$ (see Figure 37(a)):

  Since $(\psi, \beta, \theta_0, \delta'_0) \in \mathbf{U_2}$, $(\psi, \alpha, \theta_0, \delta_0) \in \mathbf{Z}$ and $\langle \delta_0 = \delta'_0 \rangle$ is a conjunct of θ_0, we have that $\neg\langle \alpha = \beta \rangle$ is a conjunct of ψ. But, on the other hand, by Remark 129, we know that $\langle \downarrow[\psi]\beta = \downarrow[\theta_0]\delta_0 \rangle$ is a conjunct of φ which implies, by Lemma 124, that $\langle \alpha = \beta \rangle$ is a conjunct of ψ, a contradiction.

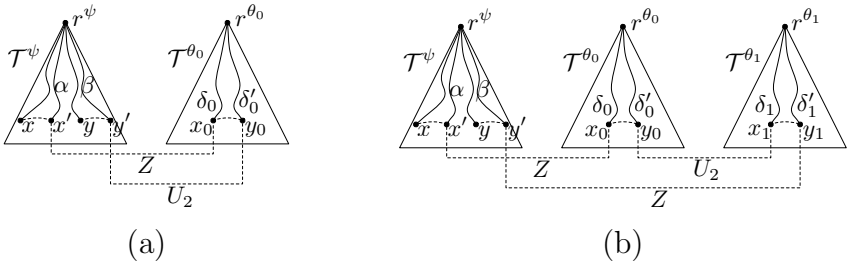


Figure 37: Proof of Lemma 135. (a) case $m = 0$. (b) case $m = 1$.

- If $m = 1$ (see Figure 37(b)):

  By Remark 129, $\langle \downarrow[\theta_0]\delta_0 = \downarrow[\theta_1]\delta'_1 \rangle$ is a conjunct of φ and then, by NeqAx7, $(\theta_0, \delta_0, \theta_1, \delta'_1) \in \mathbf{Z}$. This gives a contradiction with the fact that $(\theta_0, \delta'_0, \theta_1, \delta_1) \in \mathbf{U_2}$ plus the fact that $\langle \delta_0 = \delta'_0 \rangle$ is a conjunct of θ_0 and $\langle \delta_1 = \delta'_1 \rangle$ is a conjunct of θ_1.

- For the induction, suppose $m \geq 2$:

  In case m is odd, by Remark 129, $\langle \downarrow[\theta_{m-1}]\delta_{m-1} = \downarrow[\theta_m]\delta'_m \rangle$ is a conjunct of φ and then, by NeqAx7, $(\theta_{m-1}, \delta_{m-1}, \theta_m, \delta'_m) \in \mathbf{Z}$. By Lemma 127, $(\psi, \beta, \theta_{m-2}, \delta'_{m-2}) \in \mathbf{Z}$ and the result follows from inductive hypothesis for $m - 2$.

  In case m is even, by Remark 129, $\langle \downarrow[\theta_0]\delta_0 = \downarrow[\theta_1]\delta'_1 \rangle$ is a conjunct of φ and then, by NeqAx7 plus NeqAx7, $(\theta_0, \delta_0, \theta_1, \delta'_1) \in \mathbf{Z}$. By Lemma 127, $(\psi, \alpha, \theta_2, \delta_2) \in \mathbf{Z}$ and the result follows from inductive hypothesis for $m - 2$.

This concludes the proof. □

We conclude from Proposition 88 and the construction that:

Fact 136. *The validity of a formula in a child of r^φ is preserved in $\mathcal{T}^\varphi$. More formally:*

- *For each $\mathbf{v_1} \in \mathbf{V}_{=,\neq}$, $i \in \{1,2\}$ and $x, y \in T_i^{\mathbf{v_1}}$ we have $\mathcal{T}^\varphi, x \equiv^\downarrow \mathcal{T}_i^{\mathbf{v_1}}, x$ and $\mathcal{T}^\varphi, x, y \equiv^\downarrow \mathcal{T}_i^{\mathbf{v_1}}, x, y$.*
- *For each $\mathbf{v_2} \in \mathbf{V}_{=,\neg\neq}$ and $x, y \in T^{\mathbf{v_2}}$ we have $\mathcal{T}^\varphi, x \equiv^\downarrow \mathcal{T}^{\mathbf{v_2}}, x$ and $\mathcal{T}^\varphi, x, y \equiv^\downarrow \mathcal{T}^{\mathbf{v_2}}, x, y$.*
- *For each $\mathbf{v_3} \in \mathbf{V}_{\neg=,\neq}$ and $x, y \in T^{\mathbf{v_3}}$ we have $\mathcal{T}^\varphi, x \equiv^\downarrow \mathcal{T}^{\mathbf{v_3}}, x$ and $\mathcal{T}^\varphi, x, y \equiv^\downarrow \mathcal{T}^{\mathbf{v_3}}, x, y$.*
- *For each $\mathbf{u} \in \mathbf{U}$, $i \in \{1,2\}$ and $x, y \in T_i^{\mathbf{u}}$ we have $\mathcal{T}^\varphi, x \equiv^\downarrow \mathcal{T}_i^{\mathbf{u}}, x$ and $\mathcal{T}^\varphi, x, y \equiv^\downarrow \mathcal{T}_i^{\mathbf{u}}, x, y$.*

It only remains to prove that the conditions (C1) – (C9) from the beginning of §2.4.3 are satisfied in the tree we have constructed:

Verification of (C1). This condition is trivially satisfied.

Verification of (C2). Suppose $\langle \varepsilon = \downarrow[\psi]\alpha\rangle$ is a conjunct of φ. Then there are two possibilities, $(\psi, \alpha) \in \mathbf{V}_{=,\neq}$ or $(\psi, \alpha) \in \mathbf{V}_{=,\neg\neq}$.

- In the first case, by Rule 1 and construction, there exists $x^{\mathbf{v_1}} \in T^\varphi$ such that $[r^\varphi]_{\pi^\varphi} = [x^{\mathbf{v_1}}]_{\pi^\varphi}$ with $\mathbf{v_1} = (\psi, \alpha)$. By construction, we also know $\mathcal{T}_1^{\mathbf{v_1}}, r_1^{\mathbf{v_1}} \models \psi$ and $\mathcal{T}_1^{\mathbf{v_1}}, r_1^{\mathbf{v_1}}, x^{\mathbf{v_1}} \models \alpha$. Then, by Fact 136, $\mathcal{T}^\varphi, r^\varphi \models \langle \varepsilon = \downarrow[\psi]\alpha\rangle$.
- In the second case, $\langle\downarrow[\psi]\alpha\rangle$ is consistent. Then, by construction plus Lemma 111, there is $x \in T^\varphi$ such that $\mathcal{T}^{\mathbf{v_2}}, r^{\mathbf{v_2}} \models \psi$, $\mathcal{T}^{\mathbf{v_2}}, r^{\mathbf{v_2}}, x \models \alpha$ and $[r^\varphi]_{\pi^\varphi} = [x]_{\pi^\varphi}$ with $\mathbf{v_2} = (\psi, \alpha)$. Then, by Fact 136, $\mathcal{T}^\varphi, r^\varphi \models \langle \varepsilon = \downarrow[\psi]\alpha\rangle$.

Verification of (C3). Suppose $\langle \varepsilon \neq \downarrow[\psi]\alpha\rangle$ is a conjunct of φ. Then there are two possibilities, $(\psi, \alpha) \in \mathbf{V}_{=,\neq}$ or $(\psi, \alpha) \in \mathbf{V}_{\neg=,\neq}$.

- In the first case, by Rule 1 plus Lemmas 111, 121 and 124, there is $x \in T^\varphi$ such that (for $\mathbf{v_1} = (\psi, \alpha)$) $\mathcal{T}_2^{\mathbf{v_1}}, r_2^{\mathbf{v_1}}, x \models \alpha$ and $x \notin [z]_{\pi^\varphi}$ for all z such that $\mathcal{T}_2^{\mathbf{v_1}}, r_2^{\mathbf{v_1}}, z \models \beta$ for some $(\psi, \beta) \in \mathbf{V}_{=,\neg\neq}$ (The argument is similar to the ones used in the proof of Fact 132 to make conclusions from Lemma 121). We also know by construction that $\mathcal{T}_2^{\mathbf{v_1}}, r_2^{\mathbf{v_1}} \models \psi$. In order to conclude from Fact 136 that $\mathcal{T}^\varphi, r^\varphi \models \langle \varepsilon \neq \downarrow[\psi]\alpha\rangle$, it only remains to observe that $[r^\varphi]_{\pi^\varphi} \neq [x]_{\pi^\varphi}$ (for a sketch of the proof see Sketch 138 in §2.6).
- In the second case, by Rule 3 plus Lemma 111, there exists $x \in T^\varphi$ such that (for $\mathbf{v_3} = (\psi, \alpha)$) $\mathcal{T}^{\mathbf{v_3}}, r^{\mathbf{v_3}}, x \models \alpha$. We also know by construction that $\mathcal{T}^{\mathbf{v_3}}, r^{\mathbf{v_3}} \models \psi$. In order to conclude from Fact 136 that $\mathcal{T}^\varphi, r^\varphi \models \langle \varepsilon \neq \downarrow[\psi]\alpha\rangle$, it only remains to observe that $[r^\varphi]_{\pi^\varphi} \neq [x]_{\pi^\varphi}$ (the proof follows the same sketch than the previous case).

Verification of (C4). Suppose $\langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$ is a conjunct of φ. By the consistency of φ plus NeqAx6, neither (ψ,α) nor (ρ,β) can be in $\mathbf{V}_{\neg=,\neg\neq}$. By the consistency of φ plus NeqAx6, it cannot be the case that one of them belongs to $\mathbf{V}_{=,\neg\neq}$ and the other one to $\mathbf{V}_{\neg=,\neq}$. Then there are five possibilities to consider (we are omitting symmetric cases):

- If $(\psi,\alpha) \in \mathbf{V}_{=,\neq}$ and $(\rho,\beta) \in \mathbf{V}_{=,\neq}$, by construction, there is $x^{\mathbf{v}_1} \in T^\varphi$ such that $\mathcal{T}_1^{\mathbf{v}_1}, r_1^{\mathbf{v}_1} \models \psi$, $\mathcal{T}_1^{\mathbf{v}_1}, r_1^{\mathbf{v}_1}, x^{\mathbf{v}_1} \models \alpha$ and $[r^\varphi]_{\pi^\varphi} = [x^{\mathbf{v}_1}]_{\pi^\varphi}$, with $\mathbf{v}_1 = (\psi,\alpha)$. Since the same happens with (ρ,β), we can conclude from Fact 136 that $\mathcal{T}^\varphi, r^\varphi \models \langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$.

- If $(\psi,\alpha) \in \mathbf{V}_{=,\neq}$ and $(\rho,\beta) \in \mathbf{V}_{=,\neg\neq}$, by construction, there is $x^{\mathbf{v}_1} \in T^\varphi$ such that $\mathcal{T}_1^{\mathbf{v}_1}, r_1^{\mathbf{v}_1} \models \psi$, $\mathcal{T}_1^{\mathbf{v}_1}, r_1^{\mathbf{v}_1}, x^{\mathbf{v}_1} \models \alpha$ and $[r^\varphi]_{\pi^\varphi} = [x^{\mathbf{v}_1}]_{\pi^\varphi}$, with $\mathbf{v}_1 = (\psi,\alpha)$. By Lemma 111 plus Rule 2, there is $x \in T^{\mathbf{v}_2}$ (with $\mathbf{v}_2 = (\rho,\beta)$) such that $\mathcal{T}^{\mathbf{v}_2}, r^{\mathbf{v}_2} \models \rho$ and $\mathcal{T}^{\mathbf{v}_2}, r^{\mathbf{v}_2}, x \models \beta$. Then, by construction, $[r^\varphi]_{\pi^\varphi} = [x]_{\pi^\varphi}$ and so $[x^{\mathbf{v}_1}]_{\pi^\varphi} = [x]_{\pi^\varphi}$. We conclude from Fact 136 that $\mathcal{T}^\varphi, r^\varphi \models \langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$.

- If $(\psi,\alpha) \in \mathbf{V}_{=,\neq}$ and $(\rho,\beta) \in \mathbf{V}_{\neg=,\neq}$, by the consistency of φ plus NeqAx6, $(\psi,\alpha,\rho,\beta) = \mathbf{u} \in \mathbf{U}$. Then, by construction, there are $x^{\mathbf{u}} \in T_1^{\mathbf{u}}$, $y^{\mathbf{u}} \in T_2^{\mathbf{u}}$ such that $\mathcal{T}_1^{\mathbf{u}}, r_1^{\mathbf{u}} \models \psi$, $\mathcal{T}_2^{\mathbf{u}}, r_2^{\mathbf{u}} \models \rho$, $\mathcal{T}_1^{\mathbf{u}}, r_1^{\mathbf{u}}, x^{\mathbf{u}} \models \alpha$, $\mathcal{T}_2^{\mathbf{u}}, r_2^{\mathbf{u}}, y^{\mathbf{u}} \models \beta$ and $[x^{\mathbf{u}}]_{\pi^\varphi} = [y^{\mathbf{u}}]_{\pi^\varphi}$ (If $\mathbf{u} \in \mathbf{U_2}$ the assertion is straightforward and if $\mathbf{u} \in \mathbf{U_1}$ these nodes exist because of the gluing related to the set Z). Then, we conclude from Fact 136 that $\mathcal{T}^\varphi, r^\varphi \models \langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$.

- If $(\psi,\alpha) \in \mathbf{V}_{=,\neg\neq}$ and $(\rho,\beta) \in \mathbf{V}_{=,\neg\neq}$, by Rule 2 plus Lemma 111, there are $x \in T^{\mathbf{v}_2}$ (with $\mathbf{v}_2 = (\psi,\alpha)$) and $y \in T^{\mathbf{v}'_2}$ (with $\mathbf{v}'_2 = (\rho,\beta)$) such that $\mathcal{T}^{\mathbf{v_2}}, r^{\mathbf{v_2}} \models \psi$, $\mathcal{T}^{\mathbf{v}_2}, r^{\mathbf{v}_2}, x \models \alpha$, $\mathcal{T}^{\mathbf{v}'_2}, r^{\mathbf{v}'_2} \models \rho$ and $\mathcal{T}^{\mathbf{v}'_2}, r^{\mathbf{v}'_2}, y \models \beta$. By construction, $[x]_{\pi^\varphi} = [r^\varphi]_{\pi^\varphi} = [y]_{\pi^\varphi}$ and so, we conclude from Fact 136 that $\mathcal{T}^\varphi, r^\varphi \models \langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$.

- If $(\psi,\alpha) \in \mathbf{V}_{\neg=,\neq}$ and $(\rho,\beta) \in \mathbf{V}_{\neg=,\neq}$, then $(\psi,\alpha,\rho,\beta) = \mathbf{u} \in \mathbf{U}$ or $(\psi,\alpha,\rho,\beta) = \mathbf{z} \in \mathbf{Z}$. In the first case, the proof is exactly the same given for the case that $(\psi,\alpha) \in \mathbf{V}_{=,\neq}$ and $(\rho,\beta) \in \mathbf{V}_{\neg=,\neq}$. In the other case, by Rule 3 plus Lemma 111, there are $x \in T^{\mathbf{v}_3}$ (with $\mathbf{v}_3 = (\psi,\alpha)$), $y \in T^{\mathbf{v}'_3}$ (with $\mathbf{v}'_3 = (\rho,\beta)$) such that $\mathcal{T}^{\mathbf{v}_3}, r^{\mathbf{v}_3} \models \psi$, $\mathcal{T}^{\mathbf{v}_3}, r^{\mathbf{v}_3}, x \models \alpha$, $\mathcal{T}^{\mathbf{v}'_3}, r^{\mathbf{v}'_3} \models \rho$ and $\mathcal{T}^{\mathbf{v}'_3}, r^{\mathbf{v}'_3}, y \models \beta$. Observe that $[x]_{\pi^\varphi} = [y]_{\pi^\varphi}$ because of the way in which we have defined the partition π^φ. Then we conclude from Fact 136 that $\mathcal{T}^\varphi, r^\varphi \models \langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$.

Verification of (C5). Suppose $\langle\downarrow[\psi]\alpha \neq \downarrow[\rho]\beta\rangle$ is a conjunct of φ. By the consistency of φ plus NeqAx6, neither (ψ,α) nor (ρ,β) can be in $\mathbf{V}_{\neg=,\neg\neq}$. By the consistency of φ plus NeqAx7, it cannot be the case that they both belong to $\mathbf{V}_{=,\neg\neq}$. Then there are five possibilities to consider:

- If $(\psi,\alpha) \in \mathbf{V}_{=,\neq}$ and $(\rho,\beta) \in \mathbf{V}_{=,\neq}$, by items (C2) and (C3), there exist $x, y \in T^\varphi$, such that $\mathcal{T}^\varphi, r^\varphi, x \models \downarrow[\psi]\alpha$, $\mathcal{T}^\varphi, r^\varphi, y \models \downarrow[\rho]\beta$, $[r^\varphi]_{\pi^\varphi} = [x]_{\pi^\varphi}$ and $[r^\varphi]_{\pi^\varphi} \neq [y]_{\pi^\varphi}$. Then we conclude that $\mathcal{T}^\varphi, r^\varphi \models \langle\downarrow[\psi]\alpha \neq \downarrow[\rho]\beta\rangle$.

- If $(\psi, \alpha) \in \mathbf{V}_{=,\neq}$ and $(\rho, \beta) \in \mathbf{V}_{=,\neg\neq}$, or $(\psi, \alpha) \in \mathbf{V}_{=,\neq}$ and $(\rho, \beta) \in \mathbf{V}_{\neg=,\neq}$ or $(\psi, \alpha) \in \mathbf{V}_{=,\neg\neq}$ and $(\rho, \beta) \in \mathbf{V}_{\neg=,\neq}$, the proof is analogous to the previous one.

- If $(\psi, \alpha) = \mathbf{v}_3 \in \mathbf{V}_{\neg=,\neq}$ and $(\rho, \beta) = \mathbf{v}'_\mathbf{3} \in \mathbf{V}_{\neg=,\neq}$.

  – In case $(\psi, \alpha) \neq (\rho, \beta)$: If $\langle \alpha \neq \alpha \rangle$ is a conjunct of ψ (if $\langle \beta \neq \beta \rangle$ is a conjunct of ρ, the proof is analogous), by Lemma 111, Rule 3 and Fact 134 there exist $x, y \in T^{\mathbf{v}_3}$, $z \in T^{\mathbf{v}'_3}$ such that $\mathcal{T}^{\mathbf{v}_3}, r^{\mathbf{v}_3} \models \psi$, $\mathcal{T}^{\mathbf{v}_3}, r^{\mathbf{v}_3}, x \models \alpha$, $\mathcal{T}^{\mathbf{v}_3}, r^{\mathbf{v}_3}, y \models \alpha$, $\mathcal{T}^{\mathbf{v}'_3}, r^{\mathbf{v}'_3} \models \rho$, $\mathcal{T}^{\mathbf{v}'_3}, r^{\mathbf{v}'_3}, z \models \beta$ and $[x]_{\pi^\varphi} \neq [y]_{\pi^\varphi}$. Then we conclude from Fact 136 that $\mathcal{T}^\varphi, r^\varphi \models \langle \downarrow[\psi]\alpha \neq \downarrow[\rho]\beta \rangle$ (either x or y is not in $[z]_{\pi^\varphi}$).

    Suppose then that $\neg\langle \alpha \neq \alpha \rangle$ is a conjunct of ψ and $\neg\langle \beta \neq \beta \rangle$ is a conjunct of ρ. Then, as before, there exist $x \in T^{\mathbf{v}_3}$, $z \in T^{\mathbf{v}'_3}$ such that $\mathcal{T}^{\mathbf{v}_3}, r^{\mathbf{v}_3} \models \psi$, $\mathcal{T}^{\mathbf{v}_3}, r^{\mathbf{v}_3}, x \models \alpha$, $\mathcal{T}^{\mathbf{v}'_3}, r^{\mathbf{v}'_3} \models \rho$, $\mathcal{T}^{\mathbf{v}'_3}, r^{\mathbf{v}'_3}, z \models \beta$. To conclude the proof, it only remains to observe that, in this case, $[x]_{\pi^\varphi} \neq [z]_{\pi^\varphi}$ (for a sketch of the proof see Sketch 139 in §2.6). Then we conclude from Fact 136 that $\mathcal{T}^\varphi, r^\varphi \models \langle \downarrow[\psi]\alpha \neq \downarrow[\rho]\beta \rangle$.

  – In case $(\psi, \alpha) = (\rho, \beta)$, by consistency of φ, we have that $(\psi, \alpha, \psi, \alpha) = \mathbf{u} \in \mathbf{U}$. If $\langle \alpha \neq \alpha \rangle$ is a conjunct of ψ, by Lemma 111, Rule 3 and Fact 134 there exist $x, y \in T^{\mathbf{v}_3}$ such that $\mathcal{T}^{\mathbf{v}_3}, r^{\mathbf{v}_3} \models \psi$, $\mathcal{T}^{\mathbf{v}_3}, r^{\mathbf{v}_3}, x \models \alpha$, $\mathcal{T}^{\mathbf{v}_3}, r^{\mathbf{v}_3}, y \models \alpha$ and $[x]_{\pi^\varphi} \neq [y]_{\pi^\varphi}$. Then we conclude from Fact 136 that $\mathcal{T}^\varphi, r^\varphi \models \langle \downarrow[\psi]\alpha \neq \downarrow[\psi]\alpha \rangle$.

    Suppose then that $\neg\langle \alpha \neq \alpha \rangle$ is a conjunct of ψ. Then, as before, there exist $x \in T^{\mathbf{v}_3}$, $z \in T_1^{\mathbf{u}}$ such that $\mathcal{T}^{\mathbf{v}_3}, r^{\mathbf{v}_3} \models \psi$, $\mathcal{T}^{\mathbf{v}_3}, r^{\mathbf{v}_3}, x \models \alpha$, $\mathcal{T}_1^{\mathbf{u}}, r_1^{\mathbf{u}} \models \psi$, $\mathcal{T}_1^{\mathbf{u}}, r_1^{\mathbf{u}}, z \models \alpha$. To conclude the proof, it only remains to observe that, in this case, $[x]_{\pi^\varphi} \neq [z]_{\pi^\varphi}$ (for a sketch of the proof see Sketch 140 in §2.6). Then we conclude from Fact 136 that $\mathcal{T}^\varphi, r^\varphi \models \langle \downarrow[\psi]\alpha \neq \downarrow[\rho]\beta \rangle$.

Verification of (C6). Suppose $\neg\langle \varepsilon = \downarrow[\psi]\alpha \rangle$ is a conjunct of φ. Aiming for a contradiction, suppose that $\mathcal{T}^\varphi, r^\varphi \models \langle \varepsilon = \downarrow[\psi]\alpha \rangle$. Then there is a successor z of r^φ in which ψ holds, and, by construction plus Lemma 112, z is the root of some copy of the tree $\mathcal{T}^\psi$, i.e. $z = r^\psi$ (it might be $\widetilde{\mathcal{T}^\psi}$ and $\widetilde{r^\psi}$ but, in that case, the argument is the same). Moreover, there is $x \in T^\psi$ such that $\mathcal{T}^\psi, r^\psi, x \models \alpha$, with $[x]_{\pi^\varphi} = [r^\varphi]_{\pi^\varphi}$. In addition to this, $(\psi, \alpha) \in \mathbf{V}_{\neg=,\neq}$ or $(\psi, \alpha) \in \mathbf{V}_{\neg=,\neg\neq}$. If the latter occurs, by construction of $\mathcal{T}^\varphi$ plus Lemma 119 and Lemma 112, we have that $\neg\langle \alpha = \alpha \rangle$ is a conjunct of ψ which is a contradiction. In the former, observe that $[x]_{\pi^\varphi} \neq [r^\varphi]_{\pi^\varphi}$ (for a sketch of the proof see Sketch 141 in §2.6) which is a contradiction.

Verification of (C7). Suppose $\neg\langle \varepsilon \neq \downarrow[\psi]\alpha \rangle$ is a conjunct of φ. Aiming for a contradiction, suppose that $\mathcal{T}^\varphi, r^\varphi \models \langle \varepsilon \neq \downarrow[\psi]\alpha \rangle$. Then there is a successor z of r^φ in which ψ holds, and by construction and Lemma 112, z is the root of some copy of the tree $\mathcal{T}^\psi$, i.e. $z = r^\psi$ (it might be $\widetilde{\mathcal{T}^\psi}$ and $\widetilde{r^\psi}$ but, in that case, the argument is the same). Moreover, there is $x \in T^\psi$ such that $\mathcal{T}^\psi, r^\psi, x \models \alpha$, with $[x]_{\pi^\varphi} \neq [r^\varphi]_{\pi^\varphi}$. Then, by construction, $(\psi, \alpha) \notin \mathbf{V}_{=,\neg\neq}$. Since $\neg\langle \varepsilon \neq \downarrow[\psi]\alpha \rangle$ is a conjunct of φ, the only remaining possibility

is that $(\psi, \alpha) \in \mathbf{V}_{\neg=,\neg\neq}$ but this is a contradiction by construction plus Lemma 119 and Lemma 112.

Verification of (C8). Suppose $\neg\langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$ is a conjunct of φ. By the consistency of φ plus NeqAx6, it cannot be the case that both $(\psi, \alpha), (\rho, \beta)$ are in $\mathbf{V}_{=,\neq} \cup \mathbf{V}_{=,\neg\neq}$. In case $(\psi, \alpha) \in \mathbf{V}_{\neg=,\neg\neq}$ (if $(\rho, \beta) \in \mathbf{V}_{\neg=,\neg\neq}$, the proof is analogous), suppose that $\mathcal{T}^\varphi, r^\varphi \models \langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$. In particular, there is a successor of r^φ, z and a descendant w such that $\mathcal{T}^\varphi, z, w \models [\psi]\alpha$. But this is a contradiction by construction plus Lemma 119 and Lemma 112. Then $\mathcal{T}^\varphi, r^\varphi \models \neg\langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$. We then have three remaining cases to analyze:

- If $(\psi, \alpha) \in \mathbf{V}_{=,\neg\neq}$ and $(\rho, \beta) \in \mathbf{V}_{\neg=,\neq}$, then, by items (C6) and (C7), we have the result.
- If $(\psi, \alpha) \in \mathbf{V}_{=,\neq}$ and $(\rho, \beta) \in \mathbf{V}_{\neg=,\neq}$ or $(\psi, \alpha), (\rho, \beta) \in \mathbf{V}_{\neg=,\neq}$. In order to conclude that $\mathcal{T}^\varphi, r^\varphi \models \neg\langle\downarrow[\psi]\alpha = \downarrow[\rho]\beta\rangle$, one only have to observe that, if $x, y \in T^\varphi$ are such that $\mathcal{T}^\varphi, r^\varphi, x \models \downarrow[\psi]\alpha$ and $\mathcal{T}^\varphi, r^\varphi, y \models \downarrow[\rho]\beta$, then $[x]_{\pi^\varphi} \neq [y]_{\pi^\varphi}$ (for a sketch of the proof see Sketch 142 in §2.6).

Verification of (C9). Suppose $\neg\langle\downarrow[\psi]\alpha \neq \downarrow[\rho]\beta\rangle$ is a conjunct of φ. By the consistency of φ plus NeqAx6, it cannot be the case that one of $(\psi, \alpha), (\rho, \beta)$ is from $\mathbf{V}_{=,\neq}$ and the other from $\mathbf{V}_{=,\neg\neq}$, neither can one be from $\mathbf{V}_{=,\neq}$ and the other from $\mathbf{V}_{\neg=,\neq}$, or one from $\mathbf{V}_{=,\neg\neq}$ and the other from $\mathbf{V}_{\neg=,\neq}$, or both from $\mathbf{V}_{=,\neq}$. In case $(\psi, \alpha) \in \mathbf{V}_{\neg=,\neg\neq}$ (if $(\rho, \beta) \in \mathbf{V}_{\neg=,\neg\neq}$, the proof is analogous), suppose that $\mathcal{T}^\varphi, r^\varphi \models \langle\downarrow[\psi]\alpha \neq \downarrow[\rho]\beta\rangle$. In particular, there is a successor z of r^φ and a descendant w such that $\mathcal{T}^\varphi, z, w \models [\psi]\alpha$. But this is a contradiction by construction plus Lemma 119 and Lemma 112. We then have two remaining cases to analyze:

- If $(\psi, \alpha), (\rho, \beta) \in \mathbf{V}_{=,\neg\neq}$, by item (C7), we have the result.
- If $(\psi, \alpha), (\rho, \beta) \in \mathbf{V}_{\neg=,\neq}$, by the consistency of φ plus NeqAx6, $(\psi, \alpha, \rho, \beta) \in \mathbf{Z}$ and the result follows immediately from the construction of the model.

2.5 Bounded tree model property

The addition of an equivalence relation on top of a tree-like Kripke model, and the ability of the language to compare if two nodes at the end of path expressions are in the *same* or in *different* equivalence classes has proved to change remarkably the canonical model construction of the basic modal logic. When the language has only comparisons by 'equality', the situation is somewhat simpler, based on the fact that 'equality' is a transitive relation. Also notice that while

"all pairs of paths with certain properties end in *different* equivalence classes" (19)

is expressible when tests by equality are present,

"all pairs of paths with certain properties end in the *same* equivalence classes" (20)

is only expressible when tests by inequality are also present. Both properties are universal. However, in the construction of the canonical model, (19) is compatible with adding many disjoint copies of subtrees with disjoint partitions, while (20) is not. The axiomatization for the fragment containing both the operators of 'equality' and 'inequality' proved to be much more involved than the one containing only 'equality', as witnessed by the large amount of axioms reflecting the intricate relationships between both binary operators.

In this chapter we have considered $\mathrm{XPath}_{=}(\downarrow)$ over arbitrary data trees. Although in the database community it may make no sense to consider infinite data trees (an XML document is always finite), we allow for that possibility. Furthermore, $\mathrm{XPath}_{=}(\downarrow)$ is also suitable for reasoning about (finite or infinite) data *graphs*, as it is done in [77, 1]. In either of the alternatives (finite vs. infinite data trees vs. data graphs) it can be shown that $\mathrm{XPath}_{=}(\downarrow)$ is also axiomatizable by the system given in this chapter —notice there are no specific axioms of an underlying tree topology. Since our construction of canonical models gives us a recursively bounded finite data tree, we conclude:

Corollary 137 (Bounded tree model property)**.** *There is a primitive recursive function f such that any satisfiable node or path expression φ of $XPath_{=}(\downarrow)$ of size n over the class of finite/arbitrary data trees/data graphs is satisfiable in a data tree of size at most $f(n)$.*

This already shows that the satisfiability problem of $\mathrm{XPath}_{=}(\downarrow)$ is decidable over any of the classes of models stated above. Of course, this result —at least for $\mathrm{XPath}_{=}(\downarrow)$ over finite data trees— is not new, as mentioned in the introduction [42]. However, the canonical model construction may gives us new insight to obtain sequent calculus axiomatizations, as done in [10], which might be useful for obtaining alternative proofs of complexity for the satisfiability problem of fragments or extensions of $\mathrm{XPath}_{=}(\downarrow)$.

2.6 Technical material

Lema 114. *Let $* \in \{=, \neq\}$, $\gamma \in P_n$, $\psi_i \in N_{n-i}$ for $i = 1, \ldots, i_0$, $\alpha, \beta \in P_{n-i_0}$ such that*

$$\langle \gamma * \downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\alpha\rangle \wedge \neg\langle \gamma * \downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\beta\rangle$$

is consistent and $\neg\langle \alpha \neq \alpha\rangle$ is a conjunct of ψ_{i_0}. Then $\neg\langle \alpha = \beta\rangle$ is a conjunct of ψ_{i_0}.

Proof. Let us start with the case of $*$ being $\neq$. Aiming for a contradiction, suppose that $\langle \gamma \neq \downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\alpha\rangle \wedge \neg\langle \gamma \neq \downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\beta\rangle$ is consistent and that both $\neg\langle \alpha \neq \alpha\rangle$ and $\langle \alpha = \beta\rangle$ are conjuncts of ψ_{i_0}.

First, let us prove some facts that will be useful in the rest of the proof:

1. The following derivation:

$$\begin{aligned}
\langle \gamma \neq \downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\alpha\rangle &\leq \langle \downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\alpha\rangle && \text{(NeqAx4)}\\
&\leq \langle \downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\rangle && \text{(\textbf{Der12} Fact 90)}\\
&\leq \langle \downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\alpha = \downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\beta\rangle\\
& && \text{(EqAx5 \& \textbf{Der21} (Fact 90))}\\
&\leq \langle \downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\beta\rangle && \text{(EqAx4)}
\end{aligned}$$

   In particular, by **Der13** (Fact 90), we have that $\langle \downarrow[\psi_{i_0}]\beta\rangle$ is consistent and so $\langle \beta = \beta\rangle$ is a conjunct of ψ_{i_0} (by Lemma 111).

2. From the second line of Item 1, we have that $\langle \downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\rangle$ is consistent, and then, by **Der13** (Fact 90), ψ_{i_0} is consistent.

3. Aiming for a contradiction, let us suppose that $\langle \beta \neq \beta\rangle$ is a conjunct of ψ_{i_0}. Then

$$\begin{aligned}
&\langle \gamma \neq \downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\alpha\rangle \wedge \neg\langle \gamma \neq \downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\beta\rangle\\
&\quad \leq \langle \gamma\rangle \wedge \langle \downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\rangle \wedge \neg\langle \gamma \neq \downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\beta\rangle && \text{(NeqAx4 \& Item 1)}\\
&\quad \leq \langle \gamma\rangle \wedge \langle \downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\beta \neq \downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\beta\rangle \wedge \neg\langle \gamma \neq \downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\beta\rangle\\
& && \text{(NeqAx5 \& \textbf{Der21} (Fact 90))}\\
&\quad \equiv \langle \gamma \neq \downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\beta\rangle \wedge \neg\langle \gamma \neq \downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\beta\rangle && \text{(NeqAx7)}\\
&\quad \equiv \text{FALSE} && \text{(Boolean)}
\end{aligned}$$

   which is a contradiction. Then $\neg\langle \beta \neq \beta\rangle$ is a conjunct of ψ_{i_0}.

4. Because ψ_{i_0} is consistent (Item 2), by the previous Item plus NeqAx7, $\neg\langle \alpha \neq \beta\rangle$ is a conjunct of ψ_{i_0}.

Then we have

$$\begin{aligned}
&\langle \gamma{\neq}\downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\alpha\rangle \wedge \neg\langle \gamma{\neq}\downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\beta\rangle\\
&\leq \langle \gamma{\neq}\downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\beta\rangle \wedge \neg\langle \gamma{\neq}\downarrow[\psi_1] \ldots \downarrow[\psi_{i_0}]\beta\rangle\\
& && \text{(Items 1 and 4 \& NeqAx9 \& \textbf{Der21} (Fact 90))}\\
&\equiv \text{FALSE} && \text{(Boolean)}
\end{aligned}$$

which is contradiction, from the assumption that $\langle \alpha = \beta\rangle$ was a conjunct of ψ_{i_0}. Therefore, $\neg\langle \alpha = \beta\rangle$ is a conjunct of ψ_{i_0}.

For the case of $*$ being $=$, use NeqAx10 plus **Der21** of Fact 90. □

Lema 115. *Let $\psi \in N_n$, $\alpha, \beta \in P_n$ such that $\langle \downarrow[\psi]\alpha \neq \downarrow[\psi]\alpha\rangle \wedge \neg\langle \downarrow[\psi]\gamma \neq \downarrow[\psi]\gamma\rangle$ is consistent and $\neg\langle \alpha \neq \alpha\rangle$ is a conjunct of ψ. Then $\neg\langle \alpha = \gamma\rangle$ is a conjunct of ψ.*

Proof. Aiming for a contradiction, suppose that $\langle \downarrow[\psi]\alpha \neq \downarrow[\psi]\alpha\rangle \wedge \neg\langle \downarrow[\psi]\gamma \neq \downarrow[\psi]\gamma\rangle$ is consistent and both $\neg\langle \alpha \neq \alpha\rangle$ and $\langle \alpha = \gamma\rangle$ are conjuncts of ψ.

Let us prove some facts that will be useful in the rest of the proof:

1. The following derivation:

$$\begin{aligned}
\langle\downarrow[\psi]\alpha \neq \downarrow[\psi]\alpha\rangle &\leq \langle\downarrow[\psi]\alpha\rangle && \text{(NeqAx4)}\\
&\leq \langle\downarrow[\psi]\rangle && \text{(\textbf{Der12} (Fact 90))}\\
&\leq \langle\downarrow[\psi]\alpha = \downarrow[\psi]\gamma\rangle && \text{(EqAx5 \& \textbf{Der21}(Fact 90))}\\
&\leq \langle\downarrow[\psi]\gamma\rangle && \text{(EqAx4)}
\end{aligned}$$

   In particular, we have that $\langle\gamma = \gamma\rangle$ is a conjunct of ψ (by Lemma 111).

2. From the second line of Item 1, we have that $\langle\downarrow[\psi]\rangle$ is consistent, and by **Der13** (Fact 90), ψ is consistent.

3. Aiming for a contradiction, let us suppose that $\langle\gamma \neq \gamma\rangle$ is a conjunct of ψ. Then

$$\begin{aligned}
&\langle\downarrow[\psi]\alpha \neq \downarrow[\psi]\alpha\rangle \wedge \neg\langle\downarrow[\psi]\gamma \neq \downarrow[\psi]\gamma\rangle\\
&\quad\leq \langle\downarrow[\psi]\rangle \wedge \neg\langle\downarrow[\psi]\gamma \neq \downarrow[\psi]\gamma\rangle && \text{(Item 1)}\\
&\quad\leq \langle\downarrow[\psi]\gamma \neq \downarrow[\psi]\gamma\rangle \wedge \neg\langle\downarrow[\psi]\gamma \neq \downarrow[\psi]\gamma\rangle && \text{(NeqAx5 \& \textbf{Der21} (Fact 90))}\\
&\quad\equiv \text{FALSE} && \text{(Boolean)}
\end{aligned}$$

   which is a contradiction. Then $\neg\langle\gamma \neq \gamma\rangle$ is a conjunct of ψ.

4. Because ψ is consistent (Item 2), by the previous item plus NeqAx7, $\neg\langle\alpha \neq \gamma\rangle$ is a conjunct of ψ.

Then we have

$$\begin{aligned}
&\langle\downarrow[\psi]\alpha \neq \downarrow[\psi]\alpha\rangle \wedge \neg\langle\downarrow[\psi]\gamma \neq \downarrow[\psi]\gamma\rangle\\
&\quad\leq \langle\downarrow[\psi]\gamma\rangle \wedge \langle\downarrow[\psi]\alpha \neq \downarrow[\psi]\alpha\rangle \wedge \neg\langle\downarrow[\psi]\gamma \neq \downarrow[\psi]\gamma\rangle && \text{(Item 1)}\\
&\quad\leq \langle\downarrow[\psi]\alpha \neq \downarrow[\psi]\gamma\rangle \wedge \neg\langle\downarrow[\psi]\gamma \neq \downarrow[\psi]\gamma\rangle && \text{(NeqAx7)}\\
&\quad\leq \langle\downarrow[\psi]\gamma \neq \downarrow[\psi]\gamma\rangle \wedge \neg\langle\downarrow[\psi]\gamma \neq \downarrow[\psi]\gamma\rangle && \text{(Items 1 and 4, \& NeqAx9 \& \textbf{Der21} (Fact 90))}\\
&\quad\equiv \text{FALSE} && \text{(Boolean)}
\end{aligned}$$

which is contradiction, from the assumption that $\langle\alpha = \gamma\rangle$ was a conjunct of ψ. Therefore, $\neg\langle\alpha = \gamma\rangle$ is a conjunct of ψ. □

Lema 119. *Let $\psi \in N_n$, $\alpha \in P_n$, $\gamma \in P_{n+1}$. If $\neg\langle\gamma = \downarrow[\psi]\alpha\rangle \wedge \neg\langle\gamma \neq \downarrow[\psi]\alpha\rangle \wedge \langle\gamma\rangle \wedge \langle\downarrow[\psi]\rangle$ is consistent, then $\neg\langle\alpha = \alpha\rangle$ is a conjunct of ψ.*

Proof. Aiming for a contradiction, suppose that $\langle\alpha = \alpha\rangle$ is a conjunct of ψ. Then

$$\begin{aligned}
&\neg\langle\gamma = \downarrow[\psi]\alpha\rangle \wedge \neg\langle\gamma \neq \downarrow[\psi]\alpha\rangle \wedge \langle\gamma\rangle \wedge \langle\downarrow[\psi]\rangle\\
&\quad\leq \neg\langle\gamma = \downarrow[\psi]\alpha\rangle \wedge \neg\langle\gamma \neq \downarrow[\psi]\alpha\rangle \wedge \langle\gamma\rangle \wedge \langle\downarrow[\psi]\alpha = \downarrow[\psi]\alpha\rangle && \text{(EqAx5 \& \textbf{Der21} (Fact 90))}\\
&\quad\equiv \neg\langle\gamma\rangle \wedge \langle\gamma\rangle && \text{(NeqAx6)}\\
&\quad\equiv \text{FALSE} && \text{(Boolean)}
\end{aligned}$$

and this concludes the proof. □

Lema 121. *Let* $* \in \{=, \neq\}$, $\psi \in N_n$, $\alpha, \beta \in P_n$, $\gamma \in P_{n+1}$. *If* $\langle \gamma = \downarrow[\psi]\alpha\rangle \wedge \neg\langle \gamma \neq \downarrow[\psi]\alpha\rangle \wedge \neg\langle \gamma * \downarrow[\psi]\beta\rangle$ *is consistent, then* $\neg\langle \alpha * \beta\rangle$ *is a conjunct of* ψ.

Proof. Let us first prove the case for $*$ being $\neq$. Suppose that $\langle \gamma = \downarrow[\psi]\alpha\rangle \wedge \neg\langle \gamma \neq \downarrow[\psi]\alpha\rangle \wedge \neg\langle \gamma \neq \downarrow[\psi]\beta\rangle$ is consistent. Aiming for a contradiction, suppose that $\langle \alpha \neq \beta\rangle$ is a conjunct of ψ. Then

$$
\begin{aligned}
&\langle \gamma = \downarrow[\psi]\alpha\rangle \wedge \neg\langle \gamma \neq \downarrow[\psi]\alpha\rangle \wedge \neg\langle \gamma \neq \downarrow[\psi]\beta\rangle \\
&\quad \equiv \langle \gamma = \downarrow[\psi \wedge \langle \alpha \neq \beta\rangle]\alpha\rangle \wedge \neg\langle \gamma \neq \downarrow[\psi]\alpha\rangle \wedge \neg\langle \gamma \neq \downarrow[\psi]\beta\rangle && \text{(Hypothesis)} \\
&\quad \leq \langle \gamma\rangle \wedge \langle \downarrow[\psi \wedge \langle \alpha \neq \beta\rangle]\alpha\rangle \wedge \neg\langle \gamma \neq \downarrow[\psi]\alpha\rangle \wedge \neg\langle \gamma \neq \downarrow[\psi]\beta\rangle && \text{(EqAx1 \& EqAx4)} \\
&\quad \leq \langle \gamma\rangle \wedge \langle \downarrow[\psi \wedge \langle \alpha \neq \beta\rangle]\rangle \wedge \neg\langle \gamma \neq \downarrow[\psi]\alpha\rangle \wedge \neg\langle \gamma \neq \downarrow[\psi]\beta\rangle && \text{(\textbf{Der12} (Fact 90))} \\
&\quad \leq \langle \gamma\rangle \wedge \langle \downarrow[\psi]\alpha \neq \downarrow[\psi]\beta\rangle \wedge \neg\langle \gamma \neq \downarrow[\psi]\alpha\rangle \wedge \neg\langle \gamma \neq \downarrow[\psi]\beta\rangle && \text{(NeqAx5 \& \textbf{Der21} (Fact 90))} \\
&\quad \leq \langle \gamma \neq \downarrow[\psi]\beta\rangle \wedge \neg\langle \gamma \neq \downarrow[\psi]\beta\rangle && \text{(NeqAx7)} \\
&\quad \equiv \text{FALSE} && \text{(Boolean)}
\end{aligned}
$$

which is a contradiction. Then $\langle \alpha \neq \beta\rangle$ is a conjunct of ψ. For the case in which $*$ is $=$, the proof is similar but instead of NeqAx5 we use EqAx5 and instead of NeqAx7 we use NeqAx6.

□

Lema 124. *Let* $* \in \{=, \neq\}$, $\psi \in N_n$, $\alpha, \beta \in P_n$, $\gamma \in P_{n+1}$. *If* $\langle \gamma = \downarrow[\psi]\alpha\rangle \wedge \neg\langle \gamma \neq \downarrow[\psi]\alpha\rangle \wedge \langle \gamma * \downarrow[\psi]\beta\rangle$ *is consistent, then* $\langle \alpha * \beta\rangle$ *is a conjunct of* ψ.

Proof. Let us first prove the case for $*$ being $=$. Suppose that $\langle \gamma = \downarrow[\psi]\alpha\rangle \wedge \neg\langle \gamma \neq \downarrow[\psi]\alpha\rangle \wedge \langle \gamma = \downarrow[\psi]\beta\rangle$ is consistent. Aiming for a contradiction, suppose that $\neg\langle \alpha = \beta\rangle$ is a conjunct of ψ. Also, since $\langle \downarrow[\psi]\alpha\rangle$ is consistent (by EqAx4), then by Lemma 111 $\langle \alpha = \alpha\rangle$ is a conjunct of ψ. Then

$$
\begin{aligned}
&\langle \gamma = \downarrow[\psi]\alpha\rangle \wedge \neg\langle \gamma \neq \downarrow[\psi]\alpha\rangle \wedge \langle \gamma = \downarrow[\psi]\beta\rangle \\
&\quad \leq \neg\langle \gamma \neq \downarrow[\psi]\alpha\rangle \wedge \langle \gamma = \downarrow[\psi \wedge \neg\langle \alpha = \beta\rangle \wedge \langle \alpha\rangle]\beta\rangle && \text{(EqAx6)} \\
&\quad \leq \neg\langle \gamma \neq \downarrow[\psi]\alpha\rangle \wedge \langle \gamma \neq \downarrow[\psi]\alpha\rangle && \text{(NeqAx8 \& \textbf{Der21} (Fact 90))} \\
&\quad \equiv \text{FALSE} && \text{(Boolean)}
\end{aligned}
$$

which is a contradiction. Then $\langle \alpha = \beta\rangle$ is a conjunct of ψ. For the case in which $*$ is $\neq$, the proof is similar but using NeqAx9 instead of NeqAx8. □

Sketch 138. *Thinking in terms of sequences as in the proofs of Facts 132 and 134, one only has to observe that:*

- $[x]_{\pi_2^{\mathbf{v_1}}} \neq [z]_{\pi_2^{\mathbf{v_1}}}$ *for all* $z \in T_2^{\mathbf{v_1}}$ *in a class that was glued to the class of the* root *via a* $\text{root}_{=,\neg\neq}$*-kind gluing.*
- $\text{root}_{=,\neq}$*-kind gluings are made in different subtrees.*

- *By the same arguments given in the proofs of Facts 132 and 134, we can't have a sequence containing any of the following:*

  – $\text{root}_{=,\neq}$-Z, – Z-$\text{root}_{=,\neq}$, – $\text{root}_{=,\neg\neq}$-Z,
  – Z-$\text{root}_{=,\neg\neq}$, – $\text{root}_{=,\neq}$-U_2, – U_2-$\text{root}_{=,\neq}$,
  – $\text{root}_{=,\neg\neq}$-U_2, – U_2-$\text{root}_{=,\neg\neq}$.

Sketch 139. *Thinking in terms of sequences as in the proofs of Facts 132 and 134, one only has to observe that:*

- $[x]_{\pi^{\mathbf{v_3}}} \neq [y]_{\pi^{\mathbf{v_3}}}$ *for all* $y \in T^{\mathbf{v_3}}$ *in a class that was glued to the class of the* root *via a* $\text{root}_{=,\neg\neq}$*-kind gluing (Use Lemma 121).*

- $[z]_{\pi^{\mathbf{v'_3}}} \neq [y]_{\pi^{\mathbf{v'_3}}}$ *for all* $y \in T^{\mathbf{v'_3}}$ *in a class that was glued to the class of the* root *via a* $\text{root}_{=,\neg\neq}$*-kind gluing (Use Lemma 121).*

- $\text{root}_{=,\neq}$*-kind gluings are made in different subtrees.*

- $[x]_{\pi^{\mathbf{v_3}}}$ *and* $[z]_{\pi^{\mathbf{v'_3}}}$ *can not be glued together by a sequence of all* Z*-kind gluings because of the consistency of* φ *plus Lemmas 130 and 127.*

- $[x]_{\pi^{\mathbf{v_3}}}$ *and* $[z]_{\pi^{\mathbf{v'_3}}}$ *can not be glued together by a sequence that begins or ends with a* U_2*-kind gluing because we use new subtrees for that kind of gluings.*

- *By the same arguments given in the proofs of Facts 132 and 134, we can't have a sequence containing any of the following:*

  – $\text{root}_{=,\neq}$-Z, – Z-$\text{root}_{=,\neq}$, – $\text{root}_{=,\neg\neq}$-Z,
  – Z-$\text{root}_{=,\neg\neq}$, – $\text{root}_{=,\neq}$-U_2, – U_2-$\text{root}_{=,\neq}$,
  – $\text{root}_{=,\neg\neq}$-U_2, – U_2-$\text{root}_{=,\neg\neq}$.

- *By the same arguments given in the proof of Fact 134, we can reduce sequences with two consecutive* Z*-kind gluings to sequences that not have two consecutive* Z*-kind gluings.*

- *By the same arguments given in the proof of Fact 134, we can't have sequences containing* U_2- U_2.

- *One can prove that* $[x]_{\pi^{\mathbf{v_3}}}$ *and* $[z]_{\pi^{\mathbf{v'_3}}}$ *are not glued together by a sequence that alternates* Z*-kind gluings and* U_2*-kind gluings (starting and ending with* Z*) by induction with arguments similar to the ones used in Lemma 135.*

Sketch 140. *Thinking in terms of sequences as in the proofs of Facts 132 and 134, one only has to observe that:*

- $[x]_{\pi^{\mathbf{v_3}}} \neq [y]_{\pi^{\mathbf{v_3}}}$ *for all* $y \in T^{\mathbf{v_3}}$ *in a class that was glued to the class of the* root *via a* $\text{root}_{=,\neg\neq}$*-kind gluing (Use Lemma 121).*

- $[z]_{\pi_1^{\mathbf{u}}} \neq [y]_{\pi_1^{\mathbf{u}}}$ *for all* $y \in T_1^{\mathbf{u}}$ *in a class that was glued to the class of the* root *via a* $\text{root}_{=,\neg\neq}$*-kind gluing (Use Lemma 121).*

- $\text{root}_{=,\neq}$*-kind gluings are made in different subtrees.*

- $[x]_{\pi^{\mathbf{v_3}}}$ *and* $[z]_{\pi_1^{\mathbf{u}}}$ *can not be glued together by a sequence that begins with a* U_2*-kind gluing because we use new subtrees for that kind of gluings.*

- $[x]_{\pi^{\mathbf{v_3}}}$ *and* $[z]_{\pi_1^{\mathbf{u}}}$ *can not be glued together by a sequence that begins with a* Z*-kind gluing because of the consistency of* φ *plus* NeqAx7 *and Lemma 115.*

Sketch 141. *Thinking in terms of sequences as in the proofs of Facts 132 and 134, one only has to observe that:*

- $[x]_{\pi^\psi} \neq [y]_{\pi^\psi}$ *for all* $y \in T^\psi$ *in a class that was glued to the class of the* root *via a* $\text{root}_{=,\neg\neq}$*-kind gluing (Use Lemma 121).*

- $[x]_{\pi^\psi} \neq [y]_{\pi^\psi}$ *for all* $y \in T^\psi$ *in a class that was glued to the class of the* root *via a* $\text{root}_{=,\neq}$*-kind gluing (Rule 1).*

- *By the same arguments given in the proofs of Facts 132 and 134, we can't have a sequence containing any of the following:*

  – $\text{root}_{=,\neq}$-Z,
  – Z-$\text{root}_{=,\neq}$,
  – $\text{root}_{=,\neg\neq}$-U_2,
  – Z-$\text{root}_{=,\neg\neq}$,
  – $\text{root}_{=,\neq}$-U_2,
  – U_2-$\text{root}_{=,\neg\neq}$.
  – $\text{root}_{=,\neg\neq}$-Z,
  – U_2-$\text{root}_{=,\neq}$,

Sketch 142. *Thinking in terms of sequences as in the proofs of Facts 132 and 134, one only has to observe that:*

- *In case* $\psi = \rho$*, by consistency of* φ *plus* EqAx5 *and* **Der21** *of Fact 90,* $\neg\langle\alpha = \beta\rangle$ *is a conjunct of* ψ*.*

- $[x]_{\pi^\psi}$ *and* $[y]_{\pi^\rho}$ *can not be glued together by a sequence of all* Z*-kind gluings because of the consistency of* φ *plus Lemmas 130 and 127.*

- *By Lemma 121 plus construction of* $\mathcal{T}^\varphi$*,* $[y]_{\pi^\rho} \neq [z]_{\pi^\rho}$ *for all* $z \in T^\rho$ *in a class that was glued to the* root *via a* $\text{root}_{=,\neg\neq}$*-kind gluing.*

- *By Rule 1,* $[y]_{\pi^\rho} \neq [z]_{\pi^\rho}$ *for all* $z \in T^\rho$ *in a class that was glued to the* root *via a* $\text{root}_{=,\neq}$*-kind gluing.*

- *By the same arguments given in the proofs of Facts 132 and 134, we can't have a sequence containing any of the following:*

- $\text{root}_{=,\neq}$-Z,
- Z-$\text{root}_{=,\neq}$,
- $\text{root}_{=,\neq}$-U_2,
- U_2-$\text{root}_{=,\neq}$,
- $\text{root}_{=,\neg\neq}$-Z,
- Z-$\text{root}_{=,\neg\neq}$,
- $\text{root}_{=,\neg\neq}$-U_2,
- U_2-$\text{root}_{=,\neg\neq}$,

- *By the same arguments given in the proof of Fact 134, we can reduce sequences with two consecutive Z-kind gluings to sequences that not have two consecutive Z-kind gluings.*
- *By the same arguments given in the proof of Fact 134, we can't have sequences containing U_2-U_2.*
- *One can prove by induction that $[x]_{\pi^\psi}$ and $[y]_{\pi^\rho}$ are not glued together by a sequence that alternates Z-kind gluings and U_2-kind gluings (neither starting with Z or with U_2) with arguments similar to the ones used in Lemma 135.*

(Notation: For $\psi, \rho \in N_n$, we use the notation $\mathcal{T}^\psi = (T^\psi, \pi^\psi), \mathcal{T}^\rho = (T^\rho, \pi^\rho)$ with roots r^ψ, r^ρ respectively to denote any *tree in which ψ, ρ, respectively, are satisfiable.)*

Part B

Computational aspects

Chapter 3

Bisimulations on data graphs

> "They looked so different that I couldn't even consider them my own kind, but in the end their monstrous appearance was just a bluff."
>
> *From the New World*
> Yusuke Kishi

3.1 Introduction

In this chapter, we move from the domain of possibly infinite data trees into the study of bisimulations in the domain of *finite* data *graphs*. Our main focus is calculating the algorithmic complexity of finding bisimilarities, so our restriction to finite structures is natural. Regarding the expansion into graphs, it is partially motivated by data models that have become increasingly important with the continuous growth of the Web and Internet-related applications. It is true that, on the one hand, Web information is usually stored in hierarchical structures, such as the XML format, that can be modeled as tree-structured databases. But on the other hand, vast amounts of information are associated with new applications whose underlying data model is described by (finite) graph-structured databases, such as in the cases of social networks, the Semantic Web, biological systems, network analysis tasks, or crime detection applications.

Semi-structured databases are usually seen as edge-labeled graphs or trees, nodes can be seen as 'entities', containing the actual data (*e.g.*, the name and address in a social network), whereas labeled edges represent 'relations' between these entities (*e.g.*, 'befriends' or 'likes'). Many of the applications making use of this data model have two features in common: on the one hand the underlying data model is described by a graph or a tree, and on the other hand, in querying such graph-structured data, the *topology* of the graph is as important as the *data* itself.

When L is the basic modal logic, the notion of indistinguishability is captured by the bisimulation relation [104], and the classes of equivalence corresponding to this relation can be computed efficiently [8]. Querying graph databases, in general, requires the ability to test properties relative to their topology, reachability and subgraph patterns. One basic query language used for these types of queries is the Regular Path Query, or RPQ, which selects nodes connected by a path described by a regular language over the labeling alphabet [26]. Extensions of this basic query language, such as Propositional Dynamic Logic, have a bisimulation notion akin to that of the basic modal logic and they are thus also amenable to efficient computation of the indistinguishability relation.

However, in many scenarios, these query languages fall short of expressive power, since the actual data contained inside the nodes is completely abstracted away. One standard way of adding data is through the use of a logic such as XPath. XPath has been originally conceived for selecting nodes from XML documents (essentially trees), but its simplicity and modal behavior adapt perfectly to graphs with data, and indeed it has been already studied [78] and used [22] in this scenario.

In this chapter we expand our focus from the universe of data trees into the universe of data graphs. We also transition from our node-labeled logic $\mathrm{XPath}_{=}(\downarrow)$ to the edge-labeled $\mathrm{XPath}_{=}(\downarrow_{\mathbf{a}})$, but we remark that both formalism are intertranslatable, and thus the choice between node-labeled or edge-labeled is not essential. We show that previous results for $\mathrm{XPath}_{=}(\downarrow)$ over node-labeled data trees extend to $\mathrm{XPath}_{=}(\downarrow_{\mathbf{a}})$ over edge-labeled data graphs. We study the computational complexity of the bisimulation notion of $\mathrm{XPath}_{=}$ on (finite) semistructured databases; during this chapter, we restrict ourselves to the domain of *finite* data graphs. For this study to be thorough, we vary, on the one hand, the types of finite structures of the data we analyze: graph, tree, or DAG. On the other hand, we study two different modalities that the logic can have, focusing on the $\mathrm{XPath}_{=}(\downarrow_{\mathbf{a}})$ fragment and then extending our results to $\mathrm{XPath}_{=}(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})$, which adds the possibility of inverse navigation. Finally, we also consider some syntactical restrictions on the formulas, yielding better computational results.

3.1.1 Related work

As we mentioned in §I.1.1, the problem of bisimulation can be solved in PTime for the data-oblivious BML, and it is also polynomially solvable for some fragments of first-order logic where the universe consists of relational models with unary and binary relation symbols (i.e., labeled graphs) [8]. One of the reasons that explains the PTime complexity for these problems is that the respective bisimulation notions are *local*; in BML, this refers to the fact that **Zig** and **Zag** conditions for a pair (u, u') are defined in terms of nodes which are adjacent to u and u', respectively. However, such locality is no longer present in "data-aware" $\mathrm{XPath}_{=}$, as the $\mathbf{Zig}_{=}$ and $\mathbf{Zag}_{=}$ conditions are defined in terms of arbitrarily long paths (i.e., in a *non-local* way). As it turns out, this makes the problem of computing $\mathrm{XPath}_{=}$- bisimulations intractable. It is worth noticing that this is in line with the intractability of other *non-local* notions of bisimulations, such as the *fair bisimulations*

studied in verification [71]. An important point of departure, though, is that such notions are defined with respect to infinite paths in transition systems, while our notion considers finite paths only.

Other place where the complexity of calculating bisimulations has been studied is in [51], in the context of relational models and with the aim of generating short referring expressions. We remark that, even though efficient algorithms for computing bisimulation have been developed in the literature (see, e.g., [39, 62, 91, 61]), we found no available, off-the-shelf tools that can deal with data-aware logics.

Applications

In Chapter 1, we used data-aware bisimulations to study the expressive power of $\mathrm{XPath}_{=}(\downarrow)$ on data trees. When interpreted over data graphs, there are other potential applications for bisimulations:

Indexing. Finding bisimilar nodes over graph-structured data is the first step in many approaches to building indexing data structures for efficient query evaluation of purely navigational languages [87, 41]. Roughly speaking, these approaches are based on the following idea: If x and y are bisimilar and x is in the output of a query, we know that y is also in the output. Extending this to "data-aware" bisimulations might then serve as a building-block over which index structures for $\mathrm{XPath}_{=}(\downarrow_{\mathbf{a}})$ expressions can be constructed.

Clustering. Another motivation stems from the task of *clustering* for mining data graphs [55], that is, the division of data into groups of similar objects. As it pointed out in [14], "*representing the data by fewer clusters necessarily loses certain fine details, but achieves simplification. From a machine learning perspective clusters correspond to hidden patterns, the search for clusters is unsupervised learning, and the resulting system represents a data concept*". Now, one of the most basic analysis one can do on semi-structured databases is the *clustering* of entities, grouping entities which are "similar". One way to define similarity on data graphs is based on *observational indistinguishability*, that is, grouping together elements x, y that cannot be distinguished through a data-aware logic L: $x \equiv_L y$. For the logic $\mathrm{XPath}_{=}(\downarrow_{\mathbf{a}})$ this notion corresponds to "data-aware" bisimilarity. Further, in cases when the previous notion is too strict, it might prove useful to compute a *degree* of similarity, where more similar elements are elements that are distinguished through more complex formulas. This degree of similiarity can be defined, in turn, by restricting suitable parameters in the definition of "data-aware" bisimulations (e.g., the amount of *non-locality* allowed, as studied in this chapter).

Referring expressions generation. A basic and very active task in natural language generation is *referring expressions generation* (REG, see e.g., [51], where referring expressions are generated in the process of calculating bisimulations), which can be stated as follows: given a scene and a target element in that context, generate a grammatically correct expression, called *referring expression* (RE), in a given natural language that

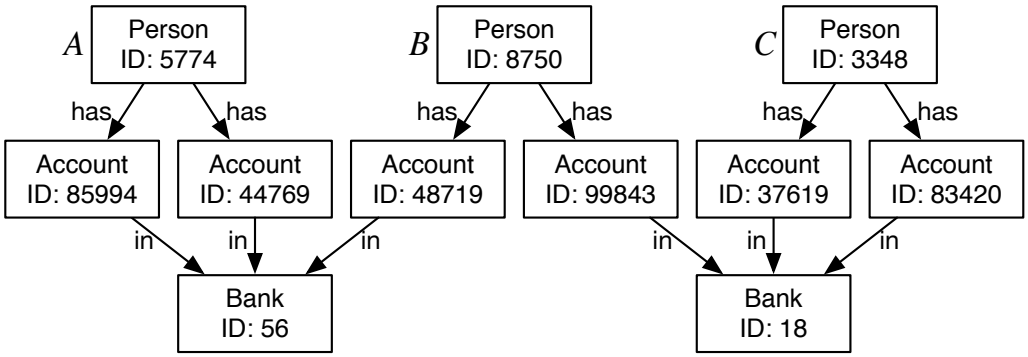


Figure 38: A scene with people, accounts, and banks.

uniquely represents the element. In the early stages, the problem of REG was stated and studied based on a very simple kind of knowledge representation: a purely propositional point of view [27]: given a finite domain of objects (like people in a room) with attributes (like gender, eye color, profession), and a target object or referent, find a set of pairs $\langle \text{attribute}, \text{value} \rangle$ whose conjunction is true only of the target (for instance $\{\langle \text{gender}, \text{F} \rangle, \langle \text{profession}, \text{doctor} \rangle, \langle \text{eye color}, \text{brown} \rangle\}$, which can be later realized as "the woman who is a doctor and has brown eyes"). In [69] it is proposed to use labeled directed graphs for representing the scene, and [7] resort to description logics (DLs) as a formalism for representing a RE. Then, in [8] it is shown that this last approach can be efficiently implemented using the notion of bisimulation. In some cases, though, a scene for the REG problem can be better modeled as a data graph. However, adding data to the models does not mean that we allow data values as constitutive terms of the referring expression. Returning to our example of persons in a room, it is not reasonable to designate the target as "the woman whose ID is 284755" or even "the person whose ID is 284755", even when data of person's ID is present in the graph. At the logical level, this means that we do not have at our disposal data values as terms in the language. However, the presence of data in the graph can still be relevant for constructing referring expressions which are more expressive than the ones considered in the work of [7]. Imagine, e.g., a scene modeling clients, accounts, and banks, as the one in Figure 38. Each object has an ID. Suppose we look for a RE for target B. One can see that it is impossible to distinguish nodes A and B using Modal Logic or the DLs used in previous works, since they are bisimilar (assuming, of course, that IDs are not part of the language). However, the RE "the person who has accounts in different banks" can be formalized in $\text{XPath}_{=}(\downarrow_{\mathbf{a}})$[15]. Extending [8], "data-aware" bisimulations might then be an efficient tool for REG in cases when REs are expressed in the language of $\text{XPath}_{=}(\downarrow_{\mathbf{a}})$.

[15]The labels 'Person', 'Account', and 'Bank' can be omitted in the data graph representation of this scene. In the resulting edge-labeled data graph, the desired expression is: $\langle \downarrow_{has} \downarrow_{in} \neq \downarrow_{has} \downarrow_{in} \rangle$.

3.1.2 Contributions

In this chapter we transition from our node-labeled logic $\text{XPath}_=(\downarrow)$ to the equivalent edge-labeled $\text{XPath}_=(\downarrow_{\mathbf{a}})$, We also transition from the universe of data trees to that of data *graphs*, and we show that the bisimulation notion for $\text{XPath}_=(\downarrow)$ can be easily adapted to work on this wider context of $\text{XPath}_=(\downarrow_{\mathbf{a}})$ over data graphs. As it has been recently shown in the context of data trees, $\text{XPath}_=(\downarrow)$ allows for a Hennessy-Milner's style characterization in terms of a natural class of "data-aware" bisimulations [45]. We notice in this chapter that such characterization extends in a straightforward way to the class of data graphs.

Our main contribution is an in-depth study of the complexity of computing "data-aware" bisimulations by fine-tuning on the level of *non-locality* allowed. This non-locality is measured in terms of (a) the lengths of the paths considered in the definition of bisimulation, and (b) the classes of models over which bisimulations are computed. In particular, we show the following:

- In its full generality, the decision problem of whether two data graphs are "data-aware" bisimilar is PSPACE-COMPLETE. This is obtained by showing that our problem is polynomially equivalent to *equivalence* of nondeterministic finite automata, which is known to be PSPACE-COMPLETE [83]. We obtain, in particular, that there are cases in which the smallest witness (π_1, π_2) to the fact that two data graphs are not bisimilar is a pair of paths of exponential size.

- The previous observation naturally calls for a restriction on the length of paths to be inspected in the definition of "data-aware" bisimulation (restriction (a) above). We start by considering paths of polynomial length only. While this decreases the complexity of the problem to the class CO-NP, we show that it still does not yield tractability.

  We thus restrict to paths of constant length only and show that this condition does guarantee tractability. Interestingly, this restricted notion of bisimulation characterizes an important fragment of the $\text{XPath}_=(\downarrow_{\mathbf{a}})$ language; namely, the one of *bounded length*. This fragment restricts the length of expressions α_1 and α_2 in formulas of the form $\langle \alpha_1 = \alpha_2 \rangle$ and $\langle \alpha_1 \neq \alpha_2 \rangle$ only (it allows for arbitrary long path expressions α in purely navigational expressions $\langle \alpha \rangle$).

- We then switch to study how the underlying graphs affect the complexity of computing bisimulations (restriction (b) above), and concentrate on the two most important classes of acyclic graphs: trees and DAGs. We show that checking "data-aware" bisimilarity is in polynomial time for the former and CO-NP-COMPLETE for the latter. Thus, acyclicity yields tractability in this case, but only if defined at the level of the underlying *undirected* graphs.

- Finally, we look at $\text{XPath}_=(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})$, which allows to traverse edges in both directions. We show that the problem of checking "data aware" bisimilarity in this context

remains PSPACE-COMPLETE. While the upper bound follows easily, for the lower bound we need a new proof. As before, the restriction to paths of polynomial length yields a CO-NP bound, and for paths of constant length we obtain tractability.

3.1.3 Organization

While the notion of extending $\text{XPath}_=(\downarrow)$ to edge-labeled data graphs has been mentioned in §I.1.2, we give the proper definition of $\text{XPath}_=(\downarrow_{\mathbf{a}})$-bisimulations in §3.2, where we also mention how results for $\text{XPath}_=(\downarrow)$ naturally extend to this framework.

The complexity of $\text{XPath}_=(\downarrow_{\mathbf{a}})$-bisimulations is studied in §3.3, where we give tight upper bounds on the complexity of the $\text{XPath}_=(\downarrow_{\mathbf{a}})$-bisimilarity and the $\text{XPath}_=(\downarrow_{\mathbf{a}})$-similarity decision problems. Restrictions on the paths considered in the conditions of $\text{XPath}_=(\downarrow_{\mathbf{a}})$-bisimulations are presented in §3.4, where we give a notion of function-bounded bisimulations which can only compare data values that are at most at a certain depth determined by the chosen function and the size of the graphs. Restrictions on the (finite) data models are considered in §3.5, where we analyze the effects on restricting the models to acyclic graphs and to trees. The two-way $\text{XPath}_=(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})$ is studied in §3.6, where we indicate how the previous results for $\text{XPath}_=(\downarrow_{\mathbf{a}})$ can help to solve the corresponding problems for this fragment that is provided with inverse navigation.

3.2 Bisimulations on data graphs

As we mentioned in §I.2, the notion of (bi)simulation for $\text{XPath}_=$ over *node-labeled data trees* was developed in [44] and later extended in [2]. In this chapter, we will make three changes to that framework:

- First, the change from $\text{XPath}_=(\downarrow)$ into the *edge-labeled* logic $\text{XPath}_=(\downarrow_{\mathbf{a}})$. Now the nodes only contain data values (or their equivalence classes), but each edge also includes a single label. To make this information accessible, we exchange the $\downarrow$ symbol and each label symbol a for $\downarrow_a$, which indicates that we descend from the current node to a children via an edge labeled a. The formal syntax and semantics were stated in §I.1.2. In Remark 143 we indicate that this edge-labeled formalism is equivalent to the node-labeled one.

- Second, we expand into the universe of *data graphs*. While this expansion could require fundamental changes in the nature of bisimulation (in order to conserve the results of Hennessy-Milner-style characterization), this is not really the case: the notion is robust and extends in a straightforward way to data graphs (for both $\text{XPath}_=(\downarrow)$ and $\text{XPath}_=(\downarrow_{\mathbf{a}})$).

- Third, since we are interested in complexity analysis for bisimilarity between data graphs, we restrict ourselves to *finite* data graphs.

Remark 143. *Each node-labeled data graph $\mathcal{G}$ can be represented as an edge-labeled data graph $\mathcal{G}'$ in a bisimilarity-preserving way, and vice versa.*

Proof. For the node-labeled to edge-labeled direction we proceed as follows: If a node x in $\mathcal{G}$ is labeled a and has outgoing edges, we label all such outgoing edges with a in $\mathcal{G}'$. Else, if node x does not have outgoing edges, we simply add a fresh node x' and an edge labeled a from x to x'. We also set $data(x) = data(x')$. It can be proved that bisimilarity is invariant under this translation. More formally, nodes x_1 and x_2 in node-labeled graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are XPath$_=(\downarrow)$-bisimilar if and only if they are bisimilar in their edge-labeled representations $\mathcal{G}_1'$ and $\mathcal{G}_2'$ in terms of the definition of XPath$_=(\downarrow_{\mathbf{a}})$-bisimulation given below in Definition 144.

For the edge-labeled to node-labeled direction, we proceed as follows: For each edge label $x \xrightarrow{a} y$, we add an additional node z between x and y, such that $x \to z \to y$, and z is labeled with a. All these additional nodes have the same data value, which is fresh respect to the data values of the original graph. Finally, we label all the nodes of the original graph with a fresh label c. Again, this translation preserves bisimilarity or non-bisimilarity between the notions.

See Figure 39 for a graphical representation of these translations between the edge-labeled and node-labeled frameworks.

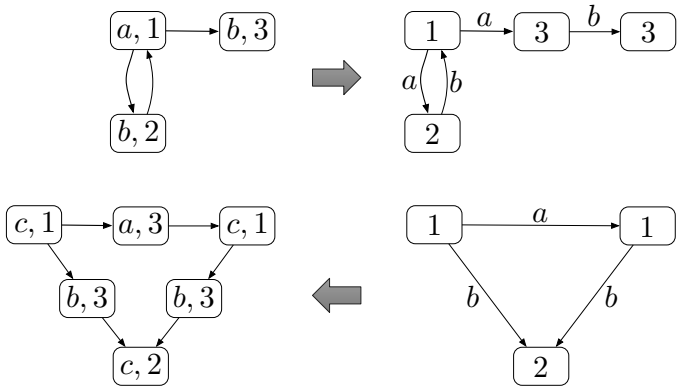


Figure 39: Examples of how to change from a node-labeled graph to an edge-labeled graph, and vice versa.

□

We now proceed to give the definitions of simulation and bisimulation for XPath$_=(\downarrow_{\mathbf{a}})$:

Definition 144 (XPath$_=(\downarrow_{\mathbf{a}})$-(bi)simulations)**.** Let $\mathcal{G}$ and $\mathcal{G}'$ be (finite or infinite) data graphs over $\mathbb{A}$. A relation $Z \subseteq G \times G'$ is said to be an **XPath$_=(\downarrow_{\mathbf{a}})$-bisimulation** if for all $(x, x') \in G \times G'$ such that xZx' we have:

- (**Zig$_=$**) For every pair of paths in $\mathcal{G}$ of the form

$$\pi_1 = x \xrightarrow{e_1} x_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} x_n \text{ and } \pi_2 = x \xrightarrow{d_1} y_1 \xrightarrow{d_2} \ldots \xrightarrow{d_m} y_m, \quad \text{where } e_i, d_j \in \mathbb{A},$$

there are paths in $\mathcal{G}'$ of the form

$$\pi'_1 = x' \xrightarrow{e_1} x'_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} x'_n \text{ and } \pi'_2 = x' \xrightarrow{d_1} y'_1 \xrightarrow{d_2} \ldots \xrightarrow{d_m} y'_m,$$

such that the following holds:

1. $x_i Z x'_i$ for all $1 \leq i \leq n$, and $y_j Z y'_j$ for all $1 \leq j \leq m$.
2. $data(x_n) = data(y_m) \Leftrightarrow data(x'_n) = data(y'_m)$.

- (**Zag**$_=$) For every paths in $\mathcal{G}'$ of the form

$$\pi'_1 = x' \xrightarrow{e_1} x'_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} x'_n \text{ and } \pi'_2 = x' \xrightarrow{d_1} y'_1 \xrightarrow{d_2} \ldots \xrightarrow{d_m} y'_m, \quad \text{where } e_i, d_j \in \mathbb{A},$$

there are paths in $\mathcal{G}$ of the form

$$\pi_1 = x \xrightarrow{e_1} x_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} x_n \text{ and } \pi_2 = x \xrightarrow{d_1} y_1 \xrightarrow{d_2} \ldots \xrightarrow{d_m} y_m$$

such that conditions 1 and 2 above are verified.

Furthermore, a **XPath$_=$($\downarrow_\mathbf{a}$)-simulation** is a relation $Z \subseteq G \times G'$ such that for all $(x, x') \in G \times G'$ such that xZx', it is the case that condition **Zig**$_=$ above is verified.

Observe that this notion of bisimulation lacks the **Harmony** clause that was present in XPath$_=$($\downarrow$)-bisimulation. The reason is that the function of that condition is now subsumed into **Zig**$_=$ and **Zag**$_=$, as they also check for the correct mirroring of the labels in the edges. Another remark is that in the universe of data graphs **Zig**$_=$ and **Zag**$_=$ must necessarily specify the whole paths, unlike the case for XPath$_=$($\downarrow$) over data trees, where it was enough to indicate the starting and the ending nodes of the paths.

We now define when $u \in G$ and $u' \in G'$ are said to be **XPath$_=$($\downarrow_\mathbf{a}$)-similar** and **XPath$_=$($\downarrow_\mathbf{a}$)-bisimilar**, respectively notated $\mathcal{G}, u \rightrightarrows^{\downarrow_\mathbf{a}} \mathcal{G}', u'$ and $\mathcal{G}, u \leftrightarrows^{\downarrow_\mathbf{a}} \mathcal{G}', u'$:

$$\mathcal{G}, u \rightrightarrows^{\downarrow_\mathbf{a}} [\leftrightarrows^{\downarrow_\mathbf{a}}] \mathcal{G}', u' \overset{\text{def}}{\Leftrightarrow} \text{ there is an XPath}_=(\downarrow_\mathbf{a})\text{-[bi]simulation } Z \text{ such that } (u, u') \in Z.$$

We remark that XPath$_=$($\downarrow_\mathbf{a}$)-(bi)simulations are closed under union: if $Z_1 \subseteq G \times G'$ and $Z_2 \subseteq G \times G'$ are XPath$_=$($\downarrow_\mathbf{a}$)-(bi)simulations between $u \in G$ and $u' \in G$, then so is $Z_1 \cup Z_2$. This immediately implies the following:

Proposition 145. *If there is an XPath$_=$($\downarrow_\mathbf{a}$)-[bi]simulation between $u \in G$ and $u' \in G$, then there is a maximal such XPath$_=$($\downarrow_\mathbf{a}$)-[bi]simulation.*

The characterization. One can verify that the following theorem, originally stated in terms of data trees, holds also in the general case of data graphs. It establishes the desired, Hennessy-Milner-style characterization of the notion of logical indistinguishability for XPath$_=$($\downarrow_\mathbf{a}$) in terms of XPath$_=$($\downarrow_\mathbf{a}$)-(bi)simulations.

Before stating the theorem, we give for our logic XPath$_=(\downarrow_{\mathbf{a}})$ the expected definitions of **equivalence** (notated $\equiv^{\downarrow_{\mathbf{a}}}$), and its partial, one-direction version (notated $\Rrightarrow^{\downarrow_{\mathbf{a}}}$). Recall that a node expression is said to be positive if it contains no negation.

$\mathcal{G}, u \equiv^{\downarrow_{\mathbf{a}}} \mathcal{G}', u' \overset{\text{def}}{\Leftrightarrow}$ for any XPath$_=(\downarrow_{\mathbf{a}})$-node expression φ, $\mathcal{G}, u \models \varphi$ iff $\mathcal{G}', u' \models \varphi$.

$\mathcal{G}, u \Rrightarrow^{\downarrow_{\mathbf{a}}} \mathcal{G}', u' \overset{\text{def}}{\Leftrightarrow}$ for any positive XPath$_=(\downarrow_{\mathbf{a}})$-node expression φ, if $\mathcal{G}, u \models \varphi$ then $\mathcal{G}', u' \models \varphi$.

Theorem 146. *Let $\mathcal{G}$ and $\mathcal{G}'$ be (finite) data graphs over the same alphabet $\mathbb{A}$, and u and u' nodes in G and G', respectively. Then $\mathcal{G}, u \equiv^{\downarrow_{\mathbf{a}}} \mathcal{G}', u'$ iff $\mathcal{G}, u \leftrightarrows^{\downarrow_{\mathbf{a}}} \mathcal{G}', u'$, and $\mathcal{G}, u \Rrightarrow^{\downarrow_{\mathbf{a}}} \mathcal{G}', u'$ iff $\mathcal{G}, u \rightrightarrows^{\downarrow_{\mathbf{a}}} \mathcal{G}', u'$.*

Proof. We only prove the theorem for the case of bisimulation, as for simulation the proof is analogous. The idea is to reduce the case of data graphs to data trees by unravelling, and then use the result of [45].

Let $\mathcal{G}$ be a data graph and u a node in G. We define $\widetilde{\mathcal{G}}_u$ as the data graph that corresponds to the *unraveling* of $\mathcal{G}$ from node u. The idea is that nodes of $\widetilde{\mathcal{G}}_u$ are finite sequences of nodes from G representing paths of $\mathcal{G}$ starting at u. Formally, the set of nodes of $\widetilde{\mathcal{G}}_u$ corresponds to the least subset $\widetilde{G}_u$ of G^* such that (i) $u \in \widetilde{G}_u$, and (ii) for any $\sigma \in G^*$ and $x, y \in G$ it is the case that

$$\sigma xy \in \widetilde{G}_u \quad \Longleftrightarrow \quad \sigma x \in \widetilde{G}_u \text{ and } x \xrightarrow{a} y \text{ is an edge of } \mathcal{G}, \text{ for some label } a \in \mathbb{A}.$$

The set of edges of $\widetilde{\mathcal{G}}_u$ is defined as follows: For $x, y \in G$, $\sigma \in G^*$, and a label a we have

$$\sigma x \xrightarrow{a} \sigma xy \in \widetilde{\mathcal{G}}_u \quad \Longleftrightarrow \quad \sigma x \in \widetilde{G}_u \text{ and } x \xrightarrow{a} y \in \mathcal{G}.$$

Finally, the data value of a node σx in $\widetilde{\mathcal{G}}_u$ is the data value of x in $\mathcal{G}$. Observe that the unraveling of a data graph is thus a possibly infinite (but finitely branching) data tree.

It can be seen that $\mathcal{G}, u \leftrightarrows^{\downarrow_{\mathbf{a}}} \widetilde{\mathcal{G}}_u, u$; in particular, that the relation that contains all pairs of the form $(x, \sigma x) \in G \times \widetilde{G}_u$ is an XPath$_=(\downarrow_{\mathbf{a}})$-bisimulation between u in $\mathcal{G}$ and u in $\widetilde{\mathcal{G}}_u$. On the other hand, it can also be proved that $\mathcal{G}, u \equiv^{\downarrow_{\mathbf{a}}} \widetilde{\mathcal{G}}_u, u$. In fact, one can show that $\mathcal{G}, u \models \varphi$ iff $\widetilde{\mathcal{G}}_u, \sigma u \models \varphi$ for every formula φ in XPath$_=(\downarrow_{\mathbf{a}})$. The proof goes by induction on the structural complexity of φ. The key points are the following:

- Let $x_0 \xrightarrow{e_1} x_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} x_n$ be a path in $\mathcal{G}$ such that x_0 is reachable form u. Then from every $\sigma_0 \in \widetilde{G}_u$ whose last symbol is x_0 there is a path $\sigma_0 \xrightarrow{e_1} \sigma_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} \sigma_n$ in $\widetilde{\mathcal{G}}_u$ such that the data value of x_i coincides with that of σ_i, for each $0 \leq i \leq n$.

- Analogously, for any path $\sigma_0 \xrightarrow{e_1} \sigma_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} \sigma_n$ in $\tilde{\mathcal{G}}_u$ there is a path $x_0 \xrightarrow{e_1} x_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} x_n$ in $\mathcal{G}$, where x_0 (reachable from u) is the last symbol of σ_0, and such that the data value of x_i coincides with that of σ_i for each $0 \leq i \leq n$.

To obtain the desired result, we lift our data trees $\mathcal{G}, u$ and $\mathcal{G}', u'$ to their corresponding unravelings and then use Theorem 8 in [45], which states the coincidence of logical equivalence and bisimulation over possibly infinite but finitely branching data trees. □

It is worth remarking that adding the transitive reflexive closure $\downarrow_a^*$ of $\downarrow_a$ to $\text{XPath}_=(\downarrow_\mathbf{a})$ does not change the notion of (bi)simulation, and, in particular, Theorem 146 continues to hold (when $\equiv^{\downarrow_\mathbf{a}}$ and $\Rrightarrow^{\downarrow_\mathbf{a}}$ are replaced with the corresponding indistinguishability notion in the extended language).

3.3 Computing $\text{XPath}_=(\downarrow_\mathbf{a})$-bisimulations

When dealing with (bi)simulations in a practical context, a fundamental problem is that of checking whether a pair of nodes is (bi)similar. In this section we study the complexity of such problem for $\text{XPath}_=(\downarrow_\mathbf{a})$-(bi)simulations and show it to be PSPACE-COMPLETE. This establishes an important difference with the problem of computing bisimulations in the absence of data, which can be solved in polynomial time, as it is essentially equal to computing bisimulations for the basic modal logic.

Formally, we study the following problems:

$\text{XPath}_=(\downarrow_\mathbf{a})$-[Bi]Similarity problem	
INPUT :	Data graphs $\mathcal{G}$ and $\mathcal{G}'$, nodes $u \in G$ and $u' \in G'$.
OUTPUT :	'Yes' iff $\mathcal{G}, u \rightrightarrows^{\downarrow_\mathbf{a}} \mathcal{G}', u'$ [iff $\mathcal{G}, u \leftrightarrows^{\downarrow_\mathbf{a}} \mathcal{G}', u'$, resp.]

Our main result establishes the following:

Theorem 147. *The problems of* $\text{XPath}_=(\downarrow_\mathbf{a})$-BISIMILARITY *and* $\text{XPath}_=(\downarrow_\mathbf{a})$-SIMILARITY *are* PSPACE-COMPLETE.

The proof of this theorem is given later, and comprises Subsection 3.3.1, where we show that the problem is PSPACE, and Subsection 3.3.2, where we demonstrate PSPACE-hardness. Before starting with this proof, we first make some observations related with this theorem and its consequences.

From Theorem 147 and Theorem 146 we obtain:

Corollary 148. *The problem of checking* $\mathcal{G}, u \equiv^{\downarrow_\mathbf{a}} \mathcal{G}', u'$ *or* $\mathcal{G}, u \Rrightarrow^{\downarrow_\mathbf{a}} \mathcal{G}', u'$*, given data graphs* $\mathcal{G}$ *and* $\mathcal{G}'$ *and nodes* $u \in G$ *and* $u' \in G'$*, is* PSPACE-COMPLETE.

Furthermore, the proof of Theorem 147 will imply that the PSPACE lower bound is quite resilient, as it holds even when checking indistinguishability for the restricted class of formulas of the form $\langle \varepsilon = \downarrow_{e_1} \ldots \downarrow_{e_n} \rangle$, for $e_1 \ldots e_n \in \mathbb{A}$. These formulas simply check if starting from a node u it is possible to follow a path labeled $e_1 \ldots e_n$ and reach a node with the same data value as u. Formally, let us call $\text{XPath}_=^{\text{paths}}(\downarrow_\mathbf{a})$ the fragment that only contains formulas of the type $\langle \varepsilon = \downarrow_{e_1} \ldots \downarrow_{e_n} \rangle$. We then say that $\mathcal{G}, u \equiv_{\text{paths}}^{\downarrow_\mathbf{a}} \mathcal{G}', u'$ if $\mathcal{G}, u$ and $\mathcal{G}', u'$ are indistinguishable with respect to $\text{XPath}_=^{\text{paths}}(\downarrow_\mathbf{a})$; that is:

$$\mathcal{G}, u \equiv^{\downarrow_{\mathbf{a}}}_{\text{paths}} \mathcal{G}', u' \quad \overset{\text{def}}{\Leftrightarrow} \quad \begin{array}{l}\text{for every } e_1, \ldots, e_n \in \mathbb{A}: \\ \mathcal{G}, u \models \langle \varepsilon = \downarrow_{e_1} \ldots \downarrow_{e_n} \rangle \text{ iff } \mathcal{G}', u' \models \langle \varepsilon = \downarrow_{e_1} \ldots \downarrow_{e_n} \rangle.\end{array}$$

Analogously, we define $\mathcal{G}, u \Rrightarrow^{\downarrow_{\mathbf{a}}}_{\text{paths}} \mathcal{G}', u'$. We then obtain the following:

Corollary 149. *Checking* $\mathcal{G}, u \equiv^{\downarrow_{\mathbf{a}}}_{\text{paths}} \mathcal{G}', u'$ *or* $\mathcal{G}, u \Rrightarrow^{\downarrow_{\mathbf{a}}}_{\text{paths}} \mathcal{G}', u'$ *is* PSPACE-COMPLETE.

The proof of Theorem 147 shows that (essentially) XPATH$_=$($\downarrow_{\mathbf{a}}$)-[BI]SIMILARITY is polynomially equivalent to the *containment problem* [resp., *equivalence problem*] for non-deterministic finite automata (NFAs). Although the proof is not very involved, both directions are non-trivial, and, in particular, the reduction from containment to bisimilarity requires a clever encoding. Below, we briefly recall basic notions on NFA and its containment and equivalence problems.

Basics on finite automata. Recall that an NFA over a finite alphabet $\mathbb{A}$ is given as a tuple $\mathcal{A} = (Q, q_0, F, \delta)$, where Q is a finite set of *states*, $q_0 \in Q$ is the *initial* state, $F \subseteq Q$ is the set of *final* states, and $\delta \subseteq Q \times \mathbb{A} \times Q$ is the *transition relation*. A word $w \in \mathbb{A}^*$ is *accepted* by $\mathcal{A}$ if there is an *accepting run* of $\mathcal{A}$ over w that respects the transition relation δ. Formally, if $w = a_1 \ldots a_n$, where each a_i is a symbol in $\mathbb{A}$, an *accepting run* of $\mathcal{A}$ over w is a sequence $q_0 \ldots q_n$ of states in Q such that (i) $(q_i, a_{i+1}, q_{i+1}) \in \delta$ for each $0 \leq i < n$, and (ii) $q_n \in F$. (Notice that the first state of this accepting run corresponds to the initial state of $\mathcal{A}$). The set of words in $\mathbb{A}^*$ that are accepted by $\mathcal{A}$ is the *language* of $\mathcal{A}$, denoted with $L(\mathcal{A})$.

The **containment problem** for NFAs is defined as follows: Given NFAs $\mathcal{A}_1$ and $\mathcal{A}_2$ over $\mathbb{A}$, is $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$? Respectively, we define the **equivalence problem** for NFAs, but this time we ask whether $L(A_1) = L(A_2)$. Both containment and equivalence of NFAs are PSPACE-COMPLETE [83]. We prove Theorem 147 below by reductions to and from these problems.

3.3.1 Upper bound

We start by explaining how to obtain a PSPACE upper bound for XPATH$_=$($\downarrow_{\mathbf{a}}$)-SIMILARITY. The algorithm receives a pair of graphs $\mathcal{G}, \mathcal{G}'$ and a pair of nodes u, u', it guesses a candidate relation $Z \subseteq G \times G'$ containing (u, u'), and then checks that Z satisfies the **Zig**$_=$ property. Since Z is of polynomial size and PSPACE = NPSPACE from Savitch's Theorem [98], i.e., PSPACE is closed under non-determinism, we only need to show that the latter can be checked in PSPACE. This is done by reducing the problem in polynomial time to containment of NFAs. Since NFA containment is in PSPACE the result follows (as PSPACE-computable functions are closed under composition).

Let us explain now the reduction to containment of NFAs. Given a node $x \in G$, we construct (details below) in polynomial time an NFA $\mathcal{A}_{\mathcal{G},x}$ over the alphabet $\mathbb{A} \times G \cup \{=, \neq\}$ that accepts precisely the language L of words of the form:

$$(e_1, x_1) \ldots (e_n, x_n) \star (f_1, y_1) \ldots (f_m, y_m), \tag{21}$$

for $(e_i, x_i), (f_j, y_j) \in \mathbb{A} \times G$ and $\star \in \{=, \neq\}$, such that $\mathcal{G}$ contains paths:

$$x \xrightarrow{e_1} x_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} x_n \quad \text{and} \quad x \xrightarrow{f_1} y_1 \xrightarrow{f_2} \ldots \xrightarrow{f_m} y_m$$

for which $data(x_n) \star data(y_m)$.

We also construct (details below) an NFA $\mathcal{A}^{\mathcal{G},Z}_{\mathcal{G}',x'}$ over the alphabet $\mathbb{A} \times G \cup \{=, \neq\}$ that accepts precisely the language L' of words of the form (21) such that $\mathcal{G}'$ contains paths:

$$x' \xrightarrow{e_1} x'_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} x'_n \quad \text{and} \quad x' \xrightarrow{d_1} y'_1 \xrightarrow{d_2} \ldots \xrightarrow{d_m} y'_m,$$

and the following holds:

- $x_i Z x'_i$ for each $1 \leq i \leq n$, and $y_j Z y'_j$ for each $1 \leq j \leq m$.
- $data(x'_n) \star data(y'_m)$.

Finally, we verify (details below) that

$$Z \text{ satisfies } \mathbf{Zig}_{=} \iff L(\mathcal{A}_{\mathcal{G},x}) \subseteq L(\mathcal{A}^{\mathcal{G},Z}_{\mathcal{G}',x}), \text{ for each } (x, x') \in Z. \tag{22}$$

Since Z is of polynomial size, Equation (22) tells us that in order to check whether $\mathbf{Zig}_{=}$ holds we only need to check containment for a polynomial number of pairs of NFAs. This establishes the upper bound since each such containment can be checked in PSPACE.

The proof for bisimilarity is analogous. In fact, Z satisfies $\mathbf{Zig}_{=}$ and $\mathbf{Zag}_{=}$ if and only if for each $(x, x') \in Z$ it is the case that:

$$L(\mathcal{A}_{\mathcal{G},x}) \subseteq L(\mathcal{A}^{\mathcal{G},Z}_{\mathcal{G}',x'}) \text{ and } L(\mathcal{A}_{\mathcal{G}',x'}) \subseteq L(\mathcal{A}^{\mathcal{G}',Z^{-1}}_{\mathcal{G},x}).$$

This can clearly be checked in PSPACE.

In what follows, we give the definitions of $\mathcal{A}_{\mathcal{G},x}$ and $\mathcal{A}^{\mathcal{G},Z}_{\mathcal{G}',x'}$, and the verification that (22) holds.

Definition of $\mathcal{A}_{\mathcal{G},x}$. The set of states of $\mathcal{A}_{\mathcal{G},x}$ is defined by:

$$\{y,\ y^{=}_d,\ y^{\neq}_d \mid y \in G \text{ and } d \text{ is the data value of some node in } \mathcal{G}\}.$$

The initial state is x and the set of final states corresponds to:

$$\{y^{=}_d \mid data(y) = d\} \ \cup \ \{y^{\neq}_d \mid data(y) \neq d\}.$$

Finally, the transition relation of $\mathcal{A}_{\mathcal{G},x}$ corresponds to the union of the following sets:

1. $\{(y,\ (e, z)\ , z) \mid (y \xrightarrow{e} z) \text{ is an edge in } \mathcal{G}\}$.
2. $\{(y^{=}_d,\ (e, z)\ , z^{=}_d) \mid (y \xrightarrow{e} z) \text{ is an edge in } \mathcal{G} \text{ and } d \text{ is a data value in } \mathcal{G}\}$.
3. $\{(y^{\neq}_d,\ (e, z)\ , z^{\neq}_d) \mid (y \xrightarrow{e} z) \text{ is an edge in } \mathcal{G} \text{ and } d \text{ is a data value in } \mathcal{G}\}$.

4. $\{(y,\ (=)\,, x_d^{=})\ \mid\ data(y) = d\}$.
5. $\{(y,\ (\neq)\,, x_d^{\neq})\ \mid\ data(y) = d\}$.

Clearly, $\mathcal{A}_{\mathcal{G},x}$ can be constructed in polynomial time from $\mathcal{G}$. We prove next that $L(\mathcal{A}_{\mathcal{G},x}) = L$, where L is the language defined above.

Notice first that any accepting run of $\mathcal{A}_{\mathcal{G},x}$ must be a sequence of states of the form:

$$x\, x_1\ \ldots\ x_n\ x_d^{\star}\, (y_1)_d^{\star}\ \ldots\ (y_m)_d^{\star}.$$

By definition, the word w accepted by this run is of the form:

$$(e_1, x_1) \ldots (e_n, x_n) \star (f_1, y_1) \ldots (f_m, y_m),$$

for $e_i, f_j \in \mathbb{A}$ and $\star \in \{=, \neq\}$, and it is the case that $\mathcal{G}$ contains paths

$$x \xrightarrow{e_1} x_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} x_n \quad \text{and} \quad x \xrightarrow{f_1} y_1 \xrightarrow{f_2} \ldots \xrightarrow{f_m} y_m$$

such that $data(x_n) = d$ if and only if $\star$ corresponds to the symbol $=$. Since $(y_m)_d^{\star}$ is an accepting state, it must be the case then that

$$data(x_n) = data(y_m) \iff \star \text{ corresponds to the symbol } =.$$

We conclude that $data(x_n) \star data(y_m)$, and therefore that $w \in L$. This tells us that $L(\mathcal{A}_{\mathcal{G},x}) \subseteq L$ since w was chosen arbitrarily.

On the other hand, consider a word w in L of the form:

$$(e_1, x_1) \ldots (e_n, x_n) \star (f_1, y_1) \ldots (f_m, y_m).$$

By definition, $\mathcal{G}$ contains paths:

$$x \xrightarrow{e_1} x_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} x_n \quad \text{and} \quad x \xrightarrow{f_1} y_1 \xrightarrow{f_2} \ldots \xrightarrow{f_m} y_m$$

for which $data(x_n) \star data(y_m)$. It is easy to see then that

$$x\, x_1\ \ldots\ x_n\ x_d^{\star}\, (y_1)_d^{\star}\ \ldots\ (y_m)_d^{\star},$$

for $d = data(x_n)$, is an accepting run of $\mathcal{A}_{\mathcal{G},x}$ over w. Thus, $w \in L(\mathcal{A}_{\mathcal{G},x})$, and, therefore, $L(\mathcal{A}_{\mathcal{G},x}) \subseteq L$ since w was chosen arbitrarily.

Definition of $\mathcal{A}^{\mathcal{G},Z}_{\mathcal{G}',x'}$. The set of states of $\mathcal{A}^{\mathcal{G},Z}_{\mathcal{G}',x'}$ is defined by:

$$\{(y,y'),\ (y,y')_d^{=},\ (y,y')_d^{\neq}\ \mid\ (y,y') \in G \times G' \text{ and } d \text{ is the data value of some node in } \mathcal{G}'\}.$$

The initial state is (x, x') and the set of final states corresponds to:

$$\{(y,y')_d^{=}\ \mid\ data(y') = d\}\ \cup\ \{(y,y')_d^{\neq}\ \mid\ data(y') \neq d\}.$$

Finally, the transition relation of $\mathcal{A}^{\mathcal{G},Z}_{\mathcal{G}',x'}$ corresponds to the union of the following sets:

1. $\{\big((y,y'),(e,z),(z,z')\big) \mid (y \xrightarrow{e} z)$ is an edge in $\mathcal{G}$, $(y' \xrightarrow{e} z')$ is an edge in $\mathcal{G}'$, and $zZz'\}$.
2. $\{\big((y,y')^{=}_{d}, (e,z), (z,z')^{=}_{d}\big) \mid (y \xrightarrow{e} z)$ is an edge in $\mathcal{G}$, $(y' \xrightarrow{e} z')$ is an edge in $\mathcal{G}'$, zZz', and d is a data value in $\mathcal{G}'\}$.
3. $\{\big((y,y')^{\neq}_{d}, (e,z), (z,z')^{\neq}_{d}\big) \mid (y \xrightarrow{e} z)$ is an edge in $\mathcal{G}$, $(y' \xrightarrow{e} z')$ is an edge in $\mathcal{G}'$, zZz', and d is a data value in $\mathcal{G}'\}$.
4. $\{\big((y,y'), (=), (x,x')^{=}_{d}\big) \mid data(y') = d\}$.
5. $\{\big((y,y'), (\neq), (x,x')^{\neq}_{d}\big) \mid data(y') = d\}$.

Clearly, $\mathcal{A}^{\mathcal{G},Z}_{\mathcal{G}',x'}$ can be constructed in polynomial time from $\mathcal{G}$. It is also quite easy to prove that $L(\mathcal{A}^{\mathcal{G},Z}_{\mathcal{G}',x'}) = L'$

Verification. Note that Z satisfies $\mathbf{Zig}_{=}$ if and only if for every $(x,x') \in Z$ and paths in $\mathcal{G}$ of the form

$$\pi_1 = x \xrightarrow{e_1} x_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} x_n \quad \text{and} \quad \pi_2 = x \xrightarrow{d_1} y_1 \xrightarrow{d_2} \ldots \xrightarrow{d_m} y_m, \quad \text{where } e_i, d_j \in \mathbb{A},$$

there are paths in $\mathcal{G}'$ of the form

$$\pi'_1 = x' \xrightarrow{e_1} x'_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} x'_n \quad \text{and} \quad \pi'_2 = x' \xrightarrow{d_1} y'_1 \xrightarrow{d_2} \ldots \xrightarrow{d_m} y'_m,$$

such that the following holds:

1. $x_i Z x'_i$ for all $1 \leq i \leq n$, and $y_j Z y'_j$ for all $1 \leq j \leq m$.
2. $data(x_n) = data(y_m) \Leftrightarrow data(x'_n) = data(y'_m)$.

In other words, Z satisfies $\mathbf{Zig}_{=}$ if and only if for every $(x,x') \in Z$ and word over $\mathbb{A} \times G \cup \{=, \neq\}$ of the form (21) such that $\mathcal{G}$ contains paths:

$$x \xrightarrow{e_1} x_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} x_n \quad \text{and} \quad x \xrightarrow{f_1} y_1 \xrightarrow{f_2} \ldots \xrightarrow{f_m} y_m$$

for which $data(x_n) \star data(y_m)$, it is the case that $\mathcal{G}'$ contains paths:

$$x' \xrightarrow{e_1} x'_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} x'_n \quad \text{and} \quad x' \xrightarrow{d_1} y'_1 \xrightarrow{d_2} \ldots \xrightarrow{d_m} y'_m,$$

for which the following holds:

- $x_i Z x'_i$ for each $1 \leq i \leq n$, and $y_j Z y'_j$ for each $1 \leq j \leq m$.
- $data(x'_n) \star data(y'_m)$.

That is, Z satisfies $\mathbf{Zig}_{=}$ if and only if for every $(x,x') \in Z$ it is the case that every word in $L(\mathcal{A}_{\mathcal{G},x})$ is also in $L(\mathcal{A}^{\mathcal{G},Z}_{\mathcal{G}',x'})$, i.e., $L(\mathcal{A}_{\mathcal{G},x}) \subseteq L(\mathcal{A}^{\mathcal{G},Z}_{\mathcal{G}',x'})$.

3.3.2 Lower bound

We start by showing the lower bound for XPATH$_=(\downarrow_{\mathbf{a}})$-SIMILARITY. We proceed by constructing a polynomial time reduction from containment of NFAs to XPATH$_=(\downarrow_{\mathbf{a}})$-SIMILARITY. Let $\mathcal{A}_i = (Q_i, q_i, F_i, \delta_i)$ be NFAs over alphabet $\mathbb{A}$, for $i = 1, 2$. Here, Q_i is the finite set of states of $\mathcal{A}_i$, q_i is the initial state, $F_i \subseteq Q_i$ is the set of final states, and $\delta_i \subseteq Q_i \times \mathbb{A} \times Q_i$ is its transition relation. We assume, without loss of generality, that q_i has no incoming transitions and F_i consists of a single state $f_i \neq q_i$ without outgoing transitions. Furthermore, we assume that f_i (for $i = 1, 2$) is reachable from every state in Q_i and that $Q_1 \cap Q_2 = \emptyset$. It is easy to see that containment of NFAs continues being PSPACE-HARD even under such restrictions.

Let $u_1, u_2, v_1, v_2, w_1, w_2$ be fresh elements that do not belong to $Q_1 \cup Q_2$. For $i = 1, 2$, we define a data graph $\mathcal{G}_i = (G_i, E_i, data_i)$ as follows (see Figure 40):

1. $G_i = Q_i \cup \{u_i, v_i, w_i\}$.
2. $(x, a, y) \in E_i$ if and only if one of the following holds:
   - $(x, a, y) \in \delta_i$
   - $x \in Q_i \setminus \{q_i, f_i\}$ and $y \in \{u_i, v_i, w_i\}$
   - $x = q_i$ and $y = u_i$
   - $x = u_i$ and $y \in \{u_i, v_i, w_i\}$
3. $data_i(q) = \begin{cases} 1 & \text{if } q \in \{q_i\} \cup F_i, \\ 2 & \text{if } q \in \{u_i\} \cup Q_i \setminus \{q_i, f_i\}, \\ 3 & \text{if } q = v_i, \\ 4 & \text{if } q = w_i. \end{cases}$

Clearly, $\mathcal{G}_i$ can be constructed in polynomial time from $\mathcal{A}_i$, for $i = 1, 2$. We show next that:

$$L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2) \iff \mathcal{G}_1, q_1 \rightrightarrows^{\downarrow_{\mathbf{a}}} \mathcal{G}_2, q_2.$$

We start with the right-to-left direction. Take an arbitrary word $e_1 \dots e_n \in L(\mathcal{A}_1)$. Therefore, $n > 0$ since $\mathcal{A}_1$ does not accept the empty word. Let $\varphi := \langle \varepsilon = \downarrow_{e_1} \dots \downarrow_{e_n} \rangle$. Then by construction $\mathcal{G}_1, q_1 \models \varphi$ (as there is a path from q_1 to f_1 in $\mathcal{G}_1$ labeled $e_1 \dots e_n$ and $data_1(q_1) = data_1(f_1)$). Hence, since $\mathcal{G}_1, q_1 \rightrightarrows^{\downarrow_{\mathbf{a}}} \mathcal{G}_2, q_2$ and φ is a positive node XPath$_=$ expression, Theorem 146 tells us that $\mathcal{G}_2, q_2 \models \varphi$. That is, there exists a path

$$\pi = q_2 \xrightarrow{e_1} u_1 \dots \xrightarrow{e_n} u_n \text{ in } \mathcal{G}_2 \text{ such that } data_2(q_2) = data_2(u_n).$$

Since $n > 0$, the node u_n can only be f_2. By construction, then, all internal nodes of π must belong to $Q_2 \setminus \{q_2, f_2\}$ (as there is no path linking nodes u_2, v_2, w_2 with f_2). This implies that $e_1 \dots e_n \in L(\mathcal{A}_2)$.

For the left-to-right direction, let us define $Z \subseteq G_1 \times G_2$ as follows (see Figure 40): xZy iff one of the following holds:

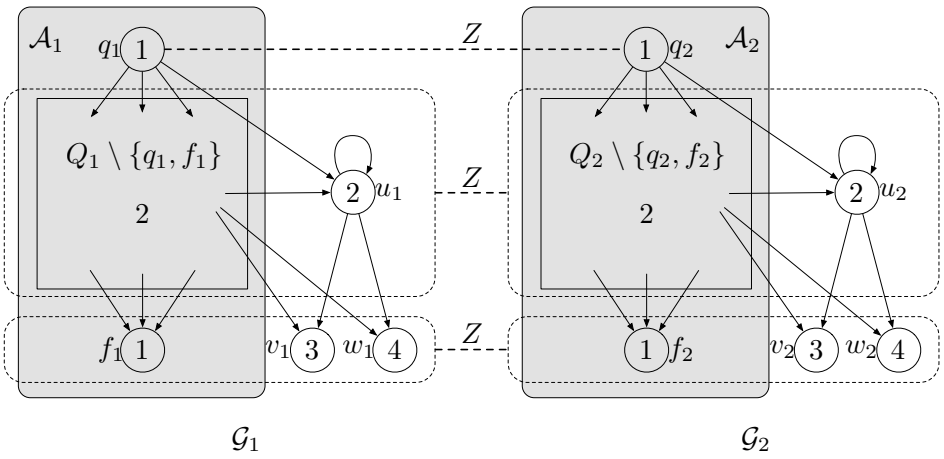


Figure 40: The data graphs $\mathcal{G}_i = (G_i, E_i, data_i)$ $(i = 1, 2)$, constructed from NFAs $\mathcal{A}_1$ and $\mathcal{A}_2$, and the bisimulation $Z \subseteq G_1 \times G_2$ used in the left-to-right direction of the reduction. All nodes inside a dotted area on $\mathcal{G}_1$ are related to all nodes inside a dotted on $\mathcal{G}_2$ area via Z.

1. $x = q_1$ and $y = q_2$.
2. $x \in \{u_1\} \cup Q_1 \setminus \{q_1, f_1\}$ and $y \in \{u_2\} \cup Q_2 \setminus \{q_2, f_2\}$.
3. $x \in \{v_1, w_1, f_1\}$ and $y \in \{v_2, w_2, f_2\}$.

We show next that Z satisfies the $\mathbf{Zig}_=$ clause for any pair $(x_1, x_2) \in G_1 \times G_2$ such that $x_1 Z x_2$. We do this by cases:

1. If $x_1 \in \{f_1, v_1, w_1\}$ and $x_2 \in \{f_2, v_2, w_2\}$, then $\mathbf{Zig}_=$ holds trivially as there are no outgoing paths from f_1, v_1 or w_1.
2. If $x_1 \in Q_1 \setminus \{q_1, f_1\} \cup \{u_1\}$ and $x_2 \in Q_2 \setminus \{q_2, f_2\} \cup \{u_2\}$, let α, β be two paths starting in x_1. Suppose first that $\alpha = x_1$ (*i.e.* the empty path starting at x_1) and

$$\beta = x_1 \xrightarrow{e_1} y_2 \xrightarrow{e_2} \ldots \xrightarrow{e_{m-1}} y_m \xrightarrow{e_m} z \quad [\text{resp. } x_1 \xrightarrow{e_1} y_2 \xrightarrow{e_2} \ldots \xrightarrow{e_{m-1}} y_m],$$

   where the y_i's are in $(Q_1 \setminus \{q_1, f_1\}) \cup \{u_1\}$ and $z \in \{f_1, v_1, w_1\}$. Then the corresponding β' in $\mathcal{G}_2$ is

$$x_2 \xrightarrow{e_1} u_2 \xrightarrow{e_2} \ldots \xrightarrow{e_{m-1}} u_2 \xrightarrow{e_m} v_2 \quad [\text{resp., } x_2 \xrightarrow{e_1} u_2 \xrightarrow{e_2} \ldots \xrightarrow{e_{m-1}} u_2],$$

   where there are $m-1$ occurrences of u_2. If both α and β have length greater than 0, then the procedure is similar, but one path may end in w_2 if the data values of the endpoints of α, β are different elements in $\{1, 3, 4\}$.
3. Finally, if $x_1 = q_1$ and $x_2 = q_2$, there are two main cases for the type of paths α, β in $\mathcal{G}_1$ to be replicated in $\mathcal{G}_2$ while respecting the $\mathbf{Zig}_=$ condition with respect to Z.

If both α and β are of length greater than 0, then copying them is straightforward using the nodes u_2, v_2, w_2. If both are of length 0 the result is trivial. Suppose one of them is length 0 and the other one is of length greater than 0, say $\alpha = q_1$ and

$$\beta = q_1 \xrightarrow{e_1} y_2 \xrightarrow{e_2} \ldots \xrightarrow{e_{n-1}} y_n \xrightarrow{e_n} z.$$

We need to find paths α' and β' in $\mathcal{G}_2$, both starting in q_2, which "copy" α and β respectively. Clearly α' is just q_2, as it has to have length 0. The definition of β' depends on z. If $z \in Q_1 \setminus \{q_1, f_1\} \cup \{u_1, v_1, w_1\}$, then β' is of the form

$$\beta' = q_2 \xrightarrow{e_1} y'_2 \xrightarrow{e_2} \ldots \xrightarrow{e_{n-1}} y'_n \xrightarrow{e_n} z', \tag{23}$$

where $y'_i = u_2$ $(i = 2 \ldots n)$ and z' is either u_2, v_2, or w_2 if z is u_1, v_1, or w_1, respectively. If $z = f_1$, then the endpoints of α and β have both data value 1. Since the word $e_1 \ldots e_n \in L(\mathcal{A}_1)$ and by hypothesis $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$, we conclude that $e_1 \ldots e_n$ is accepted by $\mathcal{A}_2$. This means that there is a path of the form (23) where $y'_i \in Q \setminus \{q_2, f_2, u_2, v_2, w_2\}$ and $z' = f_2$. This path satisfies the required condition of $\mathbf{Zig}_=$, as the condition on Z is verified and the data value of q_2 and f_2 (the endpoints of α' and β') are equal, namely have data value 1.

For XPath$_=(\downarrow_{\mathbf{a}})$-BISIMILARITY the proof is analogous, but using this time a reduction from NFA equivalence. In fact, it can be easily checked that with exactly the same construction shown above we obtain that:

$$L(\mathcal{A}_1) = L(\mathcal{A}_2) \iff \mathcal{G}_1, q_1 \leftrightarrows^{\downarrow_{\mathbf{a}}} \mathcal{G}_2, q_2.$$

Notice that this construction also immediately implies Corollary 149. In fact, for the right-to-left direction of the reduction to hold we only require invariance of $(\mathcal{G}_2, q_2)$ with respect to $(\mathcal{G}_1, q_1)$ referred to those formulas of the form $\langle \varepsilon = \downarrow_{e_1} \ldots \downarrow_{e_n} \rangle$, for $e_1 \ldots e_n \in \mathbb{A}$. In other words, we only require $(\mathcal{G}_1, q_1) \Rrightarrow^{\downarrow_{\mathbf{a}}}_{\text{paths}} (\mathcal{G}_2, q_2)$ or $(\mathcal{G}_1, q_1) \equiv^{\downarrow_{\mathbf{a}}}_{\text{paths}} (\mathcal{G}_2, q_2)$ depending on whether we are reducing from containment or equivalence of NFAs, respectively.

3.4 Restricting paths in bisimulations

The smallest witness to the fact that two NFAs are not equivalent (resp., one NFA is not contained in another one) might be a path of exponential length [83]. As a corollary to the proof of Theorem 147, we obtain then that the smallest witness to the fact that a given relation $Z \subseteq G \times G'$ does not satisfy the $\mathbf{Zig}_=$ condition might also be a pair (π_1, π_2) of paths of exponential length. This naturally calls for a restriction on the length of paths considered in the definition of XPath$_=(\downarrow_{\mathbf{a}})$-(bi)simulation as a way to obtain better complexity bounds. We consider this restriction natural for the following reasons:

- Long witnesses correspond to large distinguishing formulas in XPath$_=(\downarrow_{\mathbf{a}})$. But rarely will users be interested in checking if nodes are distinguishable by formulas that they

cannot even write. Thus, the restriction to shorter paths can be seen as an approximation to the notion of XPath$_=(\downarrow_{\mathbf{a}})$-(bi)similarity based on user-understandable formulas.

- In practice, algorithms for computing (bi)simulations in the absence of data stop after a few iterations [81, 80]. This tells us that when nodes in real-world graphs are distinguishable by BML-formulas, they are distinguishable by some small formula. One might expect a similar behavior for XPath$_=(\downarrow_{\mathbf{a}})$, implying that the restriction to shorter paths provides a fair approximation of the problem in practice.

In this section we study the complexity of XPath$_=(\downarrow_{\mathbf{a}})$-(bi)similarity for paths of restricted length. We show that the problem becomes Co-NP-complete for paths of polynomial length, and tractable for paths of constant length. This notion of bisimilarity further captures the expressive power of a natural fragment of XPath$_=(\downarrow_{\mathbf{a}})$; namely, the one formed by expressions of *bounded length*. This fragment only restricts formulas of the form $\langle \alpha \star \beta \rangle$, for $\star \in \{=, \neq\}$.

3.4.1 Bounded bisimulation and equivalence

Let $f : \mathbb{N} \to \mathbb{N}$ be a positive, non-decreasing function. We define the notion of f-XPath$_=(\downarrow_{\mathbf{a}})$-*[bi]simulation* as in Definition 144, but now in the **Zig**$_=$ [also **Zag**$_=$] condition we only consider paths π_1 and π_2 [resp., π_1' and π_2'] of length at most $f(\max(|G|, |G'|))$, where $|G|$ denotes the number of edges in $\mathcal{G}$. We call the new conditions **Zig**$^f_=$ and **Zag**$^f_=$, respectively.

More formally, an ***f*-XPath$_=(\downarrow_{\mathbf{a}})$-bisimulation** is a relation $Z \subseteq G \times G'$ such that for all $(x, x') \in G \times G'$ with xZx' we have:

- (**Zig**$^f_=$) For every paths in $\mathcal{G}$ of the form

$$\pi_1 = x \xrightarrow{e_1} x_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} x_n \text{ and } \pi_2 = x \xrightarrow{d_1} y_1 \xrightarrow{d_2} \ldots \xrightarrow{d_m} y_m$$

  such that $m, n \leq f(\max(|G|, |G'|))$, there are paths in $\mathcal{G}'$ of the form

$$\pi_1' = x' \xrightarrow{e_1} x_1' \xrightarrow{e_2} \ldots \xrightarrow{e_n} x_n' \text{ and } \pi_2' = x' \xrightarrow{d_1} y_1' \xrightarrow{d_2} \ldots \xrightarrow{d_m} y_m',$$

  such that the following holds:

  1. $x_i Z x_i'$ for all $1 \leq i \leq n$, and $y_j Z y_j'$ for all $1 \leq j \leq m$.
  2. $data(x_n) = data(y_m) \Leftrightarrow data(x_n') = data(y_m')$.

- (**Zag**$^f_=$) For every paths in $\mathcal{G}'$ of the form

$$\pi_1' = x' \xrightarrow{e_1} x_1' \xrightarrow{e_2} \ldots \xrightarrow{e_n} x_n' \text{ and } \pi_2' = x' \xrightarrow{d_1} y_1' \xrightarrow{d_2} \ldots \xrightarrow{d_m} y_m'$$

  such that $m, n \leq f(\max(|G|, |G'|))$, there are paths in $\mathcal{G}$ of the form

$$\pi_1 = x \xrightarrow{e_1} x_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} x_n \text{ and } \pi_2 = x \xrightarrow{d_1} y_1 \xrightarrow{d_2} \ldots \xrightarrow{d_m} y_m,$$

  such that conditions 1 and 2 above are verified.

- (**Zig**) For every $y \in G$ and $e \in \mathbb{A}$ such that $x \xrightarrow{e} y$ there is $y' \in G'$ such that $x' \xrightarrow{e} y'$ and yZy'.
- (**Zag**) For every $y' \in G'$ and $e \in \mathbb{A}$ such that $x' \xrightarrow{e} y'$ there is $y \in G$ such that $x \xrightarrow{e} y$ and yZy'.

The reason why we add the one-step, comparison-free rules of **Zig** and **Zag** will become clearer below, in the characterization theorem for a fragment of $\mathrm{XPath}_{=}(\downarrow_{\mathbf{a}})$ to be studied next (Theorem 151). The idea is that we want to restrict with f the length of pairs of paths which compare data values at their terminating nodes, but we do not want to restrict the length of single paths which do not compare data values.

Similarly, an ***f*-XPath$_{=}(\downarrow_{\mathbf{a}})$-simulation** is a relation $Z \subseteq G \times G'$ such that for all $(x, x') \in G \times G'$ with xZx' it is the case that condition $\mathbf{Zig}^{f}_{=}$ and **Zig** above are verified.

We now define when $u \in G$ and $u' \in G'$ are said to be ***f*-XPath$_{=}(\downarrow_{\mathbf{a}})$-[bi]similar** notated $\mathcal{G}, u \rightrightarrows_{f}^{\downarrow_{\mathbf{a}}} Gg', u'$ [resp. $\mathcal{G}, u \leftrightarrows_{f}^{\downarrow_{\mathbf{a}}} Gg', u'$]:

$$\mathcal{G}, u \rightrightarrows_{f}^{\downarrow_{\mathbf{a}}} [\leftrightarrows_{f}^{\downarrow_{\mathbf{a}}}] Gg', u' \overset{\text{def}}{\Leftrightarrow} \text{ there is an } f\text{-XPath}_{=}(\downarrow_{\mathbf{a}})\text{-[bi]simulation } Z \text{ such that } (u, u') \in Z.$$

The characterization. We define the logical counterpart of this refined notion of bisimulation. We aim at an analog of Theorem 146 relative to the adequate restriction of indistinguishability (compare for example with what was done for $\leftrightarrows_{\ell}^{\downarrow}$ and $\leftrightarrows_{r,s,k}^{\uparrow\downarrow}$ in §1.2.2). As we show below, this restriction is defined by the fragment of $\mathrm{XPath}_{=}(\downarrow_{\mathbf{a}})$ whose path expressions α occurring in an expression of the form $\langle \alpha \star \beta \rangle$ (for $\star \in \{=, \neq\}$) have length bounded by f. In the following we formalize this idea.

We recall that the *length* of a path expression α, denoted $\mathsf{len}(\alpha)$, corresponds (after the natural adaptation of its definition to the context of edge-labeled graphs) to the number of $\downarrow_a$'s occurring in α *at the uppermost level*, i.e., outside any test of the form $[\varphi]$. E.g., $\mathsf{len}(\downarrow_a[\langle \downarrow_b = \downarrow_a \downarrow_b \downarrow_c \rangle]\downarrow_b) = \mathsf{len}(\downarrow_a \downarrow_b) = 2$. We use this notion to define the *maximum length* of a node or path expression. This represents the maximum length of paths that are involved in expressions of the form $\langle \alpha \star \beta \rangle$, for $\star \in \{=, \neq\}$.

Definition 150 (Maximum length)**.** Given a node or path expression θ, we write $\mathsf{ml}(\theta)$ to denote the **maximum length** of θ. Formally, ml is recursively defined as follows:

$$\begin{aligned}
\mathsf{ml}(\lambda) &= 0 \\
\mathsf{ml}(\varepsilon\alpha) &= \mathsf{ml}(\alpha) \\
\mathsf{ml}([\varphi]\alpha) &= \max\{\mathsf{ml}(\varphi), \mathsf{ml}(\alpha)\} \\
\mathsf{ml}(\downarrow_a \alpha) &= \mathsf{ml}(\alpha) \\
\mathsf{ml}(\varphi \wedge \psi) &= \max\{\mathsf{ml}(\varphi), \mathsf{ml}(\psi)\} \\
\mathsf{ml}(\neg\varphi) &= \mathsf{ml}(\varphi) \\
\mathsf{ml}(\langle\alpha\rangle) &= \mathsf{ml}(\alpha) \\
\mathsf{ml}(\langle \alpha \star \beta \rangle) &= \max\{\mathsf{len}(\alpha), \mathsf{len}(\beta), \mathsf{ml}(\alpha), \mathsf{ml}(\beta)\},
\end{aligned}$$

where α is any path expression or the empty string λ. □

As an example:

$$\mathsf{ml}(\langle\downarrow_a[\langle\downarrow_b\downarrow_a\downarrow_c\rangle]\downarrow_b = \downarrow_b[\langle\downarrow_a\downarrow_b\downarrow_a\downarrow_b\rangle]\rangle) = 2.$$

We now state the notion of **equivalence** [and partial equivalence] for f-XPath$_=(\downarrow_{\mathbf{a}})$, notion which coincides with that of f-XPath$_=(\downarrow_{\mathbf{a}})$-[bi]simulation. Fix $f : \mathbb{N} \to \mathbb{N}$ to be a positive, non-decreasing function, and $\mathcal{G}, \mathcal{G}'$ to be two data graphs. Call $F = f(\max(|G|, |G'|))$, and remember that we say that a node expression is *positive* if it contains no negations.

$$\mathcal{G}, u \equiv_f^{\downarrow_{\mathbf{a}}} \mathcal{G}', u' \quad \overset{\text{def}}{\Leftrightarrow} \quad \begin{array}{l}\text{for all XPath}_=(\downarrow_{\mathbf{a}})\text{-node expressions } \varphi \text{ with } \mathsf{ml}(\varphi) \leq F,\\ \mathcal{G}, u \models \varphi \text{ iff } \mathcal{G}', u' \models \varphi\end{array}$$

$$\mathcal{G}, u \Rrightarrow_f^{\downarrow_{\mathbf{a}}} \mathcal{G}', u' \quad \overset{\text{def}}{\Leftrightarrow} \quad \begin{array}{l}\text{for all positive XPath}_=(\downarrow_{\mathbf{a}})\text{-node expressions } \varphi\\ \text{with } \mathsf{ml}(\varphi) \leq F,\ \mathcal{G}, u \models \varphi \Rightarrow \mathcal{G}', u' \models \varphi\end{array}$$

As in Theorem 146 we obtain the following characterization:

Proposition 151. *$\mathcal{G}, u \equiv_f^{\downarrow_{\mathbf{a}}} \mathcal{G}', u'$ iff $\mathcal{G}, u \leftrightarrows_f^{\downarrow_{\mathbf{a}}} \mathcal{G}', u'$, and $\mathcal{G}, u \Rrightarrow_f^{\downarrow_{\mathbf{a}}} \mathcal{G}', u'$ iff $\mathcal{G}, u \rightrightarrows_f^{\downarrow_{\mathbf{a}}} \mathcal{G}', u'$.*

Proof. It follows using the idea in the proof of Theorem 8 in [45]. Formulas of the form $\langle\alpha \star \beta\rangle$ of ml bounded by $f(\max(|G|, |G'|))$ (for $\star \in \{=, \neq\}$) are handled by rules $\mathbf{Zig}^f_=$ and $\mathbf{Zag}^f_=$. Formulas of the form $\langle\downarrow\alpha\rangle$ are equivalent to $\langle\downarrow[\langle\alpha\rangle]\rangle$, and these are handled by rules **Zig** and **Zag**. □

3.4.2 Computing f-XPath$_=(\downarrow_{\mathbf{a}})$-bisimulations

Here we study the complexity of computing f-XPath$_=(\downarrow_{\mathbf{a}})$-(bi)simulations:

f-XPath$_=(\downarrow_{\mathbf{a}})$-[Bi]Similarity problem	
INPUT :	Data graphs $\mathcal{G}$ and $\mathcal{G}'$, nodes $u \in G$ and $u' \in G'$.
OUTPUT :	'Yes' iff $\mathcal{G}, u \rightrightarrows_f^{\downarrow_{\mathbf{a}}} \mathcal{G}', u'$ [iff $\mathcal{G}, u \leftrightarrows_f^{\downarrow_{\mathbf{a}}} \mathcal{G}', u'$, resp.]

Since this problem is PSPACE-COMPLETE when f is an exponential function, it is natural to start by restricting f to be a polynomial. We show next that while this restriction lowers the complexity of our problem, it still does not yield tractability:

Proposition 152. *The following holds:*

1. *The problem p-XPATH$_=(\downarrow_{\mathbf{a}})$-(BI)SIMILARITY is in* CO-NP *for every non-decreasing polynomial $p : \mathbb{N} \to \mathbb{N}$.*

2. *The problem p-XPATH$_=(\downarrow_{\mathbf{a}})$-(BI)SIMILARITY can be* CO-NP-HARD *even if $p : \mathbb{N} \to \mathbb{N}$ is the identity function.*

Proof. We only prove the claim for p-XPATH$_=(\downarrow_{\mathbf{a}})$-SIMILARITY. The proof for bisimilarity is analogous. We start with item 1. Let $\mathcal{G}$ and $\mathcal{G}'$ be data graphs and u, u' nodes in G, G', respectively. In order to check whether there is an XPath$_=(\downarrow_{\mathbf{a}})$-simulation from u to u' we

can use the standard greatest fixed point algorithm for computing the maximal simulation in the absence of data. We adapt it here to the $\mathbf{Zig}^p_=$ condition of XPath$_=(\downarrow_\mathbf{a})$-simulations.

The algorithm computes the maximal XPath$_=(\downarrow_\mathbf{a})$-simulation from $\mathcal{G}$ to $\mathcal{G}'$. We start by defining $Z = G \times G'$. At each step we choose an arbitrary pair $(x, x') \in Z$. If $\mathbf{Zig}^p_=$ fails when evaluated on this pair we simply remove it from Z. We proceed iteratively until we reach a fixed point. Finally, we check whether $(u, u') \in Z$. Only if this is the case we declare that there is an XPath$_=(\downarrow_\mathbf{a})$-simulation from u to u'.

Thus, in order to check that there is *no* XPath$_=(\downarrow_\mathbf{a})$-simulation from u to u', we can simply guess a computation of the algorithm that removes the pair (u, u') from Z. Such computation consists of (a) pairs $(x_1, x'_1), \ldots, (x_m, x'_m)$; (b) relations $Z_0, \ldots, Z_m$ such that: $Z_0 = G \times G'$, $Z_i = Z_{i-1} \setminus \{(x_i, x'_i)\}$ for each $1 \leq i \leq m$, and Z_m does not contain (u, u'); and (c) suitable witnesses for the fact that (x_i, x'_i) does not satisfy $\mathbf{Zig}^p_=$ with respect to Z_{i-1}, for each $1 \leq i \leq m$. Such witness consists of a pair (π_1, π_2) of paths of length at most $p(\max(|\mathcal{G}|, |\mathcal{G}'|))$ in $\mathcal{G}$ starting from x_i, and yet another witness for the fact that no pair (π'_1, π'_2) of paths in $\mathcal{G}'$ starting from x'_i satisfies $\mathbf{Zig}^p_=$ with respect to (π_1, π_2). The latter can be represented by an accepting run of the complement of the NFA $\mathcal{A}_{\mathcal{G}', x'_i, Z_{i-1}}$ (as defined in the proof of the upper bound of Theorem 147) over the word that represents the pair (π_1, π_2) in $\mathcal{A}_{\mathcal{G}, x_i}$. Clearly, each one of the components of this guess can be represented using polynomial space. Further, it can be checked in polynomial time that the guess satisfies the required conditions It follows that checking whether there is *no* XPath$_=(\downarrow_\mathbf{a})$-simulation from u to u' is in NP (and, thus, that our problem is in Co-NP).

For item 2 we use the following claim:

Lemma 153. *The problem of checking containment of NFA $\mathcal{A}_1$ in $\mathcal{A}_2$, restricted to words of length at most* $\max(|\mathcal{A}_1|, |\mathcal{A}_2|)$ *is* Co-NP-hard. *(Here, $|\mathcal{A}_i|$ defines the number of tuples in the transition relation of $\mathcal{A}_i$, for $i = 1, 2$).*

Proof. We use a reduction from the complement of 3CNF satisfiability. Given a 3CNF formula φ of the form

$$(l^1_1 \vee l^2_1 \vee l^3_1) \wedge \ \ldots \ \wedge (l^1_n \vee l^2_n \vee l^3_n),$$

over the set $\{h_1, \ldots, h_m\}$ of propositional symbols, we construct in polynomial time NFAs $\mathcal{A}_1$ and $\mathcal{A}_2$ over the alphabet $\mathbb{A} := \{h_1, \ldots, h_m, \neg h_1, \ldots, \neg h_m\}$ such that:

$$\varphi \text{ is satisfiable} \iff L(\mathcal{A}_1) \not\subseteq L(\mathcal{A}_2). \tag{24}$$

The NFA $\mathcal{A}_1$ consists of states $q_0, \ldots, q_n$, with q_0 and q_n being the initial and final state. The transitions of $\mathcal{A}_1$ are the ones in the set:

$$\{(q_i, l^j_{i+1}, q_{i+1}) \mid 0 \leq i < n, \ j = 1, 2, 3\}.$$

That is, the words accepted by $\mathcal{A}_1$ codify the different ways in which we can choose one literal from each clause in φ. One of these words encodes a satisfying assignment of φ if and only if it does not contain an "error", i.e., it does not mention a propositional variable

and its negation. We then take $\mathcal{A}_2$ as the NFA that accepts those words over the alphabet $\mathbb{A}$ that do mention a propositional variable in $\{h_1, \ldots, h_m\}$ and its negation, i.e.,

$$L(\mathcal{A}_2) \;=\; \bigcup_{1 \leq i \leq m} \mathbb{A}^* h_i \mathbb{A}^* \neg h_i \mathbb{A}^* \cup \mathbb{A}^* \neg h_i \mathbb{A}^* h_i \mathbb{A}^*.$$

Such $\mathcal{A}_2$ can be easily defined as follows. The states of $\mathcal{A}_2$ are r_0 and s_i, t_i, s'_i, t'_i, for each $1 \leq i \leq m$. The initial state is r_0 and the set of final states is $\{t_i, t'_i \mid 1 \leq i \leq m\}$. For each symbol $a \in \mathbb{A}$ and state q in $\mathcal{A}_2$, there is a transition (q, a, q). Moroever, for each $1 \leq i \leq m$ the NFA $\mathcal{A}_2$ contains transitions

$$(r_0, h_i, s_i), (s_i, \neg h_i, t_i), (r_0, \neg h_i, s'_i), (s'_i, h_i, t'_i).$$

It is easy to see that (24) holds. In fact, φ is satisfiable if and only if there is a word in $L(\mathcal{A}_1)$ that does not contain an "error", that is, it does not belong to $L(\mathcal{A}_2)$.

Notice, in addition, that $|\mathcal{A}_1| = 3n$ and $|\mathcal{A}_2| = 8m^2 + 6m \leq 14m^2$. Moreover, the words in $L(\mathcal{A}_1)$ are of length at most n. This tells us that not only (24) holds, but it holds even if the containment of $\mathcal{A}_1$ into $\mathcal{A}_2$ is restricted to words of length at most $\max(|\mathcal{A}_1|, |\mathcal{A}_2|)) \geq 3n > n$. □

We then reduce the restricted containment problem from Lemma 153 to p-XPath$_=(\downarrow_{\mathbf{a}})$-similarity, where $p : \mathbb{N} \to \mathbb{N}$ is the identity function, by using the same construction than in the proof of the lower bound of Theorem 147. In fact, it can be readily checked that, starting from NFAs $\mathcal{A}_1$ and $\mathcal{A}_2$, such reduction constructs data graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ with distinguished nodes q_1 and q_2, respectively, such that the following are equivalent:

1. $\mathcal{A}_1$ is contained in $\mathcal{A}_2$ up to words of length at most $\max(|\mathcal{A}_1|, |\mathcal{A}_2|))$.
2. $(\mathcal{G}, q_1) \rightrightarrows^{\downarrow_{\mathbf{a}}}_{\max(|\mathcal{G}_1|, |\mathcal{G}_2|))} (\mathcal{G}_2, q_2)$.

This finishes the proof of the proposition. □

The reason why the previous restriction is not sufficient to obtain tractability is that there are too many paths of polynomial length in a data graph. We solve this issue by restricting to paths of constant length only. In the following we identify the function that takes constant value $c \in \mathbb{N}$ with the letter c.

For $c \geq 0$, we call **XPath$_=(\downarrow_{\mathbf{a}})(c)$**, or **c-XPath$_=(\downarrow_{\mathbf{a}})$**, to the syntactical fragment of XPath$_=(\downarrow_{\mathbf{a}})$ with a ml bounded by c. Further, fragments of the form c-XPath$_=(\downarrow_{\mathbf{a}})$ (for $c \geq 1$) extend multi-modal logic, which in turn coincides with 0-XPath$_=(\downarrow_{\mathbf{a}})$:

Proposition 154. *0-XPath$_=(\downarrow_{\mathbf{a}})$ is semantically equivalent to multi-modal logic with no propositions and only atoms $\top$ and $\bot$.*

Proof. In the jargon of ML, we have a language with modalities $\langle a \rangle$ for each $a \in \mathbb{A}$. On the one hand, any node expression of the form $\langle \alpha \star \beta \rangle$ in 0-XPath$_=(\downarrow_{\mathbf{a}})$ is also of the form

$$\langle [\varphi_1] \ldots [\varphi_n] \star [\psi_1] \ldots [\psi_m] \rangle,$$

which is equivalent to $\bigwedge_i \varphi_i \wedge \bigwedge_j \psi_j$, if $\star$ is $=$, and to a contradiction (e.g., $\neg\langle\varepsilon\rangle$) otherwise. On the other hand, a node expression $\langle\downarrow_a \alpha\rangle$ is equivalent to $\langle\downarrow_a[\langle\alpha\rangle]\rangle$. Any node expression of the form $\langle\downarrow_a[\varphi]\rangle$ of 0-XPath$_=(\downarrow_{\mathbf{a}})$ can be straightforwardly translated (in a truth-preserving way) to ML via T as $\langle a\rangle T(\varphi)$. The translation from modal logic to 0-XPath$_=(\downarrow_{\mathbf{a}})$ is obvious. □

Proposition 155. *The problem of* c-XPath$_=(\downarrow_{\mathbf{a}})$-(BI)SIMILARITY *is* PTIME-COMPLETE *for each constant* $c > 0$.

Proof. We use the same algorithm as in the proof of the previous upper bound. The difference now is that checking whether a pair $(x, x') \in Z$ satisfies $\mathbf{Zig}^c_=$ can be solved efficiently for each $c > 0$. This is because there is at most a polynomial number of paths of length $\leq c$ in $\mathcal{G}$ starting from x. We conclude that checking whether $\mathcal{G}, u \rightrightarrows_c^{\downarrow_{\mathbf{a}}} \mathcal{G}', u'$ is in PTIME. The same holds for $\mathcal{G}, u \leftrightarrows_c^{\downarrow_{\mathbf{a}}} \mathcal{G}', u'$. The lower bound follows straightforwardly from Proposition 154 and PTIME-hardness for usual ML-bisimulations [12]. □

Proposition 151 establishes that c-XPath$_=(\downarrow_{\mathbf{a}})$-simulations characterize the expressive power of the fragment of XPath$_=(\downarrow_{\mathbf{a}})$ defined by formulas of ml bounded by the constant c. The following corollary to the proof of Proposition 155 states that when two nodes are not c-XPath$_=(\downarrow_{\mathbf{a}})$-bisimilar, it is possible to compute in polynomial time a node expression in this fragment that distinguishes them.

Corollary 156. *There is a polynomial time algorithm which given* $\mathcal{G}, u \not\leftrightarrows_c^{\downarrow_{\mathbf{a}}} \mathcal{G}', u'$ *[resp.,* $\mathcal{G}, u \not\rightrightarrows_c^{\downarrow_{\mathbf{a}}} \mathcal{G}', u'$*], constructs*[16] *a [positive] node expression* φ *of* $c - XPath_=(\downarrow_{\mathbf{a}})$ *such that* $\mathcal{G}, u \models \varphi$ *and* $\mathcal{G}', u' \not\models \varphi$.

Proof. We adapt the algorithm given in [8] for BML to our notion of bisimulation. We only explain the case of c-XPath$_=(\downarrow_{\mathbf{a}})$-simulation, as for bisimulation the proof is analogous.

We construct a polynomial time algorithm which, on input $\mathcal{G}$ and $\mathcal{G}'$, computes, for each $x \in G$, a set $S(x) = \{x' \in G' \mid x \rightrightarrows_c^{\downarrow_{\mathbf{a}}} x'\}$ and a positive node expression $N(x)$ such that $\mathcal{G}, x \models N(x)$ and $[\![N(x)]\!]^{\mathcal{G}'} = S(x)$. The existence of this algorithm is enough to prove the desired result: if $\mathcal{G}, u \not\rightrightarrows_c^{\downarrow_{\mathbf{a}}} \mathcal{G}', u'$, we have that $u' \notin S(u)$ and therefore $\mathcal{G}, x \models N(x)$ and $\mathcal{G}', x' \not\models N(x)$, hence satisfying the statement of the Corollary.

The algorithm is as follows. We start by setting, for all $x \in G$, $S(x) := G'$ and $N(x) := \top$ (representing any positive tautology, *e.g.*$\langle\varepsilon\rangle$).We repeat the following process until none of the items below are true:

- There are $(x, x') \in G \times G'$ such that $\mathbf{Zig}^c_=$ does not hold at (x, x'), in the sense that there are paths in $\mathcal{G}$ of the form

$$\pi_1 = x \xrightarrow{e_1} x_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} x_n \text{ and } \pi_2 = x \xrightarrow{d_1} y_1 \xrightarrow{d_2} \ldots \xrightarrow{d_m} y_m$$

  with $m, n \leq c$, and such that there are no paths in $\mathcal{G}'$ of the form

$$\pi'_1 = x' \xrightarrow{e_1} x'_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} x'_n \text{ and } \pi'_2 = x' \xrightarrow{d_1} y'_1 \xrightarrow{d_2} \ldots \xrightarrow{d_m} y'_m,$$

[16] Provided an adequate representation for such formula is chosen [51].

satisfying $x'_i \in S(x_i)$ for all $1 \leq i \leq n$, $y'_i \in S(y_i)$ for all $1 \leq j \leq m$, and $data(x_n) = data(y_m)$ iff $data(x'_n) = data(y'_m)$. In this case we set $S(x) := S(x) \setminus \{x'\}$ and

$$N(x) := N(x) \wedge \langle \downarrow_{e_1}[N(x_1)]\downarrow_{e_2} \ldots \downarrow_{e_n}[N(x_n)] \star \downarrow_{d_1}[N(y_1)]\downarrow_{d_2} \ldots \downarrow_{d_m}[N(y_m)]\rangle,$$

where $\star$ is $=$ in case $data(x_n) = data(y_m)$ and $\neq$ otherwise.

- There are $(x, x') \in G \times G'$ such that **Zig** does not hold at (x, x'), in the sense that there is $y \in G$ such that $x \xrightarrow{e} y$ and there is no $y' \in G'$ such that $y \xrightarrow{e} y'$ and $y' \in S(y)$. In this case, we set $S(x) := S(x) \setminus \{x'\}$ and $N(x) := N(x) \wedge \langle \downarrow_e [N(y)]\rangle$.

The idea is that at each step, if either $\mathbf{Zig}^f_=$ or **Zig** are false, we shrink $S(x)$ for some x, and the "reason" behind the falsity of $\mathbf{Zig}^f_=$ or **Zig** is encoded in the node expression $N(x)$. The invariant of the cycle states that for all $x \in G$ we have $[\![N(x)]\!]^{\mathcal{G}'} \subseteq S(x)$ and $\mathcal{G}, x \models N(x)$. Since at execution of the body of the cycle, one element is removed from $S(x)$, for some $(x, x') \in G \times G'$, the total number of iterations is polynomial in the size of the data graphs. Furthermore, at each iteration, we only search for paths of length at most c, and so the total number of steps taken by this algorithm is polynomial. □

3.5 Restricting the models

Here we follow a different approach from the one in Section 3.4: We constrain the topology of graphs instead of the (bi)simulations. Since our goal is restricting the number and/or length of the paths considered in the analysis of (bi)simulations, it is natural to look into acyclic graphs; namely, trees and DAGs.

Let us start with data trees, i.e., data graphs whose underlying graph is a directed tree. This case is relevant as XML documents are (essentially) data trees, and the study of $\mathrm{XPath}_=(\downarrow_{\mathbf{a}})$-bisimulations was started in such context. Notice that for data trees both the number and the length of paths one needs to consider when checking the $\mathbf{Zig}_=$ and $\mathbf{Zag}_=$ conditions are polynomial. This implies that the problem of computing $\mathrm{XPath}_=(\downarrow_{\mathbf{a}})$-(bi)simulations over data trees is tractable:

Theorem 157. *The problem of* XPath$_=(\downarrow_{\mathbf{a}})$-(bi)similarity *for data trees is in* PTime.

As a second case, let us consider *data DAGs*, which allow for undirected cycles only. In this case the length, but not the number, of the paths one needs to consider at the moment of checking the $\mathbf{Zig}_=$ and $\mathbf{Zag}_=$ conditions is polynomial. The first observation helps lowering the complexity of computing $\mathrm{XPath}_=(\downarrow_{\mathbf{a}})$-(bi)simulations in this context from PSpace to Co-NP, while the second one prevents us from obtaining tractability.

Lemma 158. *The problems of containment or equivalence of NFAs whose underlying graph is a DAG are both* Co-NP-complete.

Proof. The Co-NP-hardness of the NFA DAG containment problem follows from Lemma 153, since the number of tuples in the transition relations of a DAG form an upper bound

to the length of words accepted by it. The completeness of the problem is then immediate, as a witness of $L(\mathcal{A}_1) \not\subseteq L(\mathcal{A}_2)$ can be checked in polynomial time, since there is a linear bound on the length of words accepted by DAGs. For the hardness of the equivalence problem, we can reduce the problem of DAG containment to equivalence by adding a disjoint copy of $\mathcal{A}_2$ to $\mathcal{A}_1$ and then joining by a same parent node: $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$ iff $L(\mathcal{A}_1 \sqcup \mathcal{A}_2) = L(\mathcal{A}_2)$. The completeness follows as in the case of containment. □

Proposition 159. *The problem* XPATH$_=(\downarrow_{\mathbf{a}})$-(BI)SIMILARITY *for data DAGs is* CO-NP-COMPLETE.

Proof. The problem is in CO-NP as a consequence of the first item of Proposition 152, since paths in DAGs are of linear size. To prove CO-NP-hardness, we reduce the problem of DAG containment (equivalence) to the problem of data DAG (bi)similarity, and then use Lemma 158. We will see the proof only for simulation, as the case of bisimulation is analogous.

Given two DAG NFAs $\mathcal{A}_i = (Q_i, q_i, F_i, \delta_i)$, for $i = 1, 2$, (*i.e.*, NFA whose transition graphs have no cycles) we construct data DAGs $\mathcal{G}_i = (G_i, E_i, data_i)$, with $i = 1, 2$, as in the proof of the lower bound of Theorem 147. The main difference is that the u_i of Figure 40 will be replaced by some DAGs that we will now construct. Let $n = \max\{|Q_1|, |Q_2|\}$, and let Q_i^j be the set of nodes $q \in Q_i$ at "maximum distance j" from f_i, that is, so that there is a directed path from q to f_i of length j but there is no such path of length $> j$ (note that $Q_i = \bigcup_{j \leq n} Q_i^j$). Now, each u_i of Figure 40 is replaced with n fresh nodes $u_i^1, \ldots, u_i^n$ of data value 2, with every possible edge from Q_i^j to u_i^{j-1}, from u_i^j to v_i, w_i, and from u_i^j to $u_i^{j'}$ for every $j' < j$. It is straightforward to check that the resulting data graphs are DAGs, and that as in the lower bound proof of Theorem 147 we have that $\mathcal{G}_1, q_1 \rightrightarrows^{\downarrow_{\mathbf{a}}} \mathcal{G}_2, q_2$ if and only if $L(A_1) \subseteq L(A_2)$. □

3.6 Two-way XPath$_=$

A common expressive extension for languages on graphs is to consider a *two-way* version that allows to traverse edges in both directions (see, e.g., [20, 78]). We call XPath$_=(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})$ the extension of XPath$_=(\downarrow_{\mathbf{a}})$ with path expressions of the form $\uparrow_a$, for $a \in \mathbb{A}$. The semantics of these expressions over $\mathcal{G} = \langle G, E, data\rangle$ is as follows: $[\![\uparrow_a]\!]^{\mathcal{G}} = \{(x, y) \mid (y, a, x) \in E\}$. We write the expected definitions of **equivalence** and partial equivalence for XPath$_=(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})$:

$$\mathcal{G}, u \equiv^{\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}}} \mathcal{G}', u' \quad \overset{\text{def}}{\Leftrightarrow} \quad \begin{array}{l}\text{for any XPath}_=(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})\text{-node expression } \varphi,\\ \mathcal{G}, u \models \varphi \text{ iff } \mathcal{G}', u' \models \varphi.\end{array}$$

$$\mathcal{G}, u \Rrightarrow^{\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}}} \mathcal{G}', u' \quad \overset{\text{def}}{\Leftrightarrow} \quad \begin{array}{l}\text{for any positive XPath}_=(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})\text{-node expression } \varphi,\\ \text{if } \mathcal{G}, u \models \varphi \text{ then } \mathcal{G}', u' \models \varphi.\end{array}$$

A notion of (bi)simulation for XPath$_=(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})$ *over data trees* was introduced in [45], and turns out to be tractable. It heavily relies on the determinism of $\uparrow_a$ over trees, and hence does not fit in the context of data graphs. However, there is a simple way to adapt XPath$_=(\downarrow_{\mathbf{a}})$-bisimulations to the case of XPath$_=(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})$.

Given a data graph $\mathcal{G} = \langle G, E, data\rangle$ over $\mathbb{A}$, we define its *completion* over $\mathbb{A} \cup \mathbb{A}^-$, where $\mathbb{A}^- := \{a^- \mid a \in \mathbb{A}\}$, as the data graph $\mathcal{G}_c = \langle G, E_c, data\rangle$, where E_c extends E by adding the edge (v, a^-, u), for each edge $(u, a, v) \in E$. That is, $\mathcal{G}_c$ extends $\mathcal{G}$ with the "inverse" of every edge in E.

We also define a bijection $\varphi \mapsto \varphi_c$ mapping node expressions of $\text{XPath}_=$ over $\mathbb{A}$ to node expressions of $\text{XPath}_=(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})$ over $\mathbb{A} \cup \mathbb{A}^-$ as follows: φ_c is the result of replacing each occurrence of $\uparrow_a$ in φ (for $a \in \mathbb{A}$) with $\downarrow_{a^-}$. The following proposition is straightforward:

Proposition 160. *$\mathcal{G}, u \models \varphi$ iff $\mathcal{G}_c, u \models \varphi_c$.*

We say that there is an **$\text{XPath}_=(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})$-bisimulation** between $u \in G$ and $u' \in G'$ (denoted $\mathcal{G}, u \leftrightarrows^{\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}}} \mathcal{G}', u'$) if $\mathcal{G}_c, u \leftrightarrows^{\downarrow_{\mathbf{a}}} \mathcal{G}'_c, u'$ (over the extended alphabet $\mathbb{A} \cup \mathbb{A}^-$). Similarly, we define **$\text{XPath}_=(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})$-simulations** $\rightrightarrows^{\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}}}$. Analogously to Theorem 146, one can show:

Theorem 161. *$\mathcal{G}, u \equiv^{\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}}} \mathcal{G}', u'$ iff $\mathcal{G}, u \leftrightarrows^{\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}}} \mathcal{G}', u'$, and $\mathcal{G}, u \rightrightarrows^{\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}}} \mathcal{G}', u'$ iff $\mathcal{G}, u \rightrightarrows^{\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}}} \mathcal{G}', u'$.*

We study the complexity of the following problem:

$\text{XPath}_=(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})$-[Bi]Similarity problem	
INPUT :	Data graphs $\mathcal{G}$ and $\mathcal{G}'$, nodes $u \in G$ and $u' \in G'$.
OUTPUT :	'Yes' iff $\mathcal{G}, u \rightrightarrows^{\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}}} \mathcal{G}', u'$ [iff $\mathcal{G}, u \leftrightarrows^{\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}}} \mathcal{G}', u'$, resp.]

The bounded notions of bisimulation introduced in §3.4 are defined over $\text{XPath}_=(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})$ and alphabet $\mathbb{A}$ in the expected way: reducing to $\text{XPath}_=$ over the signature $\mathbb{A} \cup \mathbb{A}^-$ and the corresponding completion of the data graphs. We use symbols $\rightrightarrows_f^{\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}}}$ [resp. $\leftrightarrows_f^{\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}}}$] for referring to f-$\text{XPath}_=(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})$-[bi]similarity. We then study:

f-$\text{XPath}_=(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})$-[Bi]Similarity problem	
INPUT :	Data graphs $\mathcal{G}$ and $\mathcal{G}'$, nodes $u \in G$ and $u' \in G'$.
OUTPUT :	'Yes' iff $\mathcal{G}, u \rightrightarrows_f^{\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}}} \mathcal{G}', u'$ [iff $\mathcal{G}, u \leftrightarrows_f^{\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}}} \mathcal{G}', u'$, resp.]

The identification of $\text{XPath}_=(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})$ over $\mathbb{A}$ with $\text{XPath}_=$ over $\mathbb{A} \cup \mathbb{A}^-$ and the corresponding completions of graphs allows us to straightforwardly transfer some upper bounds:

- $\text{XPath}_=(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})$-(BI)SIMILARITY is in PSPACE (§3.3.1).
- p-$\text{XPath}_=(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})$-(BI)SIMILARITY, for p a non-decreasing polynomial, is in CO-NP (item 1 of Proposition 152).
- c-$\text{XPath}_=(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})$-(BI)SIMILARITY, for c a constant function, is in PTIME (Proposition 155)

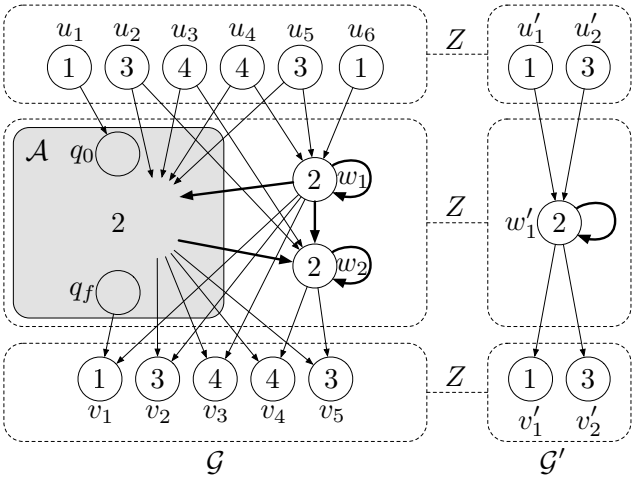


Figure 41: Definition of the data graphs $\mathcal{G}$ and $\mathcal{G}'$ based on an NFA $\mathcal{A}$ over alphabet $\mathbb{A}$. Boldface arrows have, as label, all symbols from $\mathbb{A}$. Lightface arrows have all the same label $e \notin \mathbb{A}$ for some e. All nodes of $\mathcal{A}$ (the grey area) have data value 2. All nodes inside a dotted area on $\mathcal{G}$ are related to all nodes inside a dotted area on $\mathcal{G}'$ via Z.

Regarding the lower bounds, we focus here on the general case of XPath$_=(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})$ over data graphs. One can check that the proof given in §3.3.2 does not work because in the two way context, more nodes in the graphs can be reached through the accessibility relations. The main result of this section is the following:

Theorem 162. XPath$_=(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})$-(Bi)similarity *is* PSpace-hard.

Proof. We only verify the bisimilarity case, as the similarity case can be solved in the same way.

We reduce to this problem from the PSpace-complete problem of *universality* for NFA (i.e., does an NFA accept all words?). Let $\mathcal{A}$ be a NFA over $\mathbb{A}$, with initial state q_0 and final state q_f. We build data graphs $\mathcal{G}$ and $\mathcal{G}'$ as in Figure 41.

We claim that $\mathcal{G}, u_1 \leftrightarrows^{\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}}} \mathcal{G}', u'_1$ iff $L(\mathcal{A}) = \Sigma^*$.

For the left-to-right direction we proceed as follows. Given a word $\omega = a_1 \dots a_n$ in Σ^*, we consider the formula $\varphi = \langle \varepsilon = \downarrow_e \downarrow_{a_1} \dots \downarrow_{a_n} \downarrow_e \rangle$. By construction, $\mathcal{G}', u'_1 \models \varphi$, and from the hypothesis $\mathcal{G}, u_1 \leftrightarrows^{\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}}} \mathcal{G}', u'_1$ and Theorem 161, we have that $\mathcal{G}, u_1 \models \varphi$. On the one hand, the only way to transit from u_1 is through e, getting to q_0. On the other hand, in order to satisfy φ in u_1, there is a path

$$u_1 \xrightarrow{e} q_0 \xrightarrow{a_1} \dots \xrightarrow{a_n} z_1 \xrightarrow{e} z_2$$

where $data(z_2) = data(u_1) = 1$. One can see that the only possibility is that $z_2 = v_1$. One could arrive to v_1 from q_f or w_1, but w_1 is downwardly inaccessible from $\mathcal{A}$, and therefore $z_1 = q_f$. Even more, all the nodes in the path $q_0 \xrightarrow{a_1} \dots \xrightarrow{a_n} z_1$ are in $\mathcal{A}$, and so $\omega \in L(\mathcal{A})$.

For the right-to-left direction, one can check that the relation Z depicted in Figure 41 is an XPath$_=(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})$-bisimulation. In all cases, the **Zig**$_=$ condition is easily satisfied thanks to the construction of $\mathcal{G}'$, so we will only check **Zag**$_=$ for every pair $(x, x') \in G \times G'$.

- To prove **Zag**$_=$ for the pair (u_1, u'), with $u' \in \{u'_1, u'_2\}$, let $\alpha = u' \xrightarrow{e_1} x_2 \xrightarrow{e_2} \ldots \xrightarrow{e_{m-1}} x_m$ and $\beta = u' \xrightarrow{e'_1} y_2 \xrightarrow{e'_2} \ldots \xrightarrow{e'_{n-1}} y_n$, where $e_i, e'_i \in \mathbb{A} \cup \mathbb{A}^-$. The interesting case is that where: one of the paths has length 0, say $\alpha = u'$; y_n has the same data value as u'; and all e'_i belong to $\mathbb{A}$. In this case, we only need to use the hypothesis that $L(\mathcal{A}) = \Sigma^*$, and here it is important that w_2 cannot reach down into v_1.

- For all other pairs (u, u') with $u \in \{u_2, u_3, u_4, u_5, u_6\}$, $u' \in \{u'_1, u'_2\}$ in the topmost level of the figure, the general strategy for the construction of our needed paths is to mimic the paths in $\mathcal{G}'$ by going into w_1 or w_2 whenever the original paths go to w'_1. Whenever the paths on $\mathcal{G}'$ go to u'_1, u'_2, v'_1, or v'_2, and the path does not end there, the path on $\mathcal{G}$ mirrors the move by going into any node in the top or bottom part of $\mathcal{G}$, as appropriate. Finally, depending on whether the original paths end in different or same data value, the path in $\mathcal{G}$ can go to two nodes with data value 3 and 4 or to the same node, respectively.

- To prove that **Zag**$_=$ holds for all pairs (w, w') with $w \in \mathcal{A} \cup \{w_1, w_2\}, w' = w'_1$ of the middle level, the hard case is when w is a node of the $\mathcal{A}$ portion of the graph. Here, the idea of the procedure is the same as in the previous item.

- Finally, for pairs (v, v') with $v \in \{v_1, v_2, v_3, v_4, v_5\}$, $v' \in \{v'_1, v'_2\}$ of the bottom level, we also proceed as before, going upwards into w_1 as soon as the path in $\mathcal{G}'$ goes upwards into w'_1, and afterward mimicking the behaviors of the paths as needed.

This concludes the proof. □

Chapter 4

Logics of repeating values over data trees

> It was no type of tree I had ever seen before, and I approached it slowly [...]
> It sees *all* the future. Clearly. Perfectly. Everything that can possibly come to pass, branching out endlessly from the current moment.
>
> *The Wise Man's Fear*
> Patrick Rothfuss

4.1 Introduction

In this chapter we work with an expanded definition of data trees, where we allow each node to carry not merely a single data value, but a finite collection of them (organized in an ordered way). This structure of (multi)data trees has been considered in the realm of semistructured data as another abstraction of XML documents, but also of timed automata, program verification, and generally in systems manipulating data values. Finding decidable logics or automaton models over data trees is an important quest when studying data-driven systems.

A wealth of specification formalisms on these structures (either for data trees or its 'word' version, data words) have been introduced, stemming from automata [88, 99], first-order logic [17, 65, 48, 19], XPath [66, 50, 44, 43, 47], or temporal logics [38, 79, 70, 46, 36, 67]. In full generality, most formalisms lead to undecidable reasoning problems and a well-known research trend consists of finding a good trade-off between expressiveness and decidability.

Interesting and surprising results have been exhibited about relationships between logics for data trees and counter automata [65, 50, 66], indicating that logics for data trees are not only interesting for their own sake but also for their deep relationships with counter systems.

In this chapter we study the basic mechanism of "data repetition", common to many logics used on data trees. For this, we investigate a basic logic that can navigate the structure of the tree through the use of modalities like those of CTL (computation tree logic), and on the other hand can make "data tests", by asking whether a data value is repeated in a subtree. More concretely, the data tests are formulas of the form $u \approx \mathrm{EF}v$, stating that the data value stored in attribute u of the current node is equal to the data value stored in attribute v of some descendant. This kind of *logics of repeating values*, or LRV, has been the center of a line of investigation studied in [35, 36] on *data words*, evidencing tight correspondences between their satisfiability problems and the reachability problems for Vector Addition Systems. The current chapter pursues this question further, exhibiting connections between the satisfiability problem of LRV *over data trees* and the coverability problem for branching counter systems. In order to obtain connections with branching Vector Addition Systems with States, or branching VASS [105], we also introduce a restriction where data tests are limited to use only one variable, that is, they are of the form $v \approx \mathrm{EF}v$. We denote this restriction by $\mathrm{LRV}^{\mathrm{D}}$. This symbiotic relation between counter systems and logics leads us to consider some natural extensions of both the logic and the branching counter systems. In particular, we introduce a new model of branching counter system of independent interest, with decidable coverability and control-state reachability problems.

4.1.1 Related work

The work most closely related to the topic of this chapter is the one originated by Demri et al. in [34, 35] and pursued in [36]. These works study the satisfiability problem for temporal logics on *data words*, extended with the ability to test whether a data value is repeated in the past/future. Indeed, our current study is motivated by the deep correlations evidenced by these works between counter systems and simple temporal logics on data words. The present chapter expands this analysis to *branching* logics and counter systems.

There are several works showing links between reachability-like problems for counter systems and the satisfiability problem of logics on data trees. The first prominent example is that satisfiability for Existential MSO with two variables on data words —notated $\mathrm{EMSO}^2(+1, <, \sim)$— corresponds precisely to reachability of VASS [18], in the sense that there are reductions in both directions. On the other hand, EMSO^2 over (unranked) *data trees* was shown to have tight connections with the reachability problem for an *extension* of BVASS [65], called 'EBVASS'. This extension has features which are very close to the model we introduce here, MVASS, but it does not capture, nor is captured by, MVASS. One can draw a parallel between the situation of the satisfiablity for EMSO^2 and for LRV: while on data words both are inter-reducible to VASS, the extension to data trees is non-trivial,

and they no longer correspond to BVASS, but to *extensions* thereof.

Regarding the logic in which we have focused so far in this thesis, research has been done on the satisfiability problem for XPath$_=$ on *unranked* data trees, and many decidable fragments have been identified [46, 47, 43, 50, 65]. However, most decidability procedures known so far rely on the fact that the data trees are unranked: the branching of the witnessing tree sought by the algorithms depends on the formula, sometimes polynomially (*e.g.*. for the fragment containing child and descendant axes [43]) or sometimes non-primitive-recursively (*e.g.*, for the 'vertical'[17] or 'forward'[18] fragment [50, 46]). On the other hand, on *ranked* data trees, it is known that:

- the satisfiability problem for XPath$_=$ with strict descendant (usually written $\downarrow_+$) has already a non-primitive-recursive lower bound in complexity, as can be seen by adapting techniques used for data words [49];
- the aforementioned logic is decidable, even if extended with 'child' and 'next-sibling' axes, as a consequence of the same logic being decidable on unranked data trees [46], added to the fact that the logic can express that the data tree has rank bounded by k for any fixed k;
- the satisfiability problem for XPath$_=$ with strict descendant and ancestor is undecidable, again by adapting known techniques on data words [49].

Modulo a simple coding, the central logic of this chapter, LRV, is captured by a fragment of regular-XPath$_=$, here called reg-XPath$_{\text{LRV}}$, on data trees where path expressions are allowed to use Kleene star on any expression (this what we denote by 'regular' XPath$_=$), and where all data tests are of the form $\langle \varepsilon \star \downarrow_* [\varphi] \rangle$ or $\langle \downarrow^n [\varphi] \star \downarrow^m [\psi] \rangle$ for some $n, m \in \mathbb{N}$ and $\star \in \{=, \neq\}$. There are, however, three provisos for this statement. First, in the aforementioned works on XPath$_=$, as in the past chapters of this thesis, the data model consists of structures whose every position carries exactly *one* data value. In the present chapter we study 'multi-attributed' data trees where, essentially, each node carries a set of pairs 'attribute:value'. However, by means of a simple coding, such as putting every 'attribute:value' as a leaf of the corresponding node, one can easily translate LRV-formulas to XPath$_=$-formulas. Second, our LRV-formulas are of the form $u \approx \mathrm{EF} v$ stating that the current data value under attribute u is repeated in a node x of the subtree under attribute v, but one cannot test that some property ψ further holds at the repeating node x. However, it was shown in [36] that one can extend the logic with this power, obtaining formulas of the form $x \approx \mathrm{EF} y[\psi]$, since this extended logic is PTIME-reducible to the logic without these tests. Third, the LRV-formulas cannot test for regular properties on the labeling of paths, and thus there is no precise characterization in terms of a natural fragment of regular-XPath$_=$, but one could add regular path tests to LRV to match the expressive power of reg-XPath$_{\text{LRV}}$ without changing any of our results.

[17] A nomenclature that collides with ours of 'vertical' as XPath$_=(\uparrow\downarrow)$; in [50], vertical XPath$_=$ contains the reflexive transitive closure of $\downarrow$ and $\uparrow$, that is: $\downarrow_*$ and $\uparrow^*$.

[18] The fragment with axes $\rightarrow$ (right sibling), $\downarrow$, and their reflexive transitive closures.

In fact, the fragment reg-XPath$_{\text{LRV}}$ extends also the fragment DataGL, already mentioned in Chapter 2, which is considered in [10] and [47] and contains only data tests of the form $\langle \varepsilon \star \downarrow_* [\varphi] \rangle$, a fragment which is known to be PSPACE-COMPLETE on unranked data trees [47].

It is not hard to see that the satisfiability problem of LRV on unranked data trees is PSPACE-COMPLETE following the techniques from [47]. On the other hand, on ranked data trees we know, by the discussion above, that if we would allow intermediate tests in a way to be able to encode the expressive power of XPath($\downarrow_+$) we would have a non-primitive recursive lower bound. It is therefore natural to limit the navigation *disallowing* intermediary tests. This natural fragment was already studied on data words [36], and we now study it on data trees.

4.1.2 Contributions

The aim of this chapter is to study the basic mechanism of "data repetition", common to many logics studied on data trees. For this, we study a basic logic that can navigate the structure of the tree through the use of CTL-like modalities, and on the other hand can make "data tests" by asking whether a data value is repeated in a subtree. More concretely, the data tests are formulas of the form $u \approx \mathrm{EF} v$, stating that the data value stored in attribute (also called *variable* here) u of the current node is equal to the data value stored in attribute v of some descendant. This *logic of repeating values*, or LRV, has been the center of a line of investigation studied in [35, 36] on *data words*, evidencing tight correspondences between reachability problems for Vector Addition Systems and the satisfiability problem. This chapter pursues this question further, exhibiting connections between the satisfiability problem of LRV *over data trees* and the bottom-up coverability problem for branching counter systems. In order to obtain connections with branching Vector Addition Systems with States, or branching VASS [105], we also introduce a restriction, denoted by LRV$^{\text{D}}$, where tests of the form $u \approx \mathrm{EF} v$ are only allowed when $u = v$.

While the extension of the logic LRV from words to trees is a very natural one, the techniques needed to encode the satisfiability of the logic into a counter system are not simple extensions from the ones provided on data words. The reason for this difficulty is manyfold: *a*) the fact that now the future is non-linear in addition to the possibility of having a data value repeating at several descendants in *different* variables, makes the techniques of [36] for propagating values of configurations impractical; *b*) further, this seems to be impossible for the case of data trees, and we could only show a reduction for the fragment LRV$^{\text{D}}$; *c)* in order to reduce the satisfiability problem for the full logic we need to augment the power of branching VASS with the possibility of 'merging' counters in a more powerful way, somewhat akin to what has been done for encoding the satisfiability for FO^2 [65].

We show that the satisfiability problem of our fragment LRV$^{\text{D}}$ is decidable, by reducing it to the control-sate reachability problem of VASS. The symbiotic relation between counter systems and logics leads us to consider some natural extensions of both the logic

and the branching counter systems. One extension results from the use of the operator $\mathrm{AG}_{\approx v}(\varphi)$, which expresses "every descendant with the same v-attribute verifies φ; we prove that the addition of positive instances of this operator to $\mathrm{LRV}^{\mathrm{D}}$ gives a logic with a satisfiability problem *equivalent* to that VASS. Furthermore, we introduce MVASS, a new model of branching counter system of independent interest that, as we show, captures the LRV satisfiability problem and has decidable coverability and control-state reachability problems.

4.1.3 Organization

This chapter is organized as follows:

- In §4.3 and §4.2 we introduce some basic notation and definitions. In the first part of §4.4 we introduce our models of branching counter systems and the related decision problems.
- In §4.5 we show that the satisfiability for $\mathrm{LRV}^{\mathrm{D}}$ on k-ranked data trees is reducible, in exponential space, to the control-state reachability problem for VASS_k (*i.e.*, Branching VASS of rank k). Since the control-state reachability problem is decidable [105] in 2ExpTime [37], this reduction yields a decision procedure.
- In §4.6 we consider the addition of an operator $\mathrm{AG}_{\approx v}(\varphi)$ expressing "every descendant with the same v-attribute verifies φ", and we show that the logic resulting from adding positive instances of this operator is equivalent to the control-state reachability for Branching VASS, that is, there are reductions in both directions.
- In §4.4.2 we introduce an extension of Branching VASS, called *Merging* VASS or MVASS. This model allows for merging counters in branching rules in a form which is not necessarily component-wise, allowing for some weak form of counter transfers. In §4.4.4 we show that the bottom-up coverability (and control-state reachability) problem for MVASS is in 3ExpTime. This is arguably a model of independent interest.
- In §4.7 we show that the satisfiability for LRV on k-ranked data trees can be reduced to the control-state reachability for MVASS_k. As in the case of $\mathrm{LRV}^{\mathrm{D}}$, this yields a decision procedure.

4.2 Preliminaries

We now give some basic notations that are used in this chapter. Let $\mathbb{N}^+ = \{1, 2, \dots\}$, $\mathbb{N} = \mathbb{N}^+ \cup \{0\}$, and $\underline{n} = \{1, \dots, n\}$ for every $n \in \mathbb{N}$. We use the bar notation $\bar{x}$ to denote a tuple of elements, where $\bar{x}[i]$, for $i > 0$, refers to the i-th element of the tuple. For any pair of vectors $\bar{x}, \bar{y} \in \mathbb{Z}^k$ we write $\bar{x} \leq \bar{y}$ if $\bar{x}[i] \leq \bar{y}[i]$ for all $1 \leq i \leq k$ (that is, in this context $\leq$ represents the product ordering of tuples). The constant $\bar{\emptyset}$ refers to the (unique) vector

of dimension 0, and the constant $\bar{e}_i$ refers to the vector (whose dimension will always be clear from the context) so that $\bar{e}_i[i] = 1$ and $\bar{e}_i[j] = 0$ for all $j \neq i$. We write $\bar{0}$ for the tuple of all 0's (the dimension being implicit from the context). We write $\uparrow\bar{x}$ to denote $\{\bar{x}' \in \mathbb{Z} \mid \bar{x}' \geq \bar{x}\}$ and we generalize it to any set $S \subseteq \mathbb{Z}$ in the usual way: $\uparrow S = \bigcup_{\bar{x}\in S} \uparrow\bar{x}$.

A **linear set** of dimension k is a subset of $\mathbb{N}^k$ which is either empty or described as $\{\bar{v}_0 + \alpha_1\bar{v}_1 + \cdots + \alpha_n\bar{v}_n \mid \alpha_1, \ldots, \alpha_n \in \mathbb{N}\}$ for some $n \in \mathbb{N}$ and $\bar{v}_0, \ldots, \bar{v}_n \in \mathbb{N}^k$. Henceforward we assume that linear sets are represented by the **offset** $\bar{v}_0$ and the **generators** $\bar{v}_1, \ldots, \bar{v}_n$, where numbers are represented in binary. For ease of writing we will denote a linear set like the one above by "$\bar{v}_0 + \{\bar{v}_1, \ldots, \bar{v}_n\}^*$".

In this chapter we generalize our definition of data trees so that many data values can be associated to a single node. We fix for all this chapter an infinite domain of **data values** $\mathbb{D}$. A **multidata tree** (or simply a **data tree** for this chapter) of **rank** k over a finite set of labels $\mathbb{A}$ and a finite set of attributes $\mathbb{V}$, is a finite tree whose every node x contains a pair $(a, \mu) \in \mathbb{A} \times \mathbb{D}^{\mathbb{V}}$ and has no more than k children. In general, a will be called the **label** of x and $\mu(v)$ will be called the **data value** of attribute $v \in \mathbb{V}$ at x. The ***i*-ancestor** of a node x of a data tree T is the ancestor at distance i from x (*i.e.*, the 1-ancestor is the parent); while the ***i*-descendant**s of x are all the descendants of x at distance i.

4.3 Logic of repeating values on data trees

We will work with a temporal logic using CTL* modalities [93, 40] to navigate the tree —although this is not really essential to our results, in the sense that any other MSO definable data-blind operators could also be added to the logic obtaining similar results. The **Logic of Repeating Values** LRV contains the typical modalities from CTL*, such as EF, AF, EU, etc. as well as the possibility to test for the label of the current node, and *data tests*. Data tests are restricted to being very basic, as in [35], of the form "$u \approx \mathrm{EF}v$" stating "the data value of attribute u appears again at the attribute v of some descendant", or "$u \not\approx \mathrm{EF}v$" stating "there is a descendant node whose attribute v contains a different data value from the data value of the attribute u of the current node". Since LRV is closed under Boolean connectives, this means we can also express, for instance, that attribute u of all descendants have the same data value as the current node's: $\neg(u \not\approx \mathrm{EF}u)$.

Formally, formulas of **LRV** are defined by

$$\varphi ::= a \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathrm{EU}(\varphi, \varphi) \mid u \approx \mathrm{EF}v \mid u \not\approx \mathrm{EF}v \mid u \approx \mathrm{EX}^i v \mid u \not\approx \mathrm{EX}^i v,$$

where a ranges over a finite set of labels $\mathbb{A}$, u, v range over a finite set of attribute variables $\mathbb{V}$ (also called just 'variables'), and $i \in \mathbb{N}^+$. Given a data tree T and a node x of T, the satisfaction relation $\models$ is defined in the usual way: $T, x \models a$ if a is the label of x; $T, x \models u \approx \mathrm{EF}v$ [resp. $T, x \models u \not\approx \mathrm{EF}v$] if there is a strict descendant y of x so that the u-attribute of x has the same [resp. different] data value as the v-attribute of y; $T, x \models u \approx \mathrm{EX}^i v$ [resp. $T, x \models u \not\approx \mathrm{EX}^i v$] if there exists an i-descendant of x whose v-attribute is equal [resp. distinct] to the u-attribute of x; and $T, x \models \mathrm{EU}(\varphi, \psi)$ if there is

some strict descendant y of x so that $T, y \models \varphi$ and every other node z strictly between x and y verifies $T, z \models \psi$. Note that the remaining CTL* modalities (EX, EG, EF, AX, AG, AF, AU) can be expressed using EU[19].

We call $\mathbf{LRV}^{\mathbf{D}}_{\boldsymbol{n}}$ the logic using at most n attribute variables, whose only admissible data tests are of the form $u \approx \mathrm{EF}u$, $u \not\approx \mathrm{EF}u$, $u \approx \mathrm{EX}^i u$ or $u \not\approx \mathrm{EX}^i u$ (same variable in the left and right sides). Intuitively, this corresponds to the restriction where each attribute variable ranges over a *disjoint* set of data values (hence the letter 'D').

4.4 Models of branching counter systems

We present the models of counter systems we are going to work with. The first one is a well-known model, usually known as Branching Vector Addition System with States, or "BVASS", while the second one is a useful extension of the first one where the split/merge operation of the counters is controlled by the use of linear sets.

4.4.1 Branching VASS

A **VASS of rank k and dimension n**, or **nVASS$_k$**, is a tuple $\mathcal{A} = \langle Q, U, B \rangle$, where Q is a finite set of states, $U \subseteq Q \times \mathbb{Z}^n \times Q$ is a set of unary rules, and $B \subseteq Q \times Q^{\leq k}$ is a finite set of branching rules. We notate $q \xrightarrow{\bar{v}} q'$ for a unary rule $(q, \bar{v}, q') \in U$, $q \to (q_1, \ldots, q_i)$ for a branching rule $(q, q_1, \ldots, q_i) \in B$ with $i \geq 1$, and $q \to \bar{\emptyset}$ for a branching rule in $Q \times Q^0$. A **configuration** is an element from $\mathit{Confs} := Q \times \mathbb{N}^n$. For a configuration $(q, \bar{n})$ we often use the term "**counter i**" instead of "$n[i]$" (in the case $n = 1$ we speak of *the* counter).

A **derivation tree** [resp. **incrementing derivation tree**] is a finite tree $\mathcal{D}$ whose every node x is either

- labeled with a pair $(p \xrightarrow{\bar{v}} p', (q, \bar{n})) \in U \times \mathit{Confs}$ so that $p \xrightarrow{\bar{v}} p'$ is a unary rule of U, $p = q$ and it has exactly one child, which is labeled $(r_1, (p_1, \bar{n}_1))$ so that $p' = p_1$ and

$$\bar{n} + \bar{v} = \bar{n}_1 \quad [\text{resp. } \bar{n} + \bar{v} \leq \bar{n}_1]; \tag{25}$$

- or labeled with a pair $((p, \bar{q}), (q, \bar{n})) \in B \times \mathit{Confs}$ so that $p \to \bar{q}$, with $\bar{q} \in Q^{k'}$ for some $k' \leq k$, is a branching rule of B, $p = q$ and it has exactly k' children, labeled $(r_1, (p_1, \bar{n}_1)), \ldots, (r_{k'}, (p_{k'}, \bar{n}_{k'}))$ so that $\bar{q} = (p_1, \ldots, p_{k'})$ and

$$\bar{n} = \sum_{i \leq k'} \bar{n}_i \quad [\text{resp. } \bar{n} \leq \sum_{i \leq k'} \bar{n}_i]. \tag{26}$$

We emphasize that the configurations only contain *positive* vectors; no node in a derivation tree can ever have a counter with negative value. Note that leaf nodes are necessarily

[19] $\mathrm{EX}\varphi = \mathrm{EU}(\varphi, \bot)$, $\mathrm{EF}\varphi = \mathrm{EU}(\varphi, \top)$, $\mathrm{EG}\varphi = \mathrm{EU}(\varphi \wedge \neg\mathrm{EX}\top, \varphi)$, $\mathrm{AU}(\varphi, \psi) = \neg\mathrm{EU}(\neg\psi \wedge \neg\varphi, \neg\psi) \wedge \neg\mathrm{EG}(\neg\varphi)$, etc.

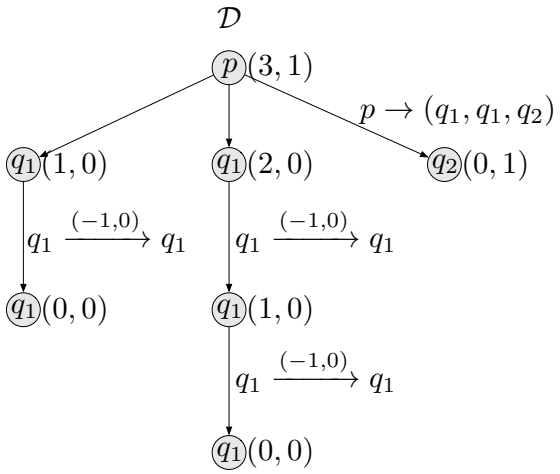


Figure 42: A derivation tree $\mathcal{D}$ for the 2VASS$_3$ $\mathcal{A} = \langle Q, U, B\rangle$ with $Q = \{p, q_1, q_2, q_3\}$, $U = \{(q_1, (-1, 0), q_1), (q_2, (-5, -1), q_3)\}$, and $B = \{(p, (q_1, q_1, q_2))\}$.

labeled with rules of the form $q \to \bar{\emptyset} \in B$. Without loss of generality we will assume that the system contains rules $q \to \bar{\emptyset}$ for every state q.

In Figure 42 we show an example of a derivation tree for a particular VASS of rank 3 and dimension 3 (that is, a 2VASS$_3$). For the sake of clarity, we write the labels of each node as follows: the configuration part is writen next to the node, and the part with an unary or binary rule is represented in the edges. We do not write the labels of the leaf nodes. In Figure 43 we show an incrementing derivation tree for the same VASS and starting from a node with the same configuration as in the previous example.

4.4.2 Merging VASS

We present an extension of the model above where the branching rules, now called *merging rules*, are more powerful: they allow us to reorganize the counters. Whereas in an (incrementing) derivation tree for VASS$_k$ the component i of the configuration of a node depends only on the component i of its children and the rule applied, MVASS$_k$ allows to have transfers *between components*. However, these transfers have some restrictions — otherwise the model would have non-elementary or undecidable coverability/reachability problems [76]. First, transfers between components are 'weak', in the sense that we cannot force a transfer of the whole value of a coordinate i to a distinct coordinate j of a child, we can only make sure that part of it will be transferred to component j and part of it will remain in component i. Second, these weak transfers can only be performed for any pair of coordinates i, j adhering to a partial order, where transfers occur from a bigger component to a smaller one.

A **Merging-VASS** of rank k and dimension n, or $\boldsymbol{n}$**MVASS**$_{\boldsymbol{k}}$, is a tuple of the form $\mathcal{A} = \langle Q, U, M, \preceq\rangle$, where $\preceq$ is partial order on $\underline{n}$, Q and U are as before, and M is a set of

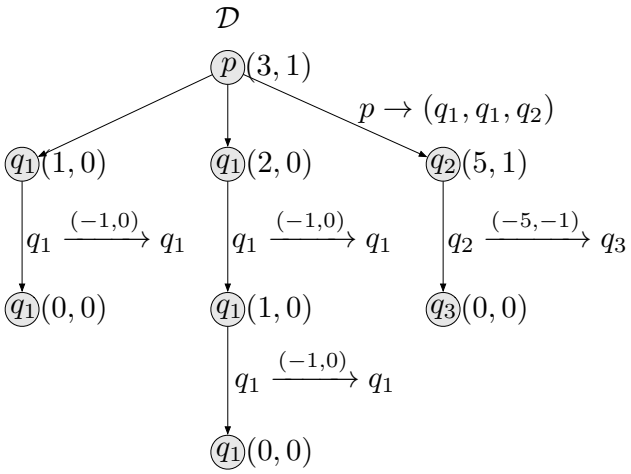


Figure 43: An *incrementing* derivation tree $\mathcal{D}$ for the 2VASS$_3$ $\mathcal{A} = \langle Q, U, B\rangle$ with $Q = \{p, q_1, q_2, q_3\}$, $U = \{(q_1, (-1,0), q_1), (q_2, (-5,-1), q_3)\}$, and $B = \{(p, (q_1, q_1, q_2))\}$. Observe that $\mathcal{D}$ is an increasing derivation tree, as $(3,1) \leq (1,0) + (2,0) + (5,1)$, but it is not an ordinary derivation since $(3,1) \neq (1,0) + (2,0) + (5,1)$.

merging rules of the form $(q, S, \bar{q})$ where $q \in Q$, $\bar{q} \in Q^{k'}$ with $k' \leq k$, and $S \subseteq \mathbb{N}^{n\cdot(k'+1)}$ is a linear set of dimension $n \cdot (k'+1)$ of the form $\bar{0} + (B \cup S_0)^*$, where

1. all the elements of B are of the form $(\bar{e}_i, \bar{x}_1, \ldots, \bar{x}_{k'})$, where for each $1 \leq \ell \leq k'$, $\bar{x}_\ell \in \mathbb{N}^n$ is either $\bar{0}$ or $\bar{e}_j$ for some $j \prec i$; and
2. S_0 consists of the following $k' \cdot n$ vectors

$$S_0 = \bigcup_{1\leq i\leq n} \{(\bar{e}_i, \bar{e}_i, \bar{0}, \bar{0}, \ldots, \bar{0}), (\bar{e}_i, \bar{0}, \bar{e}_i, \bar{0}, \ldots, \bar{0}), \ldots, (\bar{e}_i, \bar{0}, \ldots, \bar{0}, \bar{e}_i)\}. \tag{27}$$

The idea is that in point 1 we allow to transfer contents from component i to components of smaller order. For example, on dimension 3 and rank 2, a vector

$$\bar{v} = (1,0,0)(0,1,0)(0,0,1)$$

in B would imply that during the merge one can transfer a quantity $m > 0$ from component 1 of the father into component 2 of the first child and component 3 of the second child, assuming $2, 3 \prec 1$). On the other hand, point 2 tells us that for every i we can always have some quantity of component i that is *not* transfered to other components, *i.e.*, that stays in component i. Continuing our example, the children configurations $(m, m'+s, t)$ and $(m, s, m'+t)$ can be merged into $(m+m', s, t)$ for every $m, m', s, t \geq 0$, using the vector $\bar{v}$ and S_0.

A **derivation tree** [resp. **incrementing derivation tree**] is defined just as before, with the sole difference being that condition (26) is replaced with

$$(\bar{n}, \bar{n}_1, \ldots, \bar{n}_{k'}) \in S$$

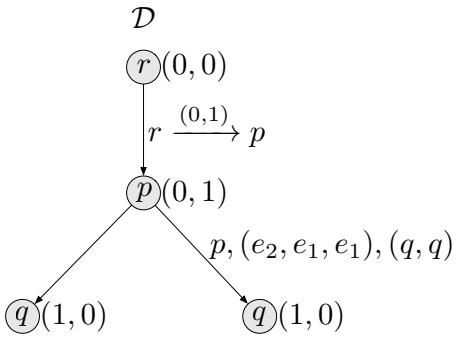


Figure 44: A derivation tree $\mathcal{D}$ for the 2MVASS$_2$ $\mathcal{A} = \langle Q, U, M, \preceq \rangle$ with $Q = \{r, p, q\}$, $U = \{(r, (0,1), p), (q, (-1,-1), q)\}$, $M = \{(p, S_1, (q,q)), (p, S_2, (q))\}$, and $\preceq$ defined as $\{(1,1), (1,2), (2,2)\}$. S_1 is the linear set with offset $\bar{0}$ generated by S_0 and $B_1 = \{(e_2, e_1, e_1)\}$, while S_2 is generated by S_0 and $B_2 = \{(e_2, e_1)\}$. The second rule of M is not used in this derivation tree.

$$[\text{resp. } (\bar{n}, \bar{n}'_1, \ldots, \bar{n}'_{k'}) \in S \text{ for } (\bar{n}'_1, \ldots, \bar{n}'_{k'}) \leq (\bar{n}_1, \ldots, \bar{n}_{k'})]. \tag{28}$$

Notice that this is a generalization of VASS$_k$. Indeed, VASS$_k$ corresponds to the restriction where all the k'-ary merging rules have $S = \bar{0} + S_0^*$ for S_0 as defined in (27). Note that an (incrementing) derivation tree for nVASS$_k$ is, in particular, an (incrementing) derivation tree for nMVASS$_k$. As before, we assume that there are always rules $(q, \emptyset, \emptyset)$ for every state q.

In Figure 44 we show an example of a derivation tree for a MVASS of dimension 2 and rank 2. Observe that, just by looking at the corresponding configurations of the nodes, this tree cannot be a derivation tree for VASS, since $(0,1) \not\leq (1,0) + (1,0)$. Figure 45 shows a related example of an incrementing derivation tree.

Jacquemard et al. [65] study an extension of BVASS, 'EBVASS', in relation to the satisfiability of FO$^2(<, +1, \sim)$ over *unranked* data trees. EBVASS has some features for merging counters. While MVASS and EBVASS are incomparable in computational power, it can be seen that without the restriction $j \prec i$ in condition 1, MVASS would capture EBVASS. In fact, this condition is necessary for the (elementary) decidability of the coverability problem for MVASS, while the status of the coverability problem for EBVASS is unknown.

4.4.3 Decision problems

Given a counter system $\mathcal{A}$, a set of states $\widehat{Q}$, and a configuration $(q, \bar{n})$ of $\mathcal{A}$, we write $\boldsymbol{(q, \bar{n}) \rightsquigarrow_{\mathcal{A}} \widehat{Q}}$ [resp. $\boldsymbol{(q, \bar{n}) \rightsquigarrow^+_{\mathcal{A}} \widehat{Q}}$] if there exists a derivation tree [resp. incrementing derivation tree] for $\mathcal{A}$ with root configuration $(q, \bar{n})$, so that all the leaves have configurations from $\widehat{Q} \times \{\bar{0}\}$. The reachability and incrementing reachability problems are defined as follows:

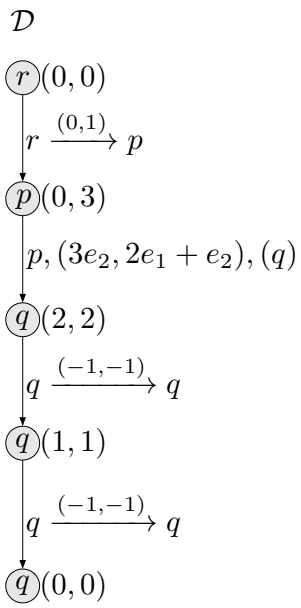


Figure 45: An *incrementing* derivation tree $\mathcal{D}$ for the 2MVASS$_2$ $\mathcal{A} = \langle Q, U, M, \preceq \rangle$ with $Q = \{r, p, q\}$, $U = \{(r, (0,1), p), (q, (-1,-1), q)\}$, $M = \{(p, S_1, (q, q)), (p, S_2, (q))\}$, and $\preceq$ defined as $\{(1,1), (1,2), (2,2)\}$. S_1 is the linear set with offset $\bar{0}$ generated by S_0 and $B_1 = \{(e_2, e_1, e_1)\}$, while S_2 is generated by S_0 and $B_2 = \{(e_2, e_1)\}$. The first rule of M is not used in this incrementing derivation tree. Notice that we are using twice the incrementing nature of the derivation: once at the beginning with a unary rule and once afterwards for a branching rule.

nVASS$_k$ reachability problem [resp. **nVASS$_k$ incrementing reachability problem**]	
INPUT :	an nVASS$_k$ $\mathcal{A}$ with states Q, a set of states $\widehat{Q} \subseteq Q$, and a configuration $(q, \bar{n})$ of $\mathcal{A}$.
OUTPUT :	'Yes' iff $(q, \bar{n}) \rightsquigarrow_{\mathcal{A}} \widehat{Q}$ [resp. $(q, \bar{n}) \rightsquigarrow^{+}_{\mathcal{A}} \widehat{Q}$].

Observe that when $k = 1$ these problems are equivalent to the reachability and coverability problems for Vector Addition Systems with States, respectively.

The nMVASS$_k$ reachability problem and nMVASS$_k$ incrementing reachability problem are defined just as before but considering $\mathcal{A}$ to be an nMVASS$_k$ instead of a nVASS$_k$:

nMVASS$_k$ [incrementing] reachability problem	
INPUT :	an nMVASS$_k$ $\mathcal{A}$ with states Q, a set of states $\widehat{Q} \subseteq Q$, and a configuration $(q, \bar{n})$ of $\mathcal{A}$.
OUTPUT :	'Yes' iff $(q, \bar{n}) \rightsquigarrow_{\mathcal{A}} [\rightsquigarrow^{+}_{\mathcal{A}}]\widehat{Q}$.

For succinctness, we will often refer to the reachability and incrementing reachability problems as Reach( ) and Reach^{+}( ), respectively. We also remark that the incrementing reachability problem can be seen as a restatement of the *bottom-up coverability problem*[20]. In particular, if $(q, \bar{n}) \rightsquigarrow^{+}_{\mathcal{A}} \widehat{Q}$ and $\bar{m} \leq \bar{n}$ then $(q, \bar{m}) \rightsquigarrow^{+}_{\mathcal{A}} \widehat{Q}$.

We define the **control-state reachability problem** CSReach as the problem of, given $\mathcal{A}, q, \widehat{Q}$, whether $(q, \bar{n}) \rightsquigarrow_{\mathcal{A}} \widehat{Q}$ for *some* $\bar{n}$. It is easy to see that this problem is equivalent to the problem of whether $(q, \bar{0}) \rightsquigarrow^{+}_{\mathcal{A}} \widehat{Q}$[21], and it is this last formulation the one we will use for our reductions:

nVASS$_k$ [nMVASS$_k$] control-state reachability problem	
INPUT :	an nVASS$_k$ [nMVASS$_k$] $\mathcal{A}$ with states Q, a set of states $\widehat{Q} \subseteq Q$, and a state q of $\mathcal{A}$.
OUTPUT :	'Yes' iff $(q, \bar{0}) \rightsquigarrow^{+}_{\mathcal{A}} \widehat{Q}$.

In [37] it is studied the coverability problem (or equivalently, the incrementing reachability problem) for a single-state formulation, called **BVAS**. A BVAS consists of a tuple $\langle n, R_1, R_2 \rangle$, where R_1 is a set of unary rules, R_2 is a set of binary rules (both rules included in $\mathbb{Z}^n$ which add up a vector). The *size* of a given BVAS is defined as $n\ell$, where ℓ represents the maximum binary size of an entry in $R_1 \cup R_2$.

[20] The **bottom-up coverability problem** asks, given $\mathcal{A}$, $\widehat{Q}$, and $(q, \bar{n})$, whether $(q, \bar{n}') \rightsquigarrow_{\mathcal{A}} \widehat{Q}$ for some $\bar{n}' \geq \bar{n}$.

[21] If the answer is 'yes' for CSReach on $\mathcal{A}, q, \widehat{Q}$, then there is $\bar{n}$ such that $(q, \bar{n}) \rightsquigarrow_{\mathcal{A}} \widehat{Q}$. If $\bar{n} = \bar{0}$ we are done. Otherwise, as the derivation tree for CSReach must end with leaves with configurations in $\widehat{Q} \times \{\bar{0}\}$, some rule is used first in its derivation tree starting from q. Now replicate that derivation tree, but starting with counter $\bar{0}$ and using the 'incrementing' part to simulate having started with a configuration with counter $\bar{n}$; this is a solution for $(q, \bar{0}) \rightsquigarrow^{+}_{\mathcal{A}} \widehat{Q}$. The idea is similar for the direction $(q, \bar{0}) \rightsquigarrow^{+}_{\mathcal{A}} \widehat{Q} \Rightarrow$ CSReach.

Proposition 163. *[37] Coverability for* BVAS *is* 2ExpTime-complete. *If the dimension n is fixed, the problem is in* ExpTime.

4.4.4 Decidability of Reach$^+$(MVASS)

The arguments used in [37] to prove Proposition 163 can be adapted to show a similar result for MVASS: the Reach$^+$ and CSReach problems are in 3ExpTime:

Theorem 164. Reach$^+$(MVASS$_k$) *and* CSReach(MVASS$_k$) *are in* 3ExpTime *for every* $k \geq 1$. *If the dimension n is fixed, the problem is in* 2ExpTime.

In order to simplify some of the arguments we use for the proof, we work with a slightly different version of the incrementing reachability problem which can easily be shown to be equivalent to our previous definition:

MVASS$_k$ incrementing reachability problem	
INPUT :	an nMVASS$_k$ $\mathcal{A}$ with states Q, a set of states $\widehat{Q} \subseteq Q$, and a configuration $(q, \bar{n})$ of $\mathcal{A}$.
OUTPUT :	'Yes' iff $(q, \bar{n}') \leadsto^+_{\mathcal{A}} \widehat{Q}$ for some $\bar{n}' \geq \bar{n}$.

The difference is that we look for an incrementing derivation of a 'bigger' configuration $(q, \bar{n}')$ than the one $(q, \bar{n})$ received as input. But it is straightforward to see that this is essentially the same problem.

We say that a tree $\mathcal{D}$ is **an incrementing derivation tree for $(q, \bar{n}) \leadsto^+_{\mathcal{A}} \widehat{Q}$** if $\mathcal{D}$ is an incrementing derivation that is a witness for $(q, \bar{n}) \leadsto^+_{\mathcal{A}} \widehat{Q}$. Throughout this section we write 'derivation' as short for 'incrementing derivation tree'. Given a derivation $\mathcal{D}$ and a node x thereof, we write $\rho_{\mathcal{D}}(x)$ to denote the vector of the configuration at x and $\sigma_{\mathcal{D}}(x)$ to denote its state. We will usually write ϵ to denote the node at the root of $\mathcal{D}$. We adapt the main concepts of the 2ExpTime proof for Reach$^+$(VASS$_k$) of [37] to our setup. A **contraction** of a derivation $\mathcal{D}$ is the result of applying a finite number of times the following operation. Let x be a node of $\mathcal{D}$ with configuration $(q, \bar{n})$ and x' a descendant of x with configuration $(q', \bar{n}')$ so that $q = q'$. Consider the result of:

- replacing the subtree at x with the subtree at x' (*i.e.*, removing all descendants of x which are not descendants of x' and identifying x' with x);
- for every ancestor y of x with configuration $(p, \bar{m})$, replacing the configuration with $(p, \bar{m} + (\bar{n}' - \bar{n}))$.

We denote this substitution with $\mathcal{D}[x \leftarrow x']$. We say that a configuration $(q, \bar{n})$ is **bigger** than a configuration $(p, \bar{m})$ if $p = q$ and $\bar{n} \geq \bar{m}$. Note that if $\mathcal{D}$ is a derivation for $(q, \bar{n}) \leadsto^+_{\mathcal{A}} \widehat{Q}$ and the configuration at node x' is bigger than the configuration at an ancestor x of x', then $\mathcal{D}[x \leftarrow x']$ is a derivation for $(q, \bar{n}') \leadsto^+_{\mathcal{A}} \widehat{Q}$ for some $\bar{n}' \geq \bar{n}$. Thus, if $\mathcal{D}$ is a witness for the incrementing reachability problem instance $(\mathcal{A}, \widehat{Q}, (a, \bar{n}'))$, so is

$\mathcal{D}[x \leftarrow x']$. (Indeed, this is true both when $\mathcal{A}$ is a VASS or a MVASS.) For a set of unary rules $U \subseteq Q \times \mathbb{Z}^k \times Q$, let $\max(U^+) \in \mathbb{N}$ be the maximum positive value contained in unary rules; that is,

$$\max\big(U^+\big) \stackrel{\text{def}}{=} \max(\{0\} \cup \{\bar{v}[i] \mid (q, \bar{v}, q') \in U, i \in \underline{k}, \bar{v}[i] > 0\}).$$

For a derivation of $\mathcal{D}$ of a kMVASS$_n$ $\mathcal{A}$, and a set $I \subseteq \underline{k}$, we define the **restriction to** $\boldsymbol{I}$ of $\mathcal{D}$, and we note it $\mathcal{D}|_I$, as the result of

- replacing each configuration $(q, \bar{n})$ of a node with $(q, \bar{n}[I])$, where $\bar{n}[I] \in \mathbb{N}^{|I|}$ is the restriction of $\bar{n}$ to the component indices of I;
- replacing every unary rule $(q, \bar{v}, q')$ in a node with $(q, \bar{v}[I], q')$; and
- replacing every merging rule $(q, S, \bar{q})$ in a node with $(q, S[I], \bar{q})$, where $S[I] = \{\bar{s}[I] \mid \bar{s} \in S\}$.

In a similar way, we consider the restriction $\mathcal{A}|_I$ of the automaton $\mathcal{A}$ as the $|I|$-MVASS$_n$ resulting from replacing the rules as described above. Note that if I is $\preceq$-downward closed (*i.e.*, if $i \preceq j$ and $j \in I$, then $i \in I$) then $\mathcal{D}|_I$ is actually a derivation of $\mathcal{A}|_I$.

Lemma 165. *Let $\mathcal{A} = \langle Q, U, M, \leq\rangle$ be a MVASS$_1$ of dimension k, and let $\mathcal{D}$ be a derivation for $(q, \bar{n}) \rightsquigarrow^+_{\mathcal{A}} \widehat{Q}$. Then, there is a contraction of $\mathcal{D}$ which is a derivation for $(q, \bar{n}') \rightsquigarrow^+_{\mathcal{A}} \widehat{Q}$ for some $\bar{n}' \geq \bar{n}$ and whose length is bounded by $(\max(U^+) + \max(\bar{n}))^{2^{p(k)}}$ for a polynomial $p(\ )$.*

Proof. This proof follows arguments similar to those from Rackoff [94, Section 3] as described in [37, Lemma 4]. Let $m(\mathcal{D}, \bar{n}, \widehat{Q}, \mathcal{A})$ be the smallest height of a contraction of $\mathcal{D}$ that is a derivation for $(q, \bar{n}') \rightsquigarrow^+_{\mathcal{A}} \widehat{Q}$ for some $\bar{n}' \geq \bar{n}$. For $L, k \in \mathbb{N}$ we let:

$$M_L(k) = \sup\ \{m(\mathcal{D}', \bar{n}, \widehat{Q}, \mathcal{A}) : \mathcal{D}' \text{ is a derivation for } (q, \bar{n}') \rightsquigarrow^+_{\mathcal{A}} \widehat{Q}, \bar{n}' \geq \bar{n} \\ \text{and } |Q| \cdot (\max(U^+) + \max(\bar{n}) + 1) \leq L\}.$$

We show that the number $M_L(k)$ is well-defined in the next lemma.

In the context of the partially ordered set $(\underline{k}, \preceq)$, let $\downarrow i = \{j \in \underline{k} \mid j \prec i\}$ for every $i \in \underline{k}$.

Lemma 166. *For all $L \in \mathbb{N}$, the following inequalities hold:*

$$M_L(k) \leq \begin{cases} L & \text{if } k = 0, \\ M_L(k-1) \cdot \prod_{i \in \underline{k}} B_i & \text{if } k \geq 1, \end{cases} \tag{29}$$

for $B_i = M_L(|\downarrow i|) \cdot \big(\prod_{j \succ i} B_j\big)^2 + L$, and $\prod \emptyset = 1$ by convention.

Proof. We proceed by induction on k. The case $k = 0$ is trivial, as there are no counters, and thus the height of minimal contractions is bounded by $|Q|$ by a pumping argument. For every $k \geq 1$, it is sufficient to prove that for every derivation $\mathcal{D}$ for $(q, \bar{n}') \leadsto^+_{\mathcal{A}} \widehat{Q}$ where $\bar{n}' \geq \bar{n}$ and $|Q| \cdot (\max(U^+) + \max(\bar{n}) + 1) \leq L$, the following inequality holds:

$$m(\mathcal{D}, \bar{n}, \widehat{Q}, \mathcal{A}) \leq M_L(k-1) \cdot \prod_{i \in \underline{k}} B_i. \tag{30}$$

For a set of components $I \subseteq \underline{k}$, we say that $\mathcal{D}$ is ***I*-bounded** if for every $i \in I$ and for every configuration $(q, \bar{v})$ of $\mathcal{D}$ we have $\bar{v}[i] < B_i$. We consider the following two cases: (a) $\mathcal{D}$ is $\underline{k}$-bounded, and (b) $\mathcal{D}$ is not $\underline{k}$-bounded. Assume that $\mathcal{D}$ has minimal height. We define $\rho_{\mathcal{D}}(x)$ [resp. $\sigma_{\mathcal{D}}(x)$] for any node x of $\mathcal{D}$ as the vector $\bar{v}$ [resp. state p] contained in the configuration $(p, \bar{v})$ of $\mathcal{D}$ at x.

(a) Assume that $\mathcal{D}$ is $\underline{k}$-bounded. Note that if $\rho_{\mathcal{D}}(x) = \rho_{\mathcal{D}}(x')$, $\sigma_{\mathcal{D}}(x) = \sigma_{\mathcal{D}}(x')$ and x is an ancestor of x' then the derivation $\mathcal{D}[x \leftarrow x']$ obtained by the contracting substitution is also a derivation for $(q, \bar{n}') \leadsto^+_{\mathcal{A}} \widehat{Q}$. By performing such substitutions repeatedly, we will eventually obtain a contraction of $\mathcal{D}$ that is a $\underline{k}$-bounded derivation for $(q, \bar{n}') \leadsto^+_{\mathcal{A}} \widehat{Q}$ with height bounded by

$$|Q| \cdot \prod_i B_i \leq L \cdot \prod_i B_i \leq M_L(k). \tag{31}$$

(b) Suppose now that $\mathcal{D}$ is not $\underline{k}$-bounded and suppose that it is minimal in the sense of contractions. Let $i_0 \in \underline{k}$ be a $\preceq$-minimal index so that $\mathcal{D}$ is $\uparrow i_0$-bounded for $\uparrow i_0 = \{j \mid j \succ i\}$. Note that

- for some node x of $\mathcal{D}$ we have $\rho_{\mathcal{D}}(x)[i_0] > B_{i_0}$; and
- for all $j \succ i_0$ and for every node y of $\mathcal{D}$ we have $\rho_{\mathcal{D}}(y)[j] \leq B_j$.

Let x_0 be the lowest node (*i.e.*, closest to the leaf) so that $\rho_{\mathcal{D}}(x_0)[i_0] > B_{i_0}$. We first bound the distance between x_0 and the root, and then we bound the distance between x_0 and the leaf.

(I) Consider the subderivation $\mathcal{D}_1$ from the root to x_0. We show that the height of $\mathcal{D}_1$ is bounded by

$$\mathit{contr} = M_L(| \downarrow i_0|) \cdot \prod_{j \succ i_0} B_j.$$

For the sake of contradiction, suppose that its height is larger than *contr*. Note that at each step of $\mathcal{D}_1$ we can increase component i_0 in at most

$$\mathit{incstep} = \sum_{j \succ i_0} B_j + \max(U^+). \tag{32}$$

If we restrict $\mathcal{D}_1$ to the components $\downarrow i_0$ we obtain a derivation for $\mathcal{A}|_{\downarrow i_0}$ with smaller dimension. In fact it is a derivation thanks to the inequality of (28) —all the increments coming from transfers from the components $j \succeq i_0$ can considered in the inequality.

By inductive hypothesis on $\mathcal{D}|_{\downarrow i_0}, \mathcal{A}|_{\downarrow i_0}$, there are two nodes y, y' (y ancestor of y') at distance $\leq$ *contr* from the root that can be contracted (*i.e.*, so that $\mathcal{D}|_{\downarrow i_0}[y \leftarrow y']$ is a derivation for $(q, \bar{n}') \rightsquigarrow^{+}_{\mathcal{A}|_{\downarrow i_0}} \widehat{Q}$ for $\bar{n}' \geq \bar{n}$), and whose values for all the components $j \succ i_0$ coincide (*i.e.*, $\rho_{\mathcal{D}}(y)[j] = \rho_{\mathcal{D}}(y')[j]$ for all $j \succ i_0$). The contraction of y, y' on $\mathcal{D}_1|_{\downarrow i_0}$ is a derivation of $\mathcal{A}|_{\downarrow i_0}$ by inductive hypothesis. Further, since the values of components $j \succ i_0$ coincide, the contraction of y, y' on $\mathcal{D}_1|_{\{j | j \neq i_0\}}$ is also a derivation. Finally, by (32), the increase from y to y' on component i_0 cannot be greater than $\mathit{contr} \cdot \mathit{incstep}$; and since

$$\begin{aligned}
\mathit{contr} \cdot \mathit{incstep} + \max(\bar{n}) &= M_L(|\downarrow i_0|) \cdot \prod_{j \succ i_0} B_j \cdot \sum_{j \succ i_0} B_j + \max(\bar{n}) \\
&\leq M_L(|\downarrow i_0|) \cdot \prod_{j \succ i_0} B_j \cdot \sum_{j \succ i_0} B_j + L \\
&\leq M_L(|\downarrow i_0|) \cdot \Big(\prod_{j \succ i_0} B_j\Big)^2 + L \\
&= B_{i_0}
\end{aligned}$$

we thus have that the contraction $\mathcal{D}'_1$ of y, y' on $\mathcal{D}_1$ is a derivation as well, for all the components. Further, the root of $\mathcal{D}'_1$ has a configuration greater than $(q, \bar{n})$ and the leaf remains unchanged, that is, it contains the configuration $(\rho_{\mathcal{D}_1}(x_0), \sigma_{\mathcal{D}_1}(x_0))$. Therefore, by minimality of $\mathcal{D}$ we have that $\mathcal{D}_1$ must have height bounded by *contr*.

(II) On the other hand, let $\mathcal{D}_2$ be the subderivation between x_0 and the leaf of $\mathcal{D}$. Since $\mathcal{D}_2$ is $(\uparrow i_0 \cup \{i_0\})$-bounded (except for the root), we have that $\mathcal{D}_2$ cannot have height larger than

$$\mathit{contr}' = M_L(|\downarrow i_0|) \cdot \prod_{j \succeq i_0} B_j$$

using similar arguments as before.

Thus, by (I) cum (II), we have that the height of $\mathcal{D} = \mathcal{D}_1\mathcal{D}_2$ is bounded by

$$\begin{aligned}
\mathit{contr} + \mathit{contr}' &= M_L(|\downarrow i_0|) \cdot \prod_{j \succ i_0} B_j \;+\; M_L(|\downarrow i_0|) \cdot \prod_{j \succeq i_0} B_j \\
&\leq M_L(k-1) \cdot \prod_{i} B_i = M_L(k)
\end{aligned}$$

□

Note that by definition of B_i, for any $\preceq$-minimal i, $B_i = M_L(0) + L = 2L$. For any other i, the number of recursive calls needed to compute B_i is bounded by $2^k \cdot k$. This is because at each recursive call for B_i we produce two instances of B_j for every $j \succ i$, and thus in the product each B_t with maximal t will have an exponent $2^{|\{j | i \prec j \preceq t\}|}$; repeating this for each such t (not more than k times) we obtain the bound. We then have, for every i, that $B_i \leq (M_L(|\downarrow i|))^{2^k \cdot k} + 2^{k+1} \cdot k \cdot L$, and thus

$$M_L(k) \leq M_L(k-1)^{2^k \cdot k^2} \cdot (2L)^{2^k \cdot k^2} + 2^{2^k} \cdot k^2 \cdot L.$$

Therefore, $M_L(k)$ is bounded by a function $L^{2^{p(k)}}$ for some polynomial $p(\,)$. □

We have now all the necessary elements to prove Theorem 164.

Proof of Theorem 164. Using Lemma 165, we show that if there is an incrementing derivation $\mathcal{D}$ for $(q, \bar{n}) \rightsquigarrow^+_{\mathcal{A}} \widehat{Q}$, where $\mathcal{A} = (Q, U, M, \preceq)$ is an nMVASS$_k$ counter system, then there is a contraction $\mathcal{D}'$ of $\mathcal{D}$ with height bounded doubly-exponentially in the dimension.

We show that if a nMVASS$_k$ $\mathcal{A} = (Q, U, M)$ has an incrementing derivation $\mathcal{D}$ for $(q, \bar{n}') \rightsquigarrow^+_{\mathcal{A}} \widehat{Q}$, $\bar{n}' \geq \bar{n}$, then there a contraction $\mathcal{D}'$ of $\mathcal{D}$ which is also an incrementing derivation for $(q, \bar{n}'') \rightsquigarrow^+_{\mathcal{A}} \widehat{Q}$, $\bar{n}'' \geq n$, whose height is bounded by

$$(\max(U^+) + \max \bar{n} + |Q|)^{2^{p(k)}} \tag{33}$$

for a polynomial function $p : \mathbb{N} \to \mathbb{N}$.

The next argument basically follows the schema (i)–(iii) of [37, p.7]. Let $\mathcal{D}$ be an incrementing derivation for $(q, \bar{n}') \rightsquigarrow^+_{\mathcal{A}} \widehat{Q}$, and let π be a root-to-leaf path of $\mathcal{D}$ which is larger than the bound. Let $\mathcal{A}'$ be a nMVASS$_k$ whose set of rules consists of:

- The unary rules $(q, \bar{v}, q')$ contained in the unary rules of π.
- Suppose we have a node x of π with configuration $((q, \bar{n}), (q, S, \bar{q}))$ and with children labeled $((q_1, \bar{n}_1), r_1), \ldots, ((q_s, \bar{n}_s), r_s)$, so that the next element after x in π is the j-th child of x. Further, suppose that this merging rule is preceded by a unary rule; that is, the parent x' of x is labeled with $((p, \bar{n}'), (p, \bar{w}, q))$ —it is not hard to see that without any loss of generality we can always assume that a merging rule is preceded by a unary rule. Let $B = \{\bar{b}_1, \ldots, \bar{b}_m\}$ be the basis of S, that is, $B^* = S$. Let $B' = \{\bar{b}_i \mid \bar{b}_i[h] = 0 \text{ for all } n \cdot (1 + j) + 1 \leq h \leq n \cdot (2 + j)\}$, and $B'' = B \setminus B'$. Note that B' is the set of bases that do not touch the j-th component. We then have

  $$(\bar{n}, \bar{n}_1, \ldots, \bar{n}_s) = \alpha_1 \bar{b}'_1 + \cdots + \alpha_{m'} \bar{b}'_{m'} + \beta_1 \bar{b}''_1 + \cdots + \beta_{m''} \bar{b}''_{m''}$$

  for $B' = \{\bar{b}'_1, \ldots, \bar{b}'_{m'}\}$ and $B'' = \{\bar{b}''_1, \ldots, \bar{b}''_{m''}\}$. Let $\bar{v}' = -(\alpha_1 \bar{b}'_1 + \cdots + \alpha_{m'} \bar{b}'_{m'}) \in \mathbb{Z}^{n \cdot (s+1)}$ and $\bar{v} \in \mathbb{Z}^n$ the restriction of $\bar{v}'$ to the first n components. Note that $\bar{v}$ contains non-positive entries only. Then, produce the unary rule $(p, \bar{w} + \bar{v}, q)$ and a merging rule (q, S', q') so that $S' \subseteq \mathbb{N}^{2n}$ is the restriction of S to the components corresponding to the j-th child.

Note that if we relabel accordingly π we obtain an incrementing derivation for $(q, \bar{n}') \rightsquigarrow^+_{\mathbb{A}'} \widehat{Q}$. Then, by Lemma 165, there is a contraction of π which is still a correct incrementing derivation for $(q, \bar{n}'') \rightsquigarrow^+_{\mathbb{A}'} \widehat{Q}$ for some $\bar{n}'' \geq \bar{n}$ and so that its length is at most

$$(\max(U'^+) + \max(\bar{n}) + 1)^{2^{p(k)}}$$

where U' is the set of unary rules of $\mathcal{A}'$. Note that $\max(U'^+) \leq \max(U^+)$ because we have only added unary rules with smaller positive entries.

We can then unfold back the subtrees hanging from nodes of π to obtain an MVASS incrementing derivation for $(q, \bar{n}'') \rightsquigarrow^+_{\mathbb{A}} \widehat{Q}$ whose number of leaves at height greater than

the bound (33) has decreased in at least 1. Repeating the same argument a finite number of times we obtain an incrementing derivation for $(q, \bar{n})$ of height bounded by (33).

Thus, to decide the incrementing reachability problem, it suffices to search for a derivation of doubly-exponential height, whose vectors may contain triply-exponential entries in principle. As a consequence of this, the verification of the existence of such a derivation can be performed in alternating double exponential space, as it is shown in [37, Theorem 8], and thus the incrementing reachability for MVASS is in 3ExpTime.

If n is fixed, the height of the witnessing derivation becomes singly exponential and thus the problem is in 2ExpTime (as explained in [37, Theorem 8]). □ □

4.5 Satisfiability of $\mathrm{LRV}^{\mathrm{D}}$ on data trees

We call SAT_k the satisfiability problem on finite k-ranked data trees. The main result of this section is the following.

Theorem 167. *SAT_k-$\mathrm{LRV}^{\mathrm{D}}_n$ is* ExpSpace*-reducible to* CSReach$(n\mathrm{VASS}_k)$.

In the proof of the theorem, the number of attribute variables of the formula will become the dimension of the VASS_k. Since the CSReach problem for VASS_k is decidable, this yields a decidable procedure for SAT_k-$\mathrm{LRV}^{\mathrm{D}}$ for every k. For the case $k = 1$, *i.e.*, on *data words*, it has been shown [36] that there is a reduction from SAT_1-LRV_n to CSReach$(2^n$-$\mathrm{VASS}_1)$, where the dimension of the VASS_1 is exponential in the number of variables. However, it is easy to see that the proof of [36] also yields a reduction from SAT_1-$\mathrm{LRV}^{\mathrm{D}}_n$ to CSReach$(n\mathrm{VASS}_1)$. Thus, this theorem has been shown for $k = 1$, and here we generalize it to $k > 1$. However, there are a number of problems that appear if one tries to "extend" the proof of [36] to the branching setup. In particular, the non-linearity of the future in addition to the possibility of having a data value repeating at several descendants in different variables, calls for a non-standard way of propagating the values of configurations, which is not contemplated in VASS_k. This is why we are only able to show the reduction for the 'disjoint' fragment $\mathrm{LRV}^{\mathrm{D}}$, and which leads us to consider the extended model MVASS_k in Section 4.7. This propagation problem does not appear when one only considers that the classes of different values are disjoint, that is, that all formulas of the type $v \star \mathrm{EF} w$ with $\star \in \{\approx, \not\approx\}$ have $v = w$, motivating the study of SAT_k-$\mathrm{LRV}^{\mathrm{D}}_n$.

For the proof of the theorem, we start in §4.5.1 by analyzing a restricted case, which serves as building block: the logic $\mathrm{LRV}^{\mathrm{D}-}_1$ whose only formulas are conjuncts of terms of the form $v \star \mathrm{EX}^i v$, $v \star \mathrm{EF} v$, or their negation, where $\star \in \{\approx, \not\approx\}$. We show that for any formula φ of $\mathrm{LRV}^{\mathrm{D}-}_1$, there is a $1\mathrm{VASS}_k$ $\mathcal{A}^k_\varphi = \langle Q, U, B\rangle$, a set of initial states $Q_0 \subseteq Q$, and a set of final states $\widehat{Q} \subseteq Q$ such that $\mathrm{SAT}_k(\varphi)$ iff there is a derivation tree with a starting node in $q_0 \in Q_0$ that is a solution to CSReach$(\mathcal{A}^k_\varphi, q_0, \widehat{Q})$—it is easy to see that this problem is equivalent to CSReach as stated in Section 4.4.3. We then extend this construction to the automaton $\mathcal{B}^k_\varphi$ in §4.5.2, enabling a reduction from the full logic $\mathrm{LRV}^{\mathrm{D}}_1$, but still restricted to only one variable. Finally, because of the disjointness of the variables,

it is easy to extend these constructions to the full logic $\mathrm{LRV}^{\mathrm{D}}_n$, and we do so in §4.5.3. In §4.5.4, using this reduction, we analyze the complexity of SAT_k-$\mathrm{LRV}^{\mathrm{D}}_n$; in order to do that, we reduce from the problem of CSReach for VASS into the problem of coverability for BVAS, and then use Proposition 163.

We now start with the results and proofs related to Theorem 167, starting with a simple logic and gradually increasing the complexity to finally obtain the desired result.

4.5.1 A simple logic: $\mathbf{LRV}^{\mathbf{D}-}_1$

We notate $\mathrm{LRV}^{\mathrm{D}-}_1$ to the logic which consists only of formulas $\varphi \in \mathrm{LRV}^{\mathrm{D}}_1$ that are conjuncts of terms of the form $v \star \mathrm{EX}^i v$, $v \star \mathrm{EF} v$, or their negation, where $\star \in \{\approx, \not\approx\}$.

In this section we show the following:

Proposition 168. *There is a reduction from* SAT_k-$\mathrm{LRV}^{\mathrm{D}-}_1$ *to* $\textsc{CSReach}(1\mathrm{VASS}_k)$.

That is, we show that for any formula φ of $\mathrm{LRV}^{\mathrm{D}-}_1$, there is a $1\mathrm{VASS}_k$ $\mathcal{A}^k_\varphi$ such that φ is satisfiable in a k-ranked data tree iff $\mathcal{A}^k_\varphi$ has a solution to the control-state reachability problem.

We begin with some definitions used for the construction of $\mathcal{A}^k_\varphi$, and then proceed to the proof of the reduction.

Valid (d,k)-frames. A **valid (d,k)-frame** (or often just **(d,k)-frame**) is a tuple $F = \langle N, E, \ell_1, \ell_2, \equiv \rangle$ such that

- $\langle N, E\rangle$ is a k-ranked ordered tree of height at most d, whose non-empty set of nodes is N and whose set of edges is E.
- $\ell_1 : N \to \{([v \approx \mathrm{EF}v], [v \not\approx \mathrm{EF}v]), ([v \approx \mathrm{EF}v], [\neg v \not\approx \mathrm{EF}v]), ([\neg v \approx \mathrm{EF}v], [v \not\approx \mathrm{EF}v]), ([\neg v \approx \mathrm{EF}v], [\neg v \not\approx \mathrm{EF}v])\}$ is a node-labeling function
- $\ell_2 : N \to \{\varepsilon, \oplus, \ominus\}$ is another node-labeling function
- $\equiv$ is an equivalence relation over N

where ℓ_1 satisfies the following validity conditions:

1. If $x \in N$ is such that $\ell_1(x) = ([\neg v \approx \mathrm{EF}v], [\neg v \not\approx \mathrm{EF}v])$ then x has no children.
2. If $x \in N$ is a leaf, and x is at distance $< d$ from the root of F, then $\ell_1(x) = ([\neg v \approx \mathrm{EF}v], [\neg v \not\approx \mathrm{EF}v])$.
3. Let y be a descendant of x, with $x \equiv y$, then $\pi_1(\ell_1(x)) = [v \approx \mathrm{EF}v]$, and if $\pi_2(\ell_1(y)) = [v \not\approx \mathrm{EF}v]$, then $\pi_2(\ell_1(x)) = [v \not\approx \mathrm{EF}v]$. If $x \not\equiv y$, then $\pi_2(\ell_1(x)) = [v \not\approx \mathrm{EF}v]$.
4. If y is a descendant of x and $\pi_2(\ell_1(x)) = [\neg v \not\approx \mathrm{EF}v]$ then $x \equiv y$ and $\pi_2(\ell_1(y)) = [\neg v \not\approx \mathrm{EF}v]$.

5. If y is a descendant of x and $\pi_1(\ell_1(x)) = [\neg v \approx \mathrm{EF}v]$, then $x \not\equiv y$.

6. If $\pi_2(\ell_1(x)) = [v \not\approx \mathrm{EF}v]$ and all children y of x satisfy $x \equiv y$, then there is some child z of x such that $\pi_2(\ell_1(z)) = [v \not\approx \mathrm{EF}v]$

and ℓ_2 satisfies the following conditions:

a. $\ell_2(r) = \oplus$ iff r is the root of F, $\pi_1(\ell_1(r)) = [v \approx \mathrm{EF}v]$, and there is no descendant x of r with $x \equiv r$

b. If $\ell_2(x) = \ominus$, then x is a leaf of F at maximum distance from the root (i.e. at distance d), and there is no node y at distance $< k$ from the root with $x \equiv y$.

c. If x, y are leaves with $x \equiv y$, and $\ell_2(x) = \ominus$, then $\ell_2(y) = \ominus$.

Let $\mathcal{F}_{d,k}$ be the set of all valid (d,k)-frames. We will work with many (d,k)-frames so we need a notation to distinguish the component of each of them. Unless otherwise stated, a (d,k)-frame F is a tuple $F = \langle N^F, E^F, \ell_1^F, \ell_2^F, \equiv^F \rangle$.

We say that a (d,k)-frame F' is an **extension** of a (d,k)-frame F, or that F is a **(d,k)-subframe** of F', if $N^F \subseteq N^{F'}$ and E^F, ℓ_1^F, ℓ_2^F, and $\equiv^F$ are the restrictions of $E^{F'}$, $\ell_1^{F'}$, $\ell_2^{F'}$, and $\equiv^{F'}$, respectively, to N^F.

Similarly as the definition of $T|_x$ given in Chapter 1, if T is any tree-shaped structure (in particular, a data tree, but it could also have, *e.g.* other node labeling functions), here we notate $\boldsymbol{T(x)}$ as the subtree of T generated by x and all its descendants (hence the root of $T(x)$ is x). Let F' be a (d,k)-frame, and let $x \in N^{F'}$. We name $\boldsymbol{F(x)}$ as the (d,k)-subframe of F' induced by x.

Let F be a (d,k)-frame with root r, such that $\ell_2^F(x) = \epsilon$ for all x in N^F, and let $x_1, \ldots, x_i$ be the children of r, ordered from left to right. Let G_i be the (d,k)-frames of F induced by x_i. We say that F is **1-consistent** with the (d,k)-frames $F_1, \ldots F_i$, and G_i is a (d,k)-subframe of F_i for all i. Further, we say that F is a **point of decrement** if there is a leaf $x \in N^F$ such that $\ell_2(x) = \ominus$. More precisely, we say that it is a point of decrement **of value p** if it is a point of decrement with a maximum of p $\equiv$-equivalence classes of leaves y with $\ell_2^F(y) = \ominus$. We say that F is a **point of increment** if $\ell_2^F(r) = \oplus$ and F is not a point of decrement.

The automaton $\mathcal{A}_\varphi^k$. We recall from Section 4.5 that the EX-length of φ is the maximum i such that the a term of the form $v \star \mathrm{EX}^i v$ is a subformula of φ. Let d be the EX-length of φ. We define the 1VASS$_k$ $\mathcal{A}_\varphi^k$ as follows:

- The set of states of $\mathcal{A}_\varphi^k$ consists of $\mathcal{F}_{d,k}$, the set of all valid (d,k)-frames.

- Unary rules. Let F_1 and F_2 be (d,k)-frames.

– $F_1 \xrightarrow{-n} F_2$ if F_1 is a point of decrement of value n, and F_2 is equal to F_1, except that $\ell_2^{F_2}$ is defined as follows:

$$\ell_2^{F_2}(x) = \begin{cases} \epsilon & x \text{ is a leaf of } F_1; \\ \ell_2^{F_1}(x) & \text{otherwise.} \end{cases}$$

– $F_1 \xrightarrow{+1} F_2$ if F_1 is a point of increment, and F_2 is equal to F_1, except that $\ell_2^{F_2}$ is defined as follows:

$$\ell_2^{F_2}(x) = \begin{cases} \epsilon & x \text{ is the root of } F_1; \\ \ell_2^{F_1}(x) & \text{otherwise.} \end{cases}$$

- Branching rules: $F \to (F_1, \ldots, F_i)$ (with $i \leq k$), if F is 1-consistent with $F_1, \ldots, F_i$.

We define the following sets of states of Q, to be used as inputs of the control-state reachability problem.

- Q_0 is the set of *initial* (d,k)-frames. We say that F, of root r, is initial iff the following conditions hold:
  – F, r satisfies all terms of φ of the form $v \star \mathrm{EX}^i v$ or $\neg v \star \mathrm{EX}^i v$;
  – if $v \approx \mathrm{EF}v$ [resp. $v \not\approx \mathrm{EF}v$] is a positive conjunct of φ, then the root $r \in N^F$ satisfies $\pi_1(\ell_1^F(r)) = [v \approx \mathrm{EF}v]$ [resp. $\pi_2(\ell_1^F(r)) = [v \not\approx \mathrm{EF}v]$];
  – if $v \approx \mathrm{EF}v$ [resp. $v \not\approx \mathrm{EF}v$] is a negative conjunct of φ, then for all descendants y of the root $r \in N^F$, it holds that $\pi_1(\ell_1^F(y)) = [\neg v \approx \mathrm{EF}v]$ [resp. $\pi_2(\ell_1^F(y)) = [\neg v \not\approx \mathrm{EF}v]$].
- $\widehat{Q}$ is the singleton containing the (d,k)-frame in $\mathcal{F}_{d,k}$ that consists solely of one node.

We define the **control-state reachability problem for initial sets CSReach$(\mathcal{A}, Q_0, \widehat{Q})$** as the problem of, given $\mathcal{A}, Q_0, \widehat{Q}$, whether $(q, \bar{n}) \rightsquigarrow_{\mathcal{A}} \widehat{Q}$ for some $\bar{n}$ and some $q \in Q_0$. It is easy to see that this problem is equivalent to the problem of whether $(q, \bar{0}) \rightsquigarrow_{\mathcal{A}}^{+} \widehat{Q}$ for some $q \in Q_0$. That is:

nVASS$_k$ **control-state reachability problem for initial sets**	
INPUT :	an nVASS$_k$ $\mathcal{A}$ with states Q, two subsets of states $Q_0, \widehat{Q} \subseteq Q$.
OUTPUT :	'Yes' iff $(q, \bar{0}) \rightsquigarrow_{\mathcal{A}}^{+} \widehat{Q}$ for some $q \in Q_0$.

Remark 169. *The problem of control-state reachability problem for initial sets is equivalent to the problem of control-state reachability, as defined in §4.4.3.*

Consequently, it is enough to prove:

Proposition 170. CSReach$(\mathcal{A}_\varphi^k, Q_0, \widehat{Q})$ *implies* $SAT_k(\varphi)$.

Proof. Suppose S is a solution tree for the control-state reachability problem of $\mathcal{A}^k_\varphi$. We first construct a (node-decorated) k-ranked data tree $T = \langle N, E, \ell_2, \equiv\rangle$ (with the labeling function such that $\ell_2(x) \in \{\epsilon, \ominus\}$) and then we define our solution data tree by removing the node decoration: $T' = \langle N, E, \equiv\rangle$. This last tree will satisfy φ at its root. The tree T will be constructed by induction: for every subtree R of S we construct a tree T_R such that, as we will formally verify in the second part of the proof, its structure of nodes and edges derives from the structure of the (d,k)-frames (states of $\mathcal{A}^k_\varphi$) of R, and where the semantics of the labels ℓ_1 in R is satisfied in T_R. The tree T will finally be T_S. We then verify that φ is true at the root of T_S.

Construction of the data tree. Given a subderivation R, we construct a (labeled) data tree $T_R = \langle N^R, E^R, \ell_2^R, \equiv^R\rangle$. We also identify each node x in a (d,k)-frame of R with a corresponding node $\mathrm{id}_R(x)$ in T_R. This mapping id_R will be surjective, and, whenever F is a (d,k)-frame in R, $\mathrm{id}_R \restriction N^F$ is injective.

We proceed by induction in the complexity of R.

Leaf. For the base case, let F be a leaf of S. Observe that by construction of $\mathcal{A}^k_\varphi$, $N^F = \{x\}$. Then the corresponding tree is $T_F = \langle N^F, E^F, \ell_2^F, \equiv^F\rangle$ and we define $\mathrm{id}_F(x)$ to be the same node in T_F, i.e. $\mathrm{id}_F(x) = x$.

Branching rule. Let S_0 be an incrementing derivation subtree of S such that its root, a (d,k)-frame F_0, branches into (d,k)-frames $F_1, \ldots, F_i$. We define T_{S_0} as follows. Let r be the root of F_0, and let $a_1, \ldots, a_i$ be the children of r, ordered from left to right. Let $S_j = S_0(F_j)$.

By inductive hypothesis, all S_j correspond with trees T_{S_j}, with equivalence relations $\equiv^{T_{S_j}}$ and labeling function $\ell_2^{T_{S_j}}$. Then we define T_{S_0} as follows: the root of T_{S_0} is some node $\tilde{r}$, and the subtrees hanging from its i children are T_{S_j}, for $1 \le j \le i$. See Figure 46 for an example in the case $k = 2$.

For each node x in some (d,k)-frame of S_0, we define $\mathrm{id}_{S_0}(x)$ as follows: if x is in a (d,k)-frame of some S_j, we keep the identification $\tilde{x}$ it had in T_{S_j}, i.e. $\mathrm{id}_{S_0}(x) = \mathrm{id}_{S_j}(x)$; if x is the root of F_0, namely $x = r$, then $\mathrm{id}_{S_0}(x) = \tilde{r}$; and for each node $x \neq r$ of F_0, let (by 1-consistency) x' be the corresponding copy of x in F_j and define $\mathrm{id}_{S_0}(x) = \mathrm{id}_{S_j}(x')$. It remains to define the labeling ℓ_2 and the equivalence relation $\equiv$ of T_{S_0}. We define ℓ_2 as follows

$$\ell_2(x) = \begin{cases} \epsilon & \text{if } x = \tilde{r}; \\ \ell_2^{T_{S_j}}(x) & \text{if } x \text{ is in a } (d,k)\text{-frame of } S_j. \end{cases}$$

We define $\equiv$ as the smallest equivalence relation such that:

- $\equiv \restriction_{T_{S_j}} = \equiv^{T_{S_j}}$ for each $1 \le j \le i$
- if $x \equiv^{F_0} y$ then $\mathrm{id}_{S_0}(x) \equiv \mathrm{id}_{S_0}(y)$.

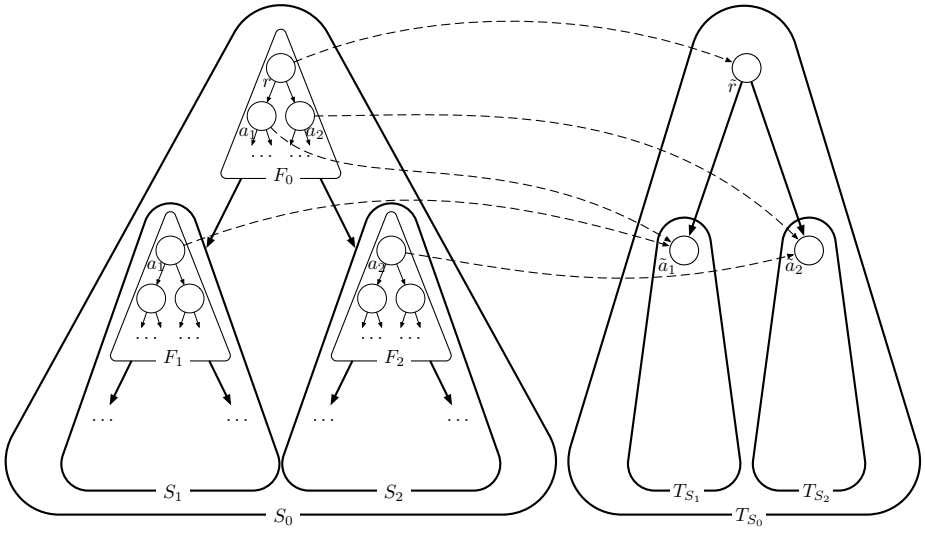


Figure 46: The incrementing derivation tree S_0, starting with a branching rule, and the constructed T_{S_0}. The dotted lines represent some pairs of the mapping id_{S_0}.

Unary rule. Let S_0 be a incrementing derivation subtree of S, with root (d,k)-frame F_0 and an only child, the (d,k)-frame F_1, as the outcome of a unary rule. Let $S_1 = S_0(F_1)$. Let $T_{S_1} = \langle N, E, \ell_2, \equiv \rangle$ be the tree that is constructed from S_1 by inductive hypothesis.

If the rule for the transition from F_0 to F_1 was a $-n$ decrement, then n is the maximal such that there are leaves $x_1, \ldots, x_n$ of F_0 with $x_i \not\equiv^1 x_j$ $(i \neq j)$, and $\ell_2^{F_0}(x_i) = \ominus$. We define $T_{S_0} = \langle N, E, \tilde{\ell}_2, \equiv \rangle$ (i.e. the same tree structure as T_{S_1}, and the same equivalence relation over it, but a different labeling function), where $\tilde{\ell}_2$ is defined below, and the identification mapping id_{S_0} is defined as follows: $\mathrm{id}_{S_0} \restriction S_1 = \mathrm{id}_{S_1}$, and for any $x \in N^{F_0}$, if x' is the corresponding copy of x in F_1 (recall that the underlying trees of F_0 and F_1 are equal), we let $\mathrm{id}_{S_0}(x) = \mathrm{id}_{S_1}(x')$. The labeling $\tilde{\ell}_2$ is defined as follows:

$$\tilde{\ell}_2(x) = \begin{cases} \ominus & \text{if } (\exists y \in N^{F_0})\ \ell_2^{F_0}(y) = \ominus \text{ and } \mathrm{id}_{S_0}(y) = x; \\ \ell_2(x) & \text{otherwise.} \end{cases}$$

In other words, $\tilde{\ell}_2$ adds $\ominus$ labels to the nodes that correspond with $\ominus$ leaves in F_0, and for all other nodes keeps the labeling of ℓ_2. See Figure 47 for an illustration of this process.

If the rule was an increment, as we will see in Lemma 171, we can assume that there is a node y in some (d,k)-frame F of S_1 such that $\ell_2^F(y) = \ominus$ and also $\ell_2(\mathrm{id}_{S_1}(y)) = \ominus$. The idea now will be to join the equivalence classes of the root of T_{S_0} with that of $\mathrm{id}_{S_0}(y) = \mathrm{id}_{S_1}(y)$, and to remove the $\ominus$ labels in the nodes of these equivalence classes. Let r be the root of F_0, and define $T_{S_0} = \langle N, E, \tilde{\ell}_2, \tilde{\equiv} \rangle$ (i.e. the same tree structure as T_{S_1}, but different equivalence relation and labeling function) where $\tilde{\ell}_2$ and $\tilde{\equiv}$ are defined below, and the identification mapping id_{S_0} is defined as in the case of a $-n$ decrement.

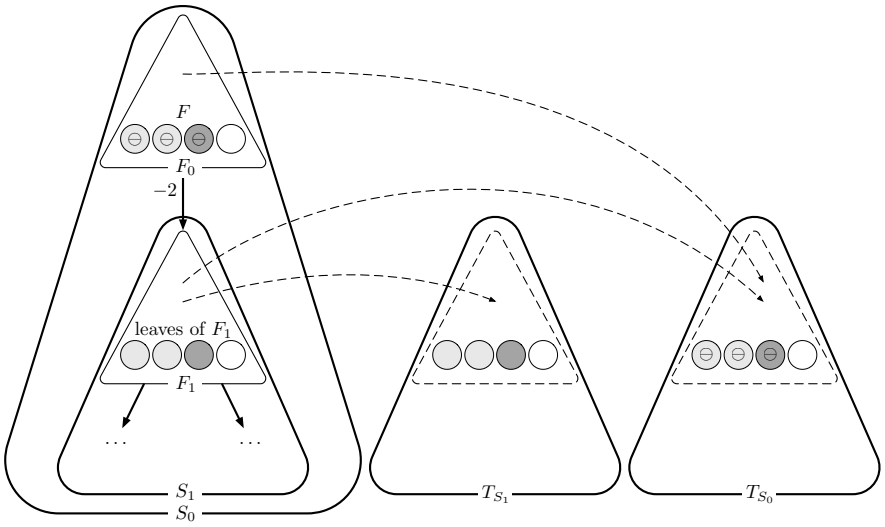


Figure 47: The incrementing derivation tree S_0, starting with a decrement -2 rule, and the constructed T_{S_0}. The colours represent the equivalence classes. The dotted lines represent the mappings id_{S_0} and id_{S_1}.

The relation $\widetilde{\equiv}$ joins the equivalence classes of $\mathrm{id}_{S_0}(r)$ and $\mathrm{id}_{S_0}(y)$, i.e.

$$z \widetilde{\equiv} w \text{ iff } z \equiv w \text{ or } (z \equiv \mathrm{id}_{S_0}(r) \text{ and } w \equiv \mathrm{id}_{S_0}(y))$$

and $\tilde{\ell}_2$ is defined as follows:

$$\tilde{\ell}_2(x) = \begin{cases} \epsilon & \text{if } x = \mathrm{id}_{S_1}(z) \text{ for some } z \equiv^F y; \\ \ell_2(x) & \text{otherwise.} \end{cases}$$

See Figure 48 for an illustration of this process.

Observe that for the construction of we are ignoring the non-deterministic increments in the derivation (which is a solution to the control-state reachability problem), as well as the exact distribution of the counters.

Lemma 171. *Let S_0 be an incrementing derivation subtree of S (a solution to the control-state reachability problem) with root (d,k)-frame F_0, such that F_0 has an only child, the (d,k)-frame F_1, which is the product of an increment rule. Let $S_1 = S_0(F_1)$. Let $T_{S_1} = \langle N, E, \ell_2, \equiv \rangle$ and $T_{S_0} = \langle N, E, \tilde{\ell_2}, \widetilde{\equiv} \rangle$ be given as in the construction. Then there is a node $p \in N$ such that $\ell_2(p) = \ominus$. Furthermore, if for any p with $\ell_2(p) = \ominus$ we define $P \subseteq N$ as*

$$P = \{x \in N \mid \ell_2(x) = \ominus \wedge x \equiv p\}$$

then if both $x, y \in P$ for some $x, y \in N$, and we have $F_x, F_y \in S_1$ such that $\tilde{x} \in F_x, \tilde{y} \in F_y$ and $\mathrm{id}_{S_1}(\tilde{x}) = x, \mathrm{id}_{S_1}(\tilde{y}) = y$ (and thus with $\ell_2^{F_x}(\tilde{x}) = \ominus$ and $\ell_2^{F_y}(\tilde{y}) = \ominus$), then we have that $F_x = F_y$ and $\tilde{x} \equiv_{F_x} \tilde{y}$.

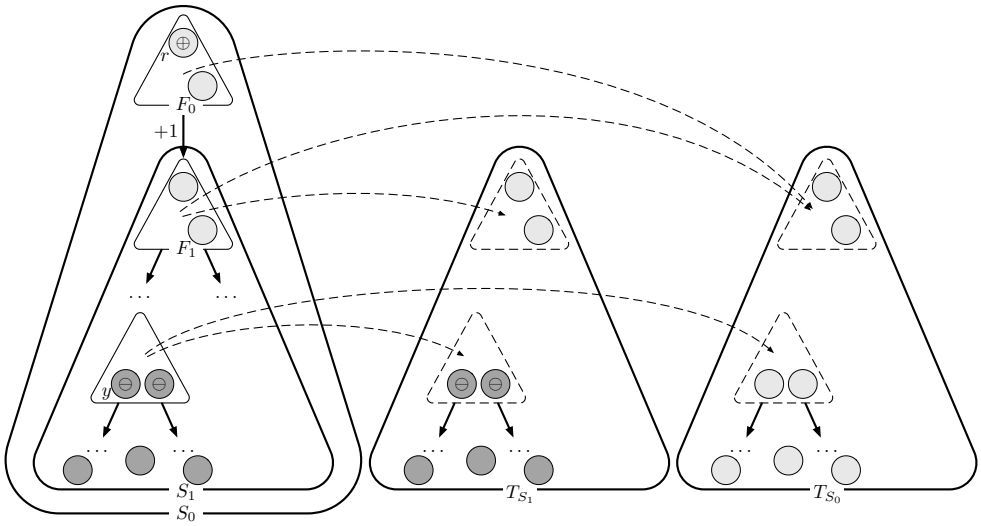


Figure 48: The incrementing derivation tree S_0, starting with an increment rule, and the constructed T_{S_0}. The colours represent the equivalence classes. The dotted lines represent the mappings id_{S_0} and id_{S_1}.

That is, the $\ominus$-labeled nodes of T_{S_1} that in T_{S_0} are ϵ-labeled correspond to the equivalence class of $\ominus$-labeled leaves of a single (d,k)-frame in S_0. In other words, when the join of classes is done in the construction of T_{S_0} because of an increment operation, the nodes y in T_{S_0} with $\ell_2^{T_{S_1}}(y) = \ominus$ that have $\ell_2^{T_{S_0}}(y) = \epsilon$ cannot correspond via $\mathrm{id}_{S_0}^{-1}$ to $\ominus$-labeled leaves in more than one (d,k)-frame; they correspond with exactly one decrement of the counter in the incrementing derivation tree.

Proof of Lemma 171. We will prove the lemma by induction on S_1.

Base case. If S_1 is such that there is no increment operation in S_1 then, as S_1 is an incrementing derivation subtree of a solution of the control-state reachability problem, S_1 must contain a point of decrement F with a node $\tilde{y}$ with $\ell_2^F(\tilde{y}) = \ominus$ and, since there is no increment operation in S_1, we have that $\ell_2(\mathrm{id}_{S_1})(\tilde{y}) = \ominus$ (that is, $p = \mathrm{id}_{S_1}(\tilde{y})$).

For the second claim, assume that $p \in N$ is any node with $\ell_2(p) = \ominus$, and P is defined accordingly. Suppose by way of contradiction that $x, y \in N$ are such that $x, y \in P$, but there is no single $F \in S_1$ such that there are $\tilde{x}, \tilde{y} \in F$ with $\mathrm{id}_{S_1}(\tilde{x}) = x, \mathrm{id}_{S_1}(\tilde{y}) = y$, $\ell_2^F(\tilde{x}) = \ominus, \ell_2^F(\tilde{y}) = \ominus$, and $\tilde{x} \equiv_F \tilde{y}$. We will prove this leads to $x \not\equiv y$, a contradiction with our assumption that $x, y \in P$, and consequently we will conclude that there is such a (d,k)-frame F.

First, observe that there cannot be a (d,k)-frame F_x and $\tilde{x} \in F_x$ with $\mathrm{id}_{S_1}(\tilde{x}) = x$ and $\ell_2^{F_x}(\tilde{x}) = \ominus$ such that $\tilde{x}$ has an ancestor $\tilde{a} \in F_x$ with $\tilde{a} \equiv_{F_x} \tilde{x}$, since leaves of (d,k)-frames are not labeled with $\ominus$ if they have an in-frame ancestor. Now, since increments have not been used in the construction of T_{S_1}, the equivalence classes of (d,k)-frames of S_1 coincide with their identification via id_{S_1} in T_{S_1}, and therefore there cannot be an ancestor

z of x at distance d or less from x with $z \equiv x$. The analogous result holds for y. Now, $x \equiv y$, and $\ell_2(x) = \ell_2(y) = \ominus$ implies that there are $F_x, F_y \in S_1$, $\tilde{x} \in F_x, \tilde{y} \in F_y$ with $\mathrm{id}_{S_1}(\tilde{x}) = x, \mathrm{id}_{S_1}(\tilde{y}) = y$ and with $\ell_2^{F_x}(\tilde{x}) = \ominus, \ell_2^{F_y}(\tilde{y}) = \ominus$. Observe from the definition of the decrement rule in $\mathcal{A}^k_\varphi$ and the conditions on $\ominus$, that there cannot be $\tilde{w} \in F_w$ with $\mathrm{id}_{S_1}(\tilde{w}) = x$ and $F_w \neq F_x$. Analogously for y. So, as we are assuming $F_x \neq F_y$, from 1-consistency and the increment-free construction of T_{S_1}, $x \equiv y$ implies that there is a least common ancestor z in N with $x \equiv z \equiv y$, and a chain of nodes $z = z_0, \ldots, z_n = x$ in N such that z_{i+1} is a descendant of z_i at distance at most d, and $z_i \equiv z_{i+1}$ (and the same for a chain towards y). But this contradicts the observation that there cannot be an ancestor z of x at distance d or less with $x \equiv z$.

Induction. Observe that, since S is a solution to the control-state reachability problem, for every increment in S_0 there must be at least one decrement in S_1. Suppose there are m increments and $n \geq m$ decrements in S_0. Then there are $m-1$ increments and $n > m-1$ decrements in S_1. From the inductive hypothesis, there must remain at least one node $p \in N$ with $\ell_2(p) = \ominus$.

For the second claim, let $p \in N$ with $\ell_2(z) = \ominus$, $P = \{x \in N \mid \ell_2(x) = \ominus \wedge x \equiv p\}$, and $x, y \in P$. Let $x, y \in P$, and let $F_x, F_y \in S_1$, $\tilde{x} \in F_x, \tilde{y} \in F_y$ be such that $\mathrm{id}_{S_1}(\tilde{x}) = x, \mathrm{id}_{S_1}(\tilde{y}) = y$ We want to prove that $F_x = F_y$ and $\tilde{x} \equiv_{F_x} \tilde{y}$.

As $\ell_2(x) = \ell_2(y) = \ominus$, x and y cannot have been joined with the equivalence class of an ancestor node of T_{S_1} as a result of an increment operation. Therefore, we are in a similar case as in the base step: there must a common ancestor z of x, y, such that $x \equiv z \equiv y$ and a chain of nodes $z = z_0, \ldots, z_n = x$ in N such that z_{i+1} is a descendant of z_i at distance at most d, and $z_i \equiv z_{i+1}$, but this is a contradiction with the fact that $\ell_2(x) = \ominus$. □

Verification. By ignoring the labeling function ℓ_2 of $T_S = \langle N, E, \equiv, \ell_2 \rangle$ we obtain the desired data tree. We show that the data tree $T = \langle N, E, \equiv \rangle$ satisfies φ at the root. To ease the notation we write id for id_S.

From the definition of Q_0 and the construction of T, it is clear that any conjunct of φ of the form $v \star \mathrm{EX}^i v$ or $\neg v \star \mathrm{EX}^i v$, with $\star \in \{\approx, \not\approx\}$, is satisfied in the root of T.

Next, we show that conjuncts of φ of the type $v \star \mathrm{EF} v$ or $\neg v \star \mathrm{EF} v$ are also satisfied in the root of T. Recall from the construction of T that all nodes of N correspond via id^{-1} to nodes $x_1, \ldots, x_n$ in (d,k)-frames $F_1, \ldots, F_n$ of S such that $\ell_1^{F_i}(x_i) = \ell_1^{F_1}(x_1)$ for all $i = 1 \ldots n$. Therefore, it is enough to verify that if x is a node in some (d,k)-frame F of S, then $\mathrm{id}(x)$ satisfies the semantics of $\pi_1(\ell_1^F(x))$ and of $\pi_2(\ell_1^F(x))$. Furthermore, it is enough to verify the above when x is the root of F. In what follows, we write ℓ_j instead of ℓ_j^F $(j = 1, 2)$.

We will use the following facts:

Fact 172. *In the construction of T, nodes that are at distance of at most d and non-equivalent in some frame are never made equivalent in T.*

Fact 173. *In the construction of T, nodes that are equivalent for some tree $T_{S'}$ with $S' \subseteq S$ are kept equivalent for T_S.*

Fact 174. *If $x \not\equiv^F z$ for all $x \neq z \in F$, and $\ell_2(x) \neq \oplus$ (recall that x is the root of F), then in the construction of T the equivalence class of* $\mathrm{id}(x)$ *is never joined with the equivalence class of a descendant.*

We consider the four different cases, two for every projection π_1 or π_2 of $\ell_1(x)$:

- In the case $\pi_1(\ell_1(x)) = [v \approx \mathrm{EF}v]$, we show the formula $v \approx \mathrm{EF}v$ is satisfied in $\mathrm{id}(x)$:
  - *Local satisfaction.* If there is some descendant $y \in F$ of x such that $x \equiv^F y$, then, from Fact 173, $\mathrm{id}(x) \equiv^T \mathrm{id}(y)$ and thus $T, \mathrm{id}(x) \models v \approx \mathrm{EF}v$.
  - *Non-local satisfaction.* If there is no such frame as in the previous sub-item, then (since $\pi_1(\ell_1(x)) = [v \approx \mathrm{EF}v]$) there is a frame F with $\ell_2^F(x) = \oplus$. Thus, from the construction of T and Fact 173, there is a descendant of $\mathrm{id}(x)$ in its equivalence class.
- In the case $\pi_1(\ell_1(x)) = [\neg v \approx \mathrm{EF}v]$, we consider two subcases to consider. If $\pi_2(\ell_1(x)) = [\neg v \not\approx \mathrm{EF}v]$, from the validity condition 1 the nodes $\mathrm{id}^{-1}(\mathrm{id}(x))$ have no descendants (and thus neither does $\mathrm{id}(x)$), and therefore $\neg v \approx \mathrm{EF}v$ is trivially satisfied at $\mathrm{id}(x)$. Otherwise, if $\pi_2(\ell_1(x)) = [v \not\approx \mathrm{EF}v]$, we show that for all descendants $\mathrm{id}(y)$ of $\mathrm{id}(x)$ at distance k of $\mathrm{id}(x)$ in T we have $\mathrm{id}(x) \not\equiv \mathrm{id}(y)$:
  - *Local satisfaction.* If $0 < k \leq d$ then taking $\tilde{y}$ in the frame F such that $\mathrm{id}(\tilde{y}) = \mathrm{id}(y)$, from condition 5, $x \not\equiv \tilde{y}$ and then from Fact 172 we have $\mathrm{id}(x) \not\equiv \mathrm{id}(y)$.
  - *Non-local satisfaction.* If $k > d$, since $\pi_1(\ell_1(x)) = [\neg v \approx \mathrm{EF}v]$ and x is the root of F, then $\ell_2(x) \neq \oplus$. As we have seen that $x \not\equiv^F z$ for all $x \neq z \in F$, Fact 174 indicates that for any $\tilde{y}$ descendant of $\mathrm{id}(x)$ we have $\mathrm{id}(x) \not\equiv \tilde{y}$.
- In the case $\pi_2(\ell_1(x)) = [v \not\approx \mathrm{EF}v]$, we distinguish two cases:
  - *Local satisfaction.* If there is a descendant $y \in F$ of x (and thus at distance at most d from x) such that $x \not\equiv^F y$, then, from construction of T, there is also a descendant $\mathrm{id}(y)$ in N such that $\mathrm{id}(x) \not\equiv^{T_{S_F}} \mathrm{id}(y)$. Thus, from Fact 172, $\mathrm{id}(x)$ satisfies $v \not\approx \mathrm{EF}v$.
  - *Non-local satisfaction.* If there is no descendant $y \in F$ with $x \not\equiv^F y$, then, from condition 6 there is chain of descendants of x, $y_1 \in F_1, \ldots, y_n \in F_n$ such that $\forall i < n\, y_i \equiv y_{i+1}$, and $\forall i\, \pi_2(\ell_1(y_i)) = [v \not\approx \mathrm{EF}v]$, and such that y_n has no child z with $z \equiv y_n$. As $\pi_2(\ell_1(y_n)) = [v \not\approx \mathrm{EF}v]$, condition 2 implies that there must be a child z of y_n, and then necessarily $z \not\equiv y_n$. From Fact 172 and Fact 173, therefore $\mathrm{id}(x) \not\equiv \mathrm{id}(z)$ and then $v \not\approx \mathrm{EF}v$ is satisfied in $\mathrm{id}(x)$.
- Finally, in the case $\pi_2(\ell_1(x)) = [\neg v \not\approx \mathrm{EF}v]$, the formula $\neg v \not\approx \mathrm{EF}v$ is satisfied in $\mathrm{id}(x)$ since, by validity condition 4 and construction of T, for all of the descendants $\mathrm{id}(y)$ of $\mathrm{id}(x)$ we have that $\mathrm{id}(x) \equiv \mathrm{id}(y)$ and $\ell_1(y) = [\neg v \not\approx \mathrm{EF}v]$.

This concludes the proof of Proposition 170. □

Proposition 175. *$SAT_k(\varphi)$ implies* $\text{CSReach}(\mathcal{A}^k_\varphi, Q_0, \widehat{Q})$.

Proof. Let T be a k-ranked finite data tree whose root satisfies φ. We want to see that there is an incrementing derivation tree S of $\mathcal{A}^k_\varphi$ that starts at a node of Q_0 with the counter at 0 and ends with all leaves in $\widehat{Q}$ with the counter in 0.

We construct the incremental derivation tree from the root to the leaves. The idea is simply to identify which states ((d,k)-frames) of the automaton $\mathcal{A}^k_\varphi$ correspond to portions of T, adding first the appropriate values of ℓ_1 and ℓ_2, and then performing unary and binary operations in the expected way.

From a node in T to a (d,k)-frame For any node x of T we associate a (d,k)-frame F_x, defined as follows:

- N^{F_x} is the maximal subtree of T height d that hangs from x
- $\equiv^{F_x} = \equiv\restriction_{N^{F_x}}$
- $\ell_1^{F_x}(y)$ is consistent with T (*i.e.* $\ell_1^{F_x}(y) = ([v \approx \mathrm{EF}v], [v \not\approx \mathrm{EF}v])$ if y has a descendant in F_x with the same data value and other with different data value, etc.)
- $\ell_2^{F_x}$ defined as follows (*cf.* items a and b of the conditions for the (d,k)-frames):

$$\ell_2(y) = \begin{cases} \oplus & \text{if } y = x,\ \pi_1(\ell_1(y)) = [v \approx \mathrm{EF}v], \text{ and} \\ & \text{there is no descendant } z \text{ of } y \text{ with } z \equiv y; \\ \ominus & \text{if } y \text{ is a descendant of } x \text{ at distance } d \text{ from it, } x \not\equiv y, \text{ and} \\ & \text{for all descendant } z \text{ of } x \text{ at distance } < d \text{ we have } y \not\equiv z; \\ \epsilon & \text{otherwise.} \end{cases}$$

Let r be the root of T. We select as the initial state of S the (d,k)-frame F_r.

Construction of the incrementing derivation tree. For this step, to make clearer whether we are referring to a node in T or in some (d,k)-frame, we will use a function id from the roots of the (d,k)-frames into T, as in Proposition 170. If $\tilde{x}$ is the root of F, and $F = F_x$, then we set $\mathrm{id}(\tilde{x}) = x$.

We now decide which rules are invoked on each (d,k)-frame F_x we construct. Let $\tilde{x} \in \mathrm{id}^{-1}(x)$ be the root of F.

If F_x is neither a point of decrement nor a point of increment, and if x has descendants, then we use a binary rule to branch F_x into the two (d,k)-frames F_{a_1} and F_{a_2}, where a_1, a_2 are the two children of x in T. The particular way in which the counter of $\mathcal{A}^k_\varphi$ has been divided between these two (d,k)-frames will be explained afterwards (see **Verification**, below).

If F_x is a point of decrement of value n then from F_x we transition, via an n-decrement, to the (d,k)-frame $F^- = \langle N^{F_x}, E^{F_x}, \ell_1^{F_x}, \ell_2^-, \equiv^{F_x}\rangle$ (*i.e.* F^- is equal to F_x except for the ℓ_2 labeling function), where

$$\ell_2^-(\tilde{y}) = \begin{cases} \epsilon & \text{if } \ell_2(\tilde{y}) = \ominus; \\ \ell_2^{F_x}(\tilde{y}) & \text{otherwise.} \end{cases}$$

If $\tilde{x}^-$ is the root of F^-, we keep $\mathrm{id}(\tilde{x}^-) = \mathrm{id}(\tilde{x})$.

If F_x is a point of increment, we transition to the (d,k)-frame $F^+ = \langle N^{F_x}, E^{F_x}, \ell_1^{F_x}, \ell_2^+, \equiv^{F_x}\rangle$ (*i.e.* F^- is equal to F_x except for the ℓ_2 labeling function), where

$$\ell_2^+(\tilde{y}) = \begin{cases} \epsilon & \text{if } \ell_2(\tilde{y}) = \oplus; \\ \ell_2(\tilde{y}) & \text{otherwise.} \end{cases}$$

If $\tilde{x}^+$ is the root of F^+, we keep $\mathrm{id}(\tilde{x}^+) = \mathrm{id}(\tilde{x})$.

Verification. We show that the constructed incrementing derivation tree S is a solution to the control-state reachability problem. Observe that labels $\oplus$ of ℓ_2 are assigned to the roots of (d,k)-frames when the corresponding node of T has a descendant at distance greater than d with the same equivalence class (but it has none at distance d or less). Also, $\ominus$ labels are assigned to leaves when they do not have an ancestor of the same class at distance less than d from the root of the frame. Thus, for every frame in S whose root r has $\ell_2(r) = \oplus$, there is at least one descendant frame in S with a leaf y with $\ell_2(y) = \ominus$.

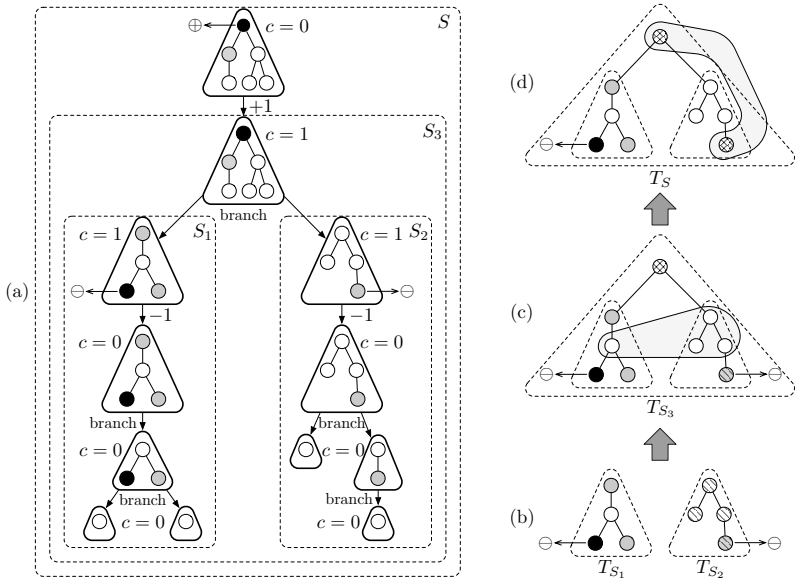
Therefore, in the incrementing derivation tree S the total number of increments of the counter has been some number m and the total value of the decrements some $n \geq m$. If $n > m$, we modify S to add an spontaneous increase of the counter in $n - m$ at the first frame, which can be done since we are working on the control-state reachability problem. So we can assume that $m = n$. It remains to assign, for each instance of the branching rule, a way in which the counter has been divided between the branches. We do that as follows: if at a node F of S the counter is at c and the next operation of S is a branching $F \to F_1 \mid \cdots \mid F_i$, then for each i we assign c_i to F_i, where c_i is the number of decrements in the derivation tree of $S(F_i)$. In this way, all the operations of S are valid (there are no n-decrements when the counter is less than n), and the counter in all leaves ends up at 0. Therefore S is a solution for the control-state reachability problem.

This concludes the proof of Proposition 175. □

Finally, from Propositions 170 and 175 we obtain Proposition 168.

Example 176. The following figure illustrates a scheme of an incrementing derivation S of the 1VASS$_2$ $\mathcal{A}^2_\varphi$ (a) and some steps (b, c and d) in the bottom up construction of the data tree T_S satisfying φ, for $\varphi = \neg v \approx \mathrm{EX}v \wedge \neg v \approx \mathrm{EX}^2 v \wedge v \approx \mathrm{EF}v$. Triangles represent $(2,2)$-frames. Shades of gray represent the equivalence classes, which only make sense inside any frame. The counter is notated with c, and arrows represent the (unary/branching) transitions of the derivation. Notice that the top branching is 'incremental', and that the

local requirements of φ (namely, $\neg v \approx \mathrm{EX}v$ and $\neg v \approx \mathrm{EX}^2 v$) are satisfied in the root of the top frame.



The construction of T_S is bottom-up, and we show three steps: (a), (b) and (c). Notice that in (b) each of T_{S_1} and T_{S_2} has its own partition (no intersection). In (c) we process the root of S_3 by tying together T_{S_1} and T_{S_2} with a common parent, who lives in a single class of the partition. Notice that the partitions of T_{S_1} and T_{S_2} are properly joined (grey area), according to the information in the root of S_3. Finally in (d) we construct T_S. The root of S is a point of increment, so we match $\oplus$ with some $\ominus$ in T_{S_3}. In this case, we match it with the right-hand $\ominus$, and so we join them by putting them in the same partition (gray area). We have satisfied the future requirement $v \approx \mathrm{EF}v$ of φ.

Remark 177. *There exists a similar reduction when considering the logic that allows node-labeling and formulas of the type $\psi := a$. In this case, the construction of $\mathcal{A}^k_\varphi$ can be easily extended so as to prove the analog of Proposition 170 in the presence of these type of formulas.*

4.5.2 Adding Boolean and *Until* operators: $\mathbf{LRV}^{\mathbf{D}}_1$

We extend $\mathrm{LRV}^{\mathrm{D}-}_1$ with $\wedge$, $\vee$, $\neg$ and EU. We will combine the idea of [34, Section 3.2] with our previous approach of §4.5.1 in order to deal with this more expressive logic, to obtain the following generalization of Proposition 168:

Proposition 178. *There is a reduction from SAT_k-$\mathrm{LRV}^{\mathrm{D}}_1$ to* $\mathrm{CSReach}(1\mathrm{VASS}_k)$.

Let φ be a formula. We define $cl(\varphi)$ to be the standard *closure* of φ: the smallest F set of formulas that contains φ, is closed under subformulas, and satisfies the following conditions:

- If $\psi \in F$ and ψ is not of the form $\neg\psi_1$ for some ψ_1, then $\neg\psi \in F$.
- If $\mathrm{EU}(\psi_2, \psi_1) \in F$ then $\mathrm{EX}(\mathrm{EU}(\psi_2, \psi_1)) \in F$.

An atom of φ is a subset A of $cl(\varphi)$ which is maximally consistent in that it satisfies the following conditions:

- For every $\neg\psi \in cl(\varphi)$, we have $\neg\psi \in A$ iff $\psi \notin A$.
- For every $\psi_1 \wedge \psi_2 \in cl(\varphi)$, we have $\psi_1 \wedge \psi_2 \in A$ iff ψ_1 and ψ_2 are in A.
- For every $\psi_1 \vee \psi_2 \in cl(\varphi)$, we have $\psi_1 \vee \psi_2 \in A$ iff ψ_1 or ψ_2 is in A.
- For every $\mathrm{EU}(\psi_2, \psi_1) \in cl(\varphi)$, we have $\mathrm{EU}(\psi_2, \psi_1) \in A$ iff either $\mathrm{EX}(\psi_2) \in A$ or both $X(\psi_1) \in A$ and $\mathrm{EX}(\mathrm{EU}(\psi_2, \psi_1)) \in A$.

We denote by $Atom(\varphi)$ the set of atoms of φ. Let $A, A_1, \ldots, A_i \in Atom(\varphi)$. We say that A is **1-consistent** with $A_1, \ldots, A_i$ if for every $\mathrm{EX}(\psi) \in cl(\varphi)$, we have $\mathrm{EX}(\psi) \in A$ iff $\psi \in \bigcup_{1 \leq j \leq i} A_j$.

We recall the definition of EX-length given in §4.5.

As in §4.5.1, we show that for any formula φ of $\mathrm{LRV}^{\mathrm{D}}_1$, there is a $1\mathrm{VASS}_k$ $\mathcal{B}^k_\varphi$ and sets $Q_0, \widehat{Q}$ such that $\mathrm{SAT}_k(\varphi)$ iff $\textsc{CSReach}(\mathcal{B}^k_\varphi, Q_0, \widehat{Q})$.

The idea behind the definition of $\mathcal{B}^k_\varphi$ is similar to the one of $\mathcal{A}^k_\varphi$ defined in §4.5.1. However, $\mathcal{B}^k_\varphi$ will encode more information. For the definition of $\mathcal{B}^k_\varphi$, recall the definition of (d,k)-frame given in §4.5.1. We say that an atom A is **locally consistent** with a (d,k)-frame F if, for r the root of F, and for all $\gamma \in A$ of the form $v \star \mathrm{EF}v$ or $\neg v \star \mathrm{EF}v$ (for $\star \in \{\approx, \not\approx\}$), r is labeled appropriately in ℓ_1.

The automaton $\mathcal{B}^k_\varphi$. We define the $1\mathrm{VASS}_k$ $\mathcal{B}^k_\varphi$ as follows:

- The set of states Q of $\mathcal{B}_\varphi$ consists of the set
$$\{(F, A) \in \mathcal{F}_{d,k} \times Atom(\varphi) \mid A \text{ is locally consistent with } F\}.$$
- Unary rules. Let F_1 and F_2 be (d,k)-frames, and let A be an atom.
  - $(F_1, A) \xrightarrow{-n} (F_2, A)$ if F_1 and F_2 are as in the rule $F_1 \xrightarrow{-n} F_2$ of $\mathcal{A}^k_\varphi$.
  - $(F_1, A) \xrightarrow{+1} (F_2, A)$ if F_1 and F_2 are as in the rule $F_1 \xrightarrow{+1} F_2$ of $\mathcal{A}^k_\varphi$.
- Branching rules: $(F, A) \to ((F_1, A_1), \ldots, (F_i, A_i))$ (with $i \leq k$), if F is 1-consistent with $F_1, \ldots, F_i$, and A is 1-consistent with $A_1, \ldots, A_i$.

We now define the two sets Q_0 and $\widehat{Q}$, to be used as inputs of the control-state reachability problem.

- Q_0 is the set of all $(F, A) \in \mathcal{F}_{d,k} \times Atom(\varphi)$ which are *initial*. We say that (F, A) is initial iff F satisfies the same conditions of an initial state of $\mathcal{A}^k_\varphi$ and $\varphi \in A$.

- $\widehat{Q}$ is the set of all the states of the form (F, A), where F is the only (d, k)-frame that consists solely of one node, and where A is such that if $\mathrm{EU}(\rho, \psi) \in cl(\varphi)$ then $\mathrm{EU}(\rho, \psi) \notin A$.

4.5.3 The general case: $\mathbf{LRV}^{\mathbf{D}}_n$

To prove Theorem 167 it is enough to introduce a small modification to the construction of $\mathcal{F}_{d,k}$ as seen in Proposition 168, changing the dimension (polynomially in n) of the codomains of ℓ_1 and of ℓ_2 in order to maintain information for all the variables. For the case of n disjoint variables and $\varphi \in \mathrm{LRV}^{\mathrm{D}}_n$ we make the following changes to the definition of the (d, k)-frames that will constitute the states of the automaton $\mathcal{B}^k_\varphi$:

- Let $V_i = \{([v_i \approx \mathrm{EF}v_i], [v_i \not\approx \mathrm{EF}v_i]), ([\neg v_i \approx \mathrm{EF}v_i], [v_i \not\approx \mathrm{EF}v_i]), ([v_i \approx \mathrm{EF}v_i], [\neg v_i \not\approx \mathrm{EF}v_i]), ([\neg v_i \approx \mathrm{EF}v_i], [\neg v_i \not\approx \mathrm{EF}v_i])\}$. Now we have $\ell_1 : N \to \prod_{1 \leq i \leq n} V_i$.

- $\ell_2 : N \to \prod_{1 \leq i \leq n} \{\epsilon_{v_i}, \oplus_{v_i}, \ominus_{v_i}\}$.

- $\equiv_1, \ldots, \equiv_n$ are equivalence relations over the nodes of the frames.

The notions of validity and 1-consistency between frames are then adjusted accordingly. There are now n counters in the automaton, and n instances of our previous unary rules for $\mathcal{B}^k_\varphi$, one for every disjoint variable.

4.5.4 Complexity

We will make a short analysis on the complexity of the reduction corresponding to Theorem 167. For this part, we will assume we are working without labels, but their addition to the logic does not change the complexity classes in our results.

First, we rapidly note that the maximum size of an entry in a unary rule of the n-counter automaton $\mathcal{B}^k_\varphi$ is bounded by k^d, corresponding with a (d, k)-frame of maximum depth where all leaves are labeled via ℓ_2 with $\ominus$ and all leaves belong to different equivalence classes; such (d, k)-frame is a point of decrement of value k^d, and there cannot be points of decrement of higher value.

By making an analysis on the size of $\mathcal{F}_{d,k}$ (the set of valid (d, k)-frames) and of $Atom(\varphi)$ (which is used for the temporal portion), we can obtain an upper bound on the size of the automaton $\mathcal{B}^k_\varphi$ of our reduction in Subsection 4.5.2, generalized to $\mathrm{LRV}^{\mathrm{D}}_n$. Each state of $\mathcal{B}^k_\varphi$ is basically a k-ranked tree of depth at most d (equipped with a node-labeling function and an equivalence class) and a set of (roughly) subformulas of φ. There are $O(2^{p_1(|\varphi|)})$ such sets, for some polynomial p_1, where $|\varphi|$ denotes the number of subformulas of φ. On the other hand, there are $O(k^{d+1})$ many k-ranked tree of depth at most d, and so there are $O((k^{d+1})^{k^{d+1}})$ many such trees with a binary relation defined on its nodes. Further,

each leaf can be labeled with $p_2(n)$-many labels, where $p_2(n)$ is the product of the size of the codomains of the functions ℓ_1 and ℓ_2 for (d,k)-frames in the n-dimensional case, as in Subsection 4.5.3. So there are $O((k^{d+1})^{k^{d+1}} \cdot p_2(n)^{k^{d+1}})$ many k-ranked trees of depth at most d equipped with node-labeling functions and an equivalence class.

Let $\mathrm{LRV}^{\mathrm{D}}_{n,d}$ be the fragment of $\mathrm{LRV}^{\mathrm{D}}_{n}$ where each formula has EX-length at most d. We have obtained:

Proposition 179. *Given $\varphi \in \mathrm{LRV}^{\mathrm{D}}_{n,d}$, the number of states of $\mathcal{B}^k_\varphi$ is*

$$O(p(n)^{k^{d+1}} \cdot (k^{d+1})^{k^{d+1}} \cdot 2^{p(|\varphi|)})$$

for some polynomial p.

Now, using Proposition 163, Theorem 167, and Proposition 179, we can obtain:

Proposition 180. *SAT_k-$\mathrm{LRV}^{\mathrm{D}}_{n,d}$ is in* ExpTime *for fixed k,n,d; it is in* 2ExpTime *for fixed k,n or fixed d,k; and it is in* 3ExpTime *for fixed k.*

Proof. Observe that the reduction of Theorem 167 can be done using only exponential space, as we can codify in exponential space the lists of states and the rules between states, while checking if each state represents a valid (d,k)-frame and if each rule corresponds to one of our unary or branching rules.

In order to use the Prop. 163 from [37], we translate our $n\mathrm{VASS}_k$ $\mathcal{B}^k_\varphi$ from Theorem 167 into a BVAS B such the control-state reachability problem of our $n\mathrm{VASS}_k$ is equivalent to the coverability problem (with $\bar{n} = 0$ in our reduction) for this B. We will find a $B = \langle n + \tilde{n}, R_1, R_2\rangle$, where $R_1, R_2 \subseteq \mathbb{Z}^{n+\tilde{n}}$ are unary and binary rules, respectively. Observe that we need to measure the maximum binary size of entries in the rules R_1, R_2, and we also need to specify a set of axioms, which adds a linear size to the input.

First we will show how we can build a branching VASS $\mathcal{C} = \langle C, U_C, B_C\rangle$ with a constant number of states; a fixed increase $\tilde{n}$ in the dimension; and a new bound (that dominates the bound of $\log_2(k^d)$ for $\mathcal{B}^k_\varphi$) to the binary size of the maximum entry of the rules, a bound that is logarithmic on $|\mathcal{B}^k_\varphi|$. Afterwards, we can translate $\mathcal{C}$ to a BVAS in a standard way, which does not increase our complexities, and only needs a single axiom.

Let $q_0, \ldots, q_N$ the states of $\mathcal{B}^k_\varphi$. $\mathcal{C}$ will have three states: $C = q_a, q_b, q_c$. For each unary rule on $\mathcal{B}^k_\varphi$ of the form $q_i \xrightarrow{\bar{v}} q_j$, U_c contains the rules:

- $q_a \xrightarrow{(\bar{v},\bar{w})} q_b$, where $\bar{w} = (-i, -(N-i), j, N-j, 0, 0)$.
- $q_b \xrightarrow{(\bar{v},\bar{w})} q_a$, where $\bar{w} = (j, N-j, -i, -(N-i), 0, 0)$.
- $q_a \xrightarrow{(\bar{v},\bar{w})} q_c$, where $\bar{w} = (-i, -(N-i), 0, 0, j, N-j)$.
- ...And so on for all combinations $q_i \to q_j$ with $i,j \in \{a,b,c\}$.

For branching rules on $\mathcal{B}^k_\varphi$ of the form $q_k \to (q_i, q_j)$, B_C contains the rules[22]:

- $q_a, q_b \xrightarrow{\bar{w}} q_c$, where $\bar{w} = (-i, -(N-i), j, -(N-j), k, N-k)$.
- And so on for all combinations with the three states.

For branching rules of higher branching, the idea is similar, but we have to introduce new counters in order to simulate k-branching with only binary branching. □

4.6 Obtaining equivalence with VASS_k

In the previous section we have seen a reduction into the control-state reachability problem for VASS_k. A natural question is whether there exists a reduction in the other direction: can $\text{CSReach}(\text{VASS}_k)$ be reduced into the k-satisfiability for LRV^{D}? For the case $k = 1$, this has been shown to be the case [36]: there exists a polynomial-space reduction from $\text{CSReach}(\text{VASS}_1)$ to $\text{SAT}_1(\text{LRV})$.

The existence of a reduction would show, intuitively, that one can express in the logic that there is a tree that verifies all the conditions for being a derivation. Without the use of data tests, one can easily encode trees that verify all the conditions except perhaps (25) and (26) regarding the vectors. For this, let us assume without loss of generality that all unary rules contain a vector $\bar{e}_i$ or $-\bar{e}_i$. The data values are used to ensure the next two conditions:

- Along any branch, every node containing a rule of the form $q \xrightarrow{\bar{e}_i} q'$ has a unique data value. In other words, we cannot find two nodes encoding an increment of component i with the same data value so that one is the ancestor of the other.
- For every node with a unary rule $q \xrightarrow{\bar{e}_i} q'$ there exists a descendant with a rule $p \xrightarrow{-\bar{e}_i} p'$ and the same data value.

These two conditions imply that after incrementing component i there must be *at least* one corresponding decrement of component i. Note that there could be *more* decrements than increments, which is not a problem since we work under the 'incrementing' semantics.

Interestingly, these two conditions can be expressed in LRV, but we do not know how to encode it in LRV^{D} (we conjecture that they are not expressible).

Adding the operator $\text{AG}_{\approx v}(\varphi)$ We add a new operator $\text{AG}_{\approx v}(\varphi)$ to LRV^{D}, where $T, x \models \text{AG}_{\approx v}(\varphi)$ if every descendant of x with the same v-attribute verifies φ. The fragment of LRV^{D}_n extended with positive occurrences of $\text{AG}_{\approx v}(\varphi)$ (that is, where $\text{AG}_{\approx}$ occurs always under an even number of negations) is called $\text{LRV}^{\text{D}}_n(\text{AG}^+_{\approx})$.

Now, in $\text{LRV}^{\text{D}}_n(\text{AG}^+_{\approx})$ one can express: *for every node x containing a rule $q \xrightarrow{\bar{e}_i} q'$, we have that all descendants of x with the same v_i attribute contain a rule of the form $p \xrightarrow{-\bar{e}_i} p'$.*

[22] The branching rules B_C of $\mathcal{C}$ will allow the addition of a vector, in order to facilitate a later transition to a BVAS; this feature is unessential, but useful for the purpose of clarity.

This, added to the property that every increment for component i must verify $v_i \approx \mathrm{EF} v_i$, ensures that the tree indeed encodes a derivation tree.

Theorem 181. $\mathrm{CSReach}(n\mathrm{VASS}_k)$ *is* PTime*-reducible to* SAT_k*-*$\mathrm{LRV}_1^{\mathrm{D}}(\mathrm{AG}_{\approx}^{+})$*, where the number of labels in the logic depends (polynomially) on the size of the* $n\mathrm{VASS}_k$*.*

Proof. We will start by proving the result for $n = 1$, as the general case follows from small modifications to the proofs for this restricted case. Let $\mathcal{C}^k = \langle Q, U, B\rangle$ be a $1\mathrm{VASS}_k$, let $q_0 \in Q$, and let $\widehat{Q} \subseteq Q$. We can assume without loss of generality that all unary rules in U of the form $q \xrightarrow{c} q'$ have either $c = 1$ or $c = -1$. We will define[23] a formula $\varphi^{\mathcal{C}^k}$ of $\mathrm{LRV}_1^{\mathrm{D}}(\mathrm{AG}_{\approx}^{+})$, over an adequate set of labels, such that $\mathrm{CSReach}(\mathcal{C}^k, q_0, \widehat{Q})$ iff $\mathrm{SAT}_k(\varphi^{\mathcal{C}^k})$.

For the signature of the logic, we will consider the set of labels:

$$L = (U \cup B \cup \{*\}) \times \underline{k},$$

where $*$ is a symbol to represent that the node is a 'dummy node' that will be ignored in the translation to an incrementing derivation tree; also, dummy nodes will always be to the right of nodes labeled with any (t, i), for $t \neq *$. We notate

$$\varphi_{\mathrm{inc}} = \bigvee_{1 \leq j \leq k} \bigvee_{t = q \xrightarrow{+1} q'} (t, j) \qquad \text{and} \qquad \varphi_{\mathrm{dec}} = \bigvee_{1 \leq j \leq k} \bigvee_{t = q \xrightarrow{-1} q'} (t, j).$$

For $t = q \xrightarrow{v} \dots$ a branching or unary rule, we notate $\pi_h(t) = q$. We want to construct a formula $\varphi^{\mathcal{C}^k}$ whose satisfiability over k-ranked data trees is equivalent to $\mathrm{CSReach}(\mathcal{C}^k, q_0, \widehat{Q})$.

The truth of $\varphi^{\mathcal{C}^k}$ in a data tree T, r (where r is the root of T) expresses:

- $\pi_h(\pi_1(\ell(r))) = q_0$
- For each node x of T, we have:
  - i. If $\pi_1(\ell(x)) = q \xrightarrow{c} q'$ then $T, x \models \mathrm{EX}(a, 1) \wedge \mathrm{EX}(*, 2) \wedge \dots \wedge \mathrm{EX}(*, k)$, where $\pi_h(a) = q'$.
  - ii. If $\pi_1(\ell(x)) = q \to (q_1, \dots q_j)$, with $j > 0$, then $T, x \models \mathrm{EX}(a_1, 1) \wedge \dots \wedge \mathrm{EX}(a_j, j) \wedge \mathrm{EX}(*, j+1) \wedge \dots \wedge \mathrm{EX}(*, k)$, where $\pi_h(a_i) = q_i$.
  - iii. If $\pi_1(\ell(x)) = *$ then $T, x \models \neg\mathrm{EX}(\top)$.
  - iv. If $\pi_1(\ell(x)) = q \to \bar{\emptyset}$, then $T, x \models \neg\mathrm{EX}(\top)$.
  - v. If $T, x \models \neg\mathrm{EX}(\top)$ and $\pi_1(\ell(x)) = q \to \bar{\emptyset}$, then $q \in \widehat{Q}$.
  - vi. If $\pi_1(\ell(x)) = q \xrightarrow{+1} q'$ then $T, x \models v \approx \mathrm{EF} v \wedge \mathrm{AG}_{\approx v}(\varphi_{\mathrm{dec}})$.

[23] Abuse of notation: $\varphi^{\mathcal{C}^k}$ also depends on q_0 and $\widehat{Q}$.

In the framework of k-ranked data trees, item i and item ii give nodes tagged with a non-empty rule, exactly k-children, where the first ones are associated with the rule itself, and the rest are dummy nodes. Item iii ensures that dummy nodes are leaves, and item iv ensures that nodes corresponding to the empty rule are leaves. Item vi says that nodes tagged with an increment rule have a descendant in its same class, and that all descendants in the same class are tagged with a decrement rule. Item v assures that leaves of the tree that are tagged with an empty rule correspond with states in $\widehat{Q}$. Observe that the conditions i, ii, iii, iv, and v taken together imply that $T, x \models \neg\mathrm{EX}\top$ iff either $\pi_1(\ell(x)) = q \to \bar{\emptyset}$ or $\pi_1(\ell(x)) = *$.

We now write the formulas of the logic that correspond to all these conditions.

$$\varphi_0^{\mathcal{C}^k} = \bigvee_{\substack{1\le z\le k \\ t=q_0 \xrightarrow{\to} \cdots \in U\cup B}} (t,z)$$

$$\varphi_1^{\mathcal{C}^k} = \bigwedge_{\substack{1\le z\le k \\ t=q \xrightarrow{c} q'\in U}} \left((t,z) \to \left(\bigvee_{\pi_h(a)=q'} \mathrm{EX}(a,1) \wedge \bigwedge_{2\le j\le k} \mathrm{EX}(*,j)\right)\right) \tag{i}$$

$$\varphi_2^{\mathcal{C}^k} = \bigwedge_{\substack{1\le z\le k \\ t=q\to(q_1,\dots q_j)\in B}} \left((t,z) \to \bigvee_{\substack{a_1,\dots,a_j\in U\cup B \\ \text{s.t.}(\forall i)\pi_h(a_i)=q_i}} \left(\bigwedge_{1\le i\le j} \mathrm{EX}(a_i,i) \wedge \bigwedge_{j<i\le k} \mathrm{EX}(*,i)\right)\right) \tag{ii}$$

$$\varphi_3^{\mathcal{C}^k} = \bigwedge_{1\le z\le k} (*,z) \to \neg\mathrm{EX}\top \tag{iii}$$

$$\varphi_4^{\mathcal{C}^k} = \bigwedge_{\substack{1\le z\le k \\ t=q\to\bar{\emptyset}}} (t,z) \to \neg\mathrm{EX}\top \tag{iv}$$

$$\varphi_5^{\mathcal{C}^k} = \left(\neg\mathrm{EX}\top \wedge \bigvee_{\substack{1\le z\le k \\ t=q\to\bar{\emptyset}}} (t,z)\right) \to \bigvee_{\substack{1\le z\le k \\ t=q\to\bar{\emptyset}, q\in\widehat{Q}}} (t,z) \tag{v}$$

$$\varphi_6^{\mathcal{C}^k} = \varphi_{\mathrm{inc}} \to \left(v \approx \mathrm{EF}v \wedge \mathrm{AG}_{\approx v}(\varphi_{\mathrm{dec}})\right) \tag{vi}$$

For $i = 1 \dots 6$, let $\psi_i^{\mathcal{C}^k} = \varphi_i^{\mathcal{C}^k} \wedge \mathrm{AG}(\varphi_i^{\mathcal{C}^k})$ and finally define our desired $\varphi^{\mathcal{C}^k}$ such that $\mathrm{CSReach}(\mathcal{C}^k, q_0, \widehat{Q})$ iff $\mathrm{SAT}_k(\varphi^{\mathcal{C}^k})$:

$$\varphi^{\mathcal{C}^k} = \varphi_0^{\mathcal{C}^k} \wedge \bigwedge_{1\le i\le 6} \psi_i^{\mathcal{C}^k}.$$

Observe that the size of φ is polynomial on the size of $\mathcal{C}^k$. Note also that $\mathrm{AG}_{\approx}$ appears only positively in $\varphi^{\mathcal{C}^k}$, and hence $\varphi^{\mathcal{C}^k} \in \mathrm{LRV}_1^{\mathrm{D}}(\mathrm{AG}_{\approx}^{+})$.

For the general case of arbitrary n, we can assume without loss of generality that all unary rules in U of the form $q \xrightarrow{v} q'$ have either $v = e_i$ or $v = -e_i$ for some $1 \leq i \leq n$. Now, adding for each counter in $\mathcal{C}^k$ adequate new versions of φ_{inc}, φ_{dec}, and $\varphi_6^{\mathcal{C}^k}$, we can extend the previous arguments, yielding a proof of Theorem 181. Observe that this does not necessitate an increase in the number of variables of the logic. □

The satisfiability for this extension still has a reduction to the control-state reachability for VASS_k:

Theorem 182. *SAT_k-$\text{LRV}^{\text{D}}(\text{AG}^+_{\approx})$ is* ExpSpace*-reducible to* CSReach(VASS_k).

To show $\text{SAT}_k\text{-LRV}^{\text{D}}_n(\text{AG}^+_{\approx})$ reduces to CSReach$(n\text{VASS}_k)$ we will proceed incrementally, as it was done in §4.5. We begin with a simple logic, notated $\text{LRV}^{\text{D}-}_1(\text{AG}^+_{\approx})$, which consists only of formulas $\varphi \in \text{LRV}^{\text{D}}_1(\text{AG}^+_{\approx})$ that are conjuncts of terms of the form $v \star \text{EX}^i v$, $v \star \text{EF} v$, or their negation, where $\star \in \{\approx, \not\approx\}$, and, all occurrences of $\text{AG}_{\approx v}(\varphi)$ are of form $\text{AG}_{\approx v}(a)$, where a is a label, and whose labels rage over a finite fixed set L.

Let φ be a formula $\varphi \in \text{LRV}^{\text{D}-}_1(\text{AG}^+_{\approx})$, let

$$H = \{a \in L \mid \text{AG}_{\approx v}(a) \text{ is a subformula of } \varphi\}$$

and let $h = \#H$. We construct a $(h+1)\text{VASS}_k$ $\mathcal{C}^k_\varphi$ (via a procedure similar to that of §4.5.1, using Observation 177 to take labels into account), with two distinguished set of states Q_0 and $\widehat{Q}$ such that $\text{SAT}_k(\varphi)$ iff $\text{CSReach}(\mathcal{C}^k_\varphi, Q_0, \widehat{Q})$ (recall Observation 169).

Valid (d,k)-frames. We adjust our notion of valid (d,k)-frames to this framework. A (d,k)-frame is now a tuple $F = \langle N, E, \ell, \ell_1, \ell_2, \equiv \rangle$ that satisfies similar conditions as those of §4.5, and $\ell : N \to L$. We state next the differences with the (d,k)-frames seen in §4.5.1.

- For $x \in N$, $\ell_1(x)$ is a tuple $([\star_1 v \approx \text{EF} v], [\star_2 v \not\approx \text{EF} v], S)$, where $\star_1, \star_2$ can be either the empty string or $\neg$, and where S is a potentially empty set containing elements of the form $\text{AG}_{\approx v}(a)$ with $a \in L$.
- *Validity:* We extend the validity conditions of §4.5.1 with the following rules:
  - If $x \in N$ satisfies $\pi_1(\ell_1(x)) = [v \approx \text{EF} v]$ and $\text{AG}_{\approx v}(a) \in \pi_3(\ell_1(x))$, and if $y \in N$ is a descendant of x with $x \equiv y$, then $\ell(y) = a$.
  - If $x \in N$ satisfies $\text{AG}_{\approx v}(a), \text{AG}_{\approx v}(b) \in \pi_3(\ell_1(x))$ for $a \neq b$, then $\pi_1(\ell_1(x)) = [\neg v \approx \text{EF} v]$.
- We change the labeling function ℓ_2, such that its codomain is

$$\{\epsilon, \ominus_{a_1}, \ldots, \ominus_{a_n}, \ominus, \oplus, \oplus_{a_1}, \ldots, \oplus_{a_n}\}.$$

- For a root $r \in N$, $\ell_2(r) = \oplus_{a_i}$ iff $\pi_1(\ell_1(x)) = [v \approx \text{EF} v]$ and $\text{AG}_{\approx v}(a) \in \pi_3(\ell_1(r))$ and there is no descendant in the same frame in the same equivalence class.

- For a root $r \in N$, $\ell_2(r) = \oplus$ iff $\pi_1(\ell_1(x)) = [v \approx \mathrm{EF}v]$ and $\pi_3(\ell_1(r)) = \emptyset$ and there is no descendant in the same frame in the same equivalence class.

- For leaves $z \in N$, $\ell_2(z) = \ominus_{a_i}$ implies that $\ell(z) = a_i$, $\mathrm{AG}_{\approx v}(a_i) \in \pi_3(\ell_1(z))$ and there is no node in the frame at distance $< d$ from the root with the same equivalence class as z.

- For leaves $z \in N$, $\ell_2(z) = \ominus$ implies that $\pi_3(\ell_1(z)) = \emptyset$ and that there is no node in the frame at distance $< d$ from the root with the same equivalence class as z.

A (d,k)-frame is an ***a*-point of decrement of value *p*** iff it has a maximum of p leaves $z_1, \ldots, z_p$ in different equivalence classes and with $\ell_2(z_j) = \ominus_a$ for all j. It is a **neutral point of decrement of value *p*** iff it has a maximum of p leaves $z_1, \ldots, z_p$ in different equivalence classes and with $\ell_2(z_j) = \ominus$ for all j. A (d,k)-frame is an ***a*-point of increment** if its root r has $\ell_2(r) = \oplus_a$ and it is not an a-point of decrement. It is a **neutral point of increment** if $\ell_2(r) = \oplus$ and it is not an a-point of decrement for any $a \in L$.

The automaton $\mathcal{C}^k_\varphi$. We define the $(h+1)\mathrm{VASS}_k$ $\mathcal{C}^k_\varphi$ and the sets $Q_0, \widehat{Q}$ similarly as in the case of $\mathcal{A}^k_\varphi$ in §4.5.1, using Observation 177 to take labels into account with the notion of 1-consistency between the (d,k)-frames just defined.

We define $\mathcal{C}^k_\varphi$ as follows:

- We fix the dimension to be $h+1$. Let $H = \{a_1, \ldots, a_h\}$. For $i \leq h$, the i-th coordinate will correspond with the label a_i. Intuitively, i-th coordinate will we used to count instances of $v \approx \mathrm{EF}v \wedge \mathrm{AG}_{\approx v}(a)$, yet unsatisfied.

- The set of states of $\mathcal{C}^k_\varphi$ is $\mathcal{F}_{d,k}$, the set consisting of all valid (d,k)-frames as defined above.

- Unary rules. Let F_1 and F_2 be (d,k)-frames. We have the following rules[24]:

  - $F_1 \xrightarrow{-ne_i} F_2$ if F_1 is an a_i point of decrement of value n, and F_2 is equal to F_1, except that $\ell_2^{F_2}$ is defined as follows:

$$\ell_2^{F_2}(x) = \begin{cases} \epsilon & x \text{ is a leaf of } F_1 \text{ with } \ell_2^{F_1}(x) = \ominus_{a_i}; \\ \ell_2^{F_1}(x) & \text{otherwise.} \end{cases}$$

  - $F_1 \xrightarrow{-me_{h+1}} F_2$ if either:

[24] There will we a non-deterministic choice of the order of decrements if there are leaves with different $\ominus_{a_j}$, but the order of these operations is not relevant.

* F_1 is a neutral point of decrement of value m and F_2 is equal to F_1, except that $\ell_2^{F_2}$ is defined as follows:

$$\ell_2^{F_2}(x) = \begin{cases} \epsilon & x \text{ is a leaf of } F_1 \text{ with } \ell_2^{F_1}(x) = \ominus; \\ \ell_2^{F_1}(x) & \text{otherwise.} \end{cases}$$

or

* F_1 is a a_i-point of decrement of value $n \geq m$ for some a_i, and F_2 is an a_i point of decrement of value $n - m$ such that F_2 has m fewer instances of distinct equivalence classes of nodes with $\ominus_{a_i}$, that is: F_2 is equal to F_1, except for $\ell_2^{F_2}$: for any node x, $\ell_2^{F_1}(x) = \ominus_{a_i}$ implies either $\ell_2^{F_2}(x) = \ominus_{a_i}$ or $\ell_2^{F_2}(x) = \ominus_{a_i}$, and if $\ell_2^{F_1}(x) \neq \ominus_{a_i}$ then $\ell_2^{F_1}(x) = \ell_2^{F_2}(x)$.

– $F_1 \xrightarrow{e_i} F_2$ if F_1 is an a_i point of increment, and F_2 is equal to F_1, except that $\ell_2^{F_2}$ is defined as follows:

$$\ell_2^{F_2}(x) = \begin{cases} \epsilon & x \text{ is the root of } F_1; \\ \ell_2^{F_1}(x) & \text{otherwise.} \end{cases}$$

– $F_1 \xrightarrow{e_{h+1}} F_2$ if F_1 is a neutral point of increment, and F_2 is equal to F_1, except that $\ell_2^{F_2}$ is defined as follows:

$$\ell_2^{F_2}(x) = \begin{cases} \epsilon & x \text{ is the root of } F_1; \\ \ell_2^{F_1}(x) & \text{otherwise.} \end{cases}$$

- Branching rules: $F \to (F_1, \ldots, F_i)$, if F is 1-consistent with $F_1, \ldots, F_i$.

We define the sets $Q_0, \widehat{Q}$:

- Q_0 consists of initial frames. A (d, k)-frame F of root r is initial iff the following conditions hold:
  – F, r satisfies all terms of φ of the form $v \star \mathrm{EX}^i v$ or $\neg v \star \mathrm{EX}^i v$;
  – if $v \approx \mathrm{EF}v$ [resp. $v \not\approx \mathrm{EF}v$] is a positive conjunct of φ, then the root $r \in N^F$ satisfies $\pi_1(\ell_1^F(r)) = [v \approx \mathrm{EF}v]$ [resp. $\pi_2(\ell_1^F(r)) = [v \not\approx \mathrm{EF}v]$];
  – if $v \approx \mathrm{EF}v$ [resp. $v \not\approx \mathrm{EF}v$] is a negative conjunct of φ, then for all descendants y of the root $r \in N^F$, it holds that $\pi_1(\ell_1^F(y)) = [\neg v \approx \mathrm{EF}v]$ [resp. $\pi_2(\ell_1^F(y)) = [\neg v \not\approx \mathrm{EF}v]$].
  – If $b \in L$ and b is a conjunct of φ, then $\ell(r) = b$, for r the root of F.
  – If $\mathrm{AG}_{\approx v}(a)$ is a conjunct of φ, then $\mathrm{AG}_{\approx v}(a) \in \pi_3(\ell_1(r))$, for r the root of F.
- $\widehat{Q}$ is the set of (d, k)-frames in $\mathcal{F}_{d,k}$ that consists solely of one node.

Proposition 183 and Proposition 183, the analogous results to Propositions 170 and 175, hold for $\varphi \in \mathrm{LRV}_1^{\mathrm{D}-}(\mathrm{AG}_{\approx}^{+})$ and the corresponding $(h+1)\mathrm{VASS}_k$ $\mathcal{C}_\varphi^k$ and the states $Q_0, \widehat{Q}$. Hence we arrive at:

$$\mathrm{SAT}_k(\varphi) \text{ iff } \textsc{CSReach}(\mathcal{C}_\varphi^k, Q_0, \widehat{Q}).$$

As in Proposition 178, we can extend these results to the full logic, and then extend for the general case where we allow any formula $\mathrm{AG}_{\approx v}(\eta)$ to appear in φ, completing the proof of Theorem 182.

Proposition 183. *If there is a solution to the problem of the* $\textsc{CSReach}(\mathcal{C}^k, q_0, \widehat{Q})$ *then there is a solution to the* SAT_k *problem of* $\varphi^{\mathcal{C}^k}$ *over the logic* $\mathrm{LRV}_1^{\mathrm{D}}(\mathrm{AG}_{\approx}^{+})$.

Sketch of the proof. Let S be an incrementing derivation tree that is a solution of $\textsc{CSReach}(\mathcal{C}^k, q_0, \widehat{Q})$. We want to prove that there is a k-ranked tree T_S where the root r satisfies $\varphi^{\mathcal{C}^k}$. Nodes of T_S will correspond to the nodes of S, and their labels will be determined by the rule that is invoked on them in S and by their position as children of some other node (and we choose to assign $\pi_2(\ell(r)) = 1$). When a node has at least one child but strictly less than k, we add dummy nodes (with labels of the form $(*, i)$) in the proper order so as to arrive to exactly k children. With all this, $\psi_1^{\mathcal{C}^k} \wedge \psi_2^{\mathcal{C}^k} \wedge \psi_3^{\mathcal{C}^k} \wedge \psi_4^{\mathcal{C}^k}$ is satisfied at r. Since S is a solution for $\textsc{CSReach}(\mathcal{C}^k, q_0, \widehat{Q})$, $\varphi_0^{\mathcal{C}^k}$ holds at the root of T_S, and all leaves z of T_S with $\pi_1(\ell(z))$ of the form $q \to \bar{\emptyset}$ have $\pi_h(\pi_1(\ell(z))) \in \widehat{Q}$, and thus $\psi_5^{\mathcal{C}^k}$ is satisfied at r. For the equivalence relation, we first put all nodes in different equivalence classes. Then, since S is a solution for the control-state reachability problem, whenever there is a $(+1)$ increment rule there is a (-1) decrement rule further down, and we can make this assignation injectively, yielding corresponding joining of the equivalence classes. This is enough to satisfy $\psi_6^{\mathcal{C}^k}$ at r. □

Proposition 184. *If there is a solution to the* SAT_k *problem of* $\varphi^{\mathcal{C}^k}$ *over* $\mathrm{LRV}_1^{\mathrm{D}}(\mathrm{AG}_{\approx}^{+})$ *then there is a solution to* $\textsc{CSReach}(\mathcal{C}^k, q_0, \widehat{Q})$.

Sketch of the proof. Let T be a k-ranked tree whose root r satisfies $\varphi^{\mathcal{C}^k}$. We want to construct S_T, an incrementing derivation tree of $\mathcal{C}^k$ that is a solution to the control-state reachability problem. The idea is that the conjuncts $\psi_1^{\mathcal{C}^k}$, $\psi_2^{\mathcal{C}^k}$, and $\psi_4^{\mathcal{C}^k}$ provide a natural translation from the nodes and labels of each node of T to the structure and invoked rules of S_T. Because of $\psi_3^{\mathcal{C}^k}$, we can ignore those nodes $x \in T$ with $\pi_h(\ell(x)) = *$. $\varphi_0^{\mathcal{C}^k}$ ensures that the derivation tree starts from q_0, while $\psi_5^{\mathcal{C}^k}$ implies that all leaves of S_T are in $\widehat{Q}$. To check that all leaves of S_T have the counter set at 0, observe that $\psi_6^{\mathcal{C}^k}$ ensures that whenever an increment rule ($+1$ to the counter) is invoked, there is further down in the derivation an application of a decrement rule (-1 to the counter), furthermore, there is at least one decrement for each increment, since $\psi_6^{\mathcal{C}^k}$ also ensures that two nodes labeled with some increment rule cannot be in the same equivalence class if one is ancestor of the another. □

4.7 From LRV to MVASS$_k$

The reduction from LRV$^{\mathrm{D}}$ to VASS$_k$ from Section 4.5 cannot be extended to treat LRV. The main problem is that the branching nature of the counters in a CSReach(VASS$_k$) will be insufficient to represent some classes of data trees (which can be needed to model some formulas). When we have tests of the form $u_1 \approx \mathrm{EF}u_2$ with $u_1 \neq u_2$ distinct variables, we can no longer reason in terms of "one coordinate i for each variable u_i", where the i-th component in the configuration of the VASS$_k$ codes, intuitively, how many distinct data values must be seen on variable u_i in the subtree as shown in Section 4.5. In fact, when working with LRV, a data value may appear in *several* variables, as a result of allowing formulas like $u_1 \approx \mathrm{EF}u_2 \wedge u_1 \approx \mathrm{EF}u_3$. This means that we need to reason in terms of *sets* of variables, where each component i is associated with a non-empty subset U_i of the variables appearing in the input formula φ; this time, component i counts how many data values must appear in the subtree under all the variables of U_i. This, in principle, poses no problem for the non-branching case: in fact, this kind of coding (indexing one coordinate of the configuration for each subset of variables) was used in [35] to show a reduction from LRV to VASS on data words. However, on data trees, this coding breaks with the semantics of the branching rules of VASS$_k$.

As an example, suppose we work with two variables u, v and we thus have dimension 3 —the first component is associated with $\{u\}$, the second with $\{v\}$ and the third with $\{u, v\}$. Suppose that there are n ancestor nodes that have to satisfy both $u \approx \mathrm{EF}u$ and $u \approx \mathrm{EF}v$, which at the current configuration of the VASS$_k$ is witnessed by the vector $(0, 0, n)$. Intuitively, this means that there are n data values that must appear in the subtree under a variable u and also under v (though not necessarily at the same node) in the data tree the automaton is trying to find. Hence, as part of the "branching" instruction of this configurations into the configuration of the left and right children, one must contemplate the possibility of obtaining, for instance, $(n, 0, 0)\,(0, n, 0)$, saying that the left subtree contains n distinct data values for u, and the right child contains n data values for v. But it could be $(n - t, 0, t)\,(0, n - t, 0)$, or $(0, 0, n - t)\,(0, 0, t)$, etc. In other words, components need to be mixed in a more complex way that is not allowed in VASS$_k$ branching rules. In particular, some sort of transfers between coordinates must be necessary. This is precisely the behavior that we can encode into MVASS.

Theorem 185. *SAT$_k$-LRV$_n$ is reducible to* CSReach(2^n-MVASS$_k$).

As a corollary, due to Theorem 164, we have that SAT$_k$-LRV is decidable. We remark that, similarly as done in [36], one can add formulas of the form $u \star \mathrm{EF}[\varphi]v$ stating that there is a descendant witnessing $u \star \mathrm{EF}v$ and verifying φ, while preserving this reduction.

Proof. Using the merging rules as described in Section 4.7, the reduction from LRV$^{\mathrm{D}}$ to VASS$_k$ of Section 4.5 can be modified to obtain a reduction from LRV to MVASS$_k$. Frames and its notion of validity are extended to treat set of variables. In particular, now the points of increment and decrement are always relative to a set of variables. This follows, very roughly, the idea of coding from [36] in the setup built in Section 4.5, but now some special

care must be considered because of the non-linearity of a tree. One must decide in advance to which leaf of the frame the satisfaction of data demands will be delegated. The resulting MVASS_k now has dimension *exponential* in the number of variables of the input formula.

Concretely, in order to encode this logic we need to make the following changes to the set of frames $\mathcal{F}_{d,k}$ we work with.

First of all, the labeling function ℓ_1 now labels pairs of *sets* of formulas. These formulas labelled by ℓ_1 are of the form

- in the first component $u \approx \mathrm{EF}v$ or $\neg(u \approx \mathrm{EF}v)$
- in the second component $u \not\approx \mathrm{EF}v$ or $\neg(u \not\approx \mathrm{EF}v)$

for any pair of variables u, v used in the input formula. For simplicity, we write $\psi \in \ell_1(x)$ (or, alternatively, that x is ℓ_1-labeled with ψ) to denote that ψ is either in the first or second component of $\ell_1(x)$.

Further, instead of having one equivalence relation $\equiv$ over the set of nodes, we have an equivalence relation $\equiv$ over pairs (x, u) where x is a node of hte frame and u an attribute variable of the input formula φ. This is to account for the possibility that different attributes can have the same data value.

In light of this, the formulas labeled by ℓ_1 must 'respect' $\equiv$. That is, if $u \approx \mathrm{EF}v \in \ell_1(x)$ [resp. $u \not\approx \mathrm{EF}v \in \ell_1(x)$] and $(x, u) \equiv (x, u')$ then $u' \approx \mathrm{EF}v \in \ell_1(x)$ [resp. $u' \not\approx \mathrm{EF}v \in \ell_1(x)$].

More importantly, the labeling ℓ_2 must be changed to reflect the fact that

(1) there may be several demands for the same attribute, as a result of formulas like $u \approx \mathrm{EF}v \wedge u' \approx \mathrm{EF}v$ (as we will see next, this is the reason for the first parameter of $\oplus$),

(2) there may be several attributes in a demand for equality, as a result of formulas like $u \approx \mathrm{EF}v \wedge u \approx \mathrm{EF}v'$,

(3) a point of decrement needs to be a point that has some attributes U which are not connected by equality to any 'local' ancestor and they are connected possibly to some other attributes V in the descendants.

Formally, the mapping ℓ_2 now labels nodes with $\oplus(U, V)$ and/or $\ominus(U, V)$, where U, V are sets of attribute variables. Each node x can receive more than one $\oplus$ or $\ominus$ label, that is, ℓ_2 is a function from nodes to subsets of $\{\oplus(U, V) \mid U, V \subseteq \mathbb{V}\} \cup \{\ominus(U, V) \mid U, V \subseteq \mathbb{V}\}$, assuming $\mathbb{V}$ is the set of variables used in the input formula.[25] The idea is that $\oplus(U, V)$ holding at x means that there must be a data value appearing in the subtree at x under all the variables of V (possibly at different nodes), which is equal to the u-attribute of the k-ancestor of x, for every $u \in U$. On the other hand, $\ominus(U, V)$ holding at x means that the data value of the U-attributes of x (which are all the same) do no not appear in any i-ancestor of x ($i \leq k$), and they will appear in the future with attributes V.

We add the following conditions.

[25] It is worth remarking that $\ell_2(x)$ is always a set of size linear in $|\mathbb{V}|$ due to the next conditions.

- For any two labels $\oplus(U,V)$ and $\oplus(U',V')$ at the same node, U and U' are disjoint. For any two labels $\ominus(U,V)$ and $\ominus(U',V')$ at the same node, U and U' are disjoint.
- For every leaf x which is ℓ_2-labeled with $\oplus(U,V)$ we have that U is an equivalence class of $\{(u,v) \mid (r,u) \equiv (r,v)\}$, where r is the root node.
- For every leaf x which is ℓ_2-labeled with $\ominus(U,V)$ we have that for some $v \in \mathbb{V}$ we have

$$U = \{u \mid (x,u) \equiv (x,v)\},$$
$$V = \{u \mid [v \approx \mathrm{EF}u] \in \ell_1(x)\},$$

  and that there is no ancestor y of x so that $(x,u) \equiv (y,v')$ for some $u \in U$, $v' \in \mathbb{V}$.
- There exists an ℓ_1-labeling $u \approx \mathrm{EF}v$ holding at the root r if, and only if, there exists a node x at some depth i so that either
  - $(r,u) \equiv (x,v)$, or
  - $(r,u) \equiv (x,v')$ for some v' and $v' \approx \mathrm{EF}v$ in $\ell_1(x)$, or
  - $i = k$ and x is ℓ_2-labeled with $\oplus(U,V)$ with $U = \{u' \mid (r,u) \equiv (r,u')\}$ and $v \in V$.
- There exists an ℓ_1-labeling $u \not\approx \mathrm{EF}v$ holding at the root r if, and only if, there exists a node x at some depth i so that either
  - $(r,u) \not\equiv (x,v)$, or
  - $(r,u) \equiv (x,v)$ and $v \not\approx \mathrm{EF}v$ in $\ell_1(x)$.

1-step consistency is preserved as before. Now a *point of increment for* V is a frame whose root is labeled with $\oplus(U,V)$ for some U; whereas a *point of decrement for* W is a frame whose root is labeled with $\ominus(U,V)$ for $U \cup V = W$.

Finally, the automaton $\mathcal{A}^k_\varphi$ is built as for the other reduction, with the exception that now its dimension is exponential. As in the previous reductions, the frames are the states of the automaton, where the initial frames are those that do not contain $\oplus/\ominus$ tags at nodes at distance $< k$ from the root, and whose root labeling is consistent with the satisfaction of the formula. In the automaton, we have one coordinate associated with every non-empty subset $V \subseteq \mathbb{V}$ of attribute variables (remember that we use $\bar{e}_V$ to denote $\bar{e}_i$ for the coordinate i associated with V and $\bar{e}_\emptyset$ to denote $\bar{0}$). Unary rules now follow a logic of first decrementing all the $\ominus$ at the root, and then incrementing all the $\oplus$ at the root. (The $\oplus/\ominus$ tags of other nodes are differed to the moment when they will be at the root of a frame.) That is, unary rules $(F_1, -\bar{e}_{U\cup V}, F_2)$ whenever F_1 has $\ominus(U,V)$ at the root, F_2 is just like F_1 but without the $\ominus(U,V)$ at the root. We have unary rules $(F_1, \bar{e}_V, F_2)$ whenever F_1 has no $\ominus$-labels at the root, it has a $\oplus(U,V)$ label at the root, and F_2 is the result of removing

$\oplus(U, V)$ at the root. Merging rules are built as it was explained before. That is, we have $(F_1, \bar{0} + B^*, (F'_1, \ldots, F'_{k'}))$ whenever F_1 is 1-consistent with $(F'_1, \ldots, F'_{k'})$, and B consists of all vectors $(\bar{e}_V \bar{e}_{U_1} \cdots \bar{e}_{U_k})$, so that $V \neq \emptyset$ and $V = \bigcup_i U_i$. The partial order $\preceq$ will then be the subset ordering on the components: $i \preceq j$ if the set associated to i is contained in that associated to j. It is not hard to check that if $\mathcal{A}^k_\varphi$ has a solution for CSReach if and only if φ is satisfiable on k-ranked data trees using precisely the same ideas as in the proof of Proposition 170. □

Conclusions and future work

> "And now it seems there are lots of other worlds as well. When I think I might die without seeing a hundredth of all there is to see it makes me feel," he paused, then added "well, humble I suppose. And very angry, of course."
>
> *The Colour of Magic*
> Terry Pratchett

Conclusions

In Chapter 1 we developed the model theory of $\mathrm{XPath}_{=}(\downarrow)$ and $\mathrm{XPath}_{=}(\uparrow\downarrow)$ both for node expressions and path expressions. For node expressions, we designed the tools of saturation and quasi-ultraproducts in order to obtain definability and separation results. For path expressions, we first developed adequate notions of binary bisimulation over two-pointed data trees, such that a Hennessy-Milner-style characterization theorem held and thus the notions coincided with logical indistinguishability. We also proved a van Benthem-style characterization theorem for binary $\mathrm{XPath}_{=}(\downarrow)$-bisimulation which connects this logic with the fragment of first order logic with two free variables. We then proceeded to definability and separation theorems for the framework of two-pointed data trees and path expressions.

These model-theoretical results are summarized in the following table:

	Node expressions		Path expressions	
	Downward	Vertical	Downward	Vertical
Bisimulation notion	[45, §3.1.2]	[45, §3.2.4]	§1.4.1 Thm. 56	§1.4.2 Thm. 69
Characterization	[45, §4.1]	Fails [45, §4.2]	§1.4.1 Thm. 63	Fails
Definition of saturation	§1.3.1	§1.3.1	§1.5.1	§1.5.1
Definition of quasi-ultraproduct	§1.3.2	§1.3.2	§1.5.2	§1.5.2
Definability	§1.3.3 Thms. 25,26,27	§1.3.3 Thms. 30,31,32	§1.5.3 Thms. 80,81,82	§1.5.3
Separation	§1.3.4 Thms. 37,38	§1.3.4 Thms. 39,40	§1.5.3 Thms. 83,84	§1.5.3

In Chapter 2 we found (sound and complete) equational axiomatizations for $\mathrm{XPath}_{=}(\downarrow)$ and for its data-inequality-free fragment $\mathrm{XPath}_{=}(\downarrow)^{-}$. In order to do this, we devised normal form theorems stating that consistent node or path expressions are provably equivalent (in our axiomatic systems) to disjunction or union of expressions in a normal form. Then we gave a method for constructing, for every consistent node expression, a finite data tree where it is satisfied at the root. Moreover, this proof-theoretical construction already showed the (unsurprising) fact that there is a primitive recursive function f such that if a formula of size n from any of these logics is satisfiable over a class of finite/arbitrary data trees/graphs, then it is satisfiable in a data tree of size bounded by $f(n)$.

In Chapter 3 we began by modifying XPath to fit into an edge-labeled framework, and we showed that the notion of $\mathrm{XPath}_{=}(\downarrow)$-bisimulation was easily adapted to this context with the $\mathrm{XPath}_{=}(\downarrow_{\mathbf{a}})$-bisimulation. More importantly, we saw that the expansion from the universe of data trees into that of data graphs did not require a modification of the bisimulation clauses.

After analyzing the complexity of deciding bisimilarity in different conditions, we saw that while computing (bi)simulations on finite data graphs is generally PSPACE-COMPLETE, tractability can be regained by either restricting the topology of the graph or by relaxing the conditions for bisimulation. Furthermore, several upper bounds continue to hold in $\mathrm{XPath}_{=}(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})$, where navigational reverses are added to the logic (the only case left open is for DAGs). The following table summarizes our results:

Model	Problem	Logic	
		$\mathrm{XPath}_{=}(\downarrow_{\mathbf{a}})$	$\mathrm{XPath}_{=}(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})$
Graph	(bi)simulation	PSPACE-c	PSPACE-c
	p-(bi)simulation	CO-NP	CO-NP
	c-(bi)simulation	PTIME-c	PTIME
DAG	(bi)simulation	CO-NP-c	?
Tree	(bi)simulation	PTIME	PTIME

In Chapter 4 we have shown connections between counter systems and data logics on ranked data trees. In particular, this has yielded decision procedures for data logics and a new model of branching computation of VASS.

We have proved that the satisfiability problem for $\mathrm{LRV}^{\mathrm{D}}$ on k-ranked data trees is reducible, in exponential space, to the control-state reachability problem for VASS_k, yielding a decision procedure. We expanded this logic with positive instances of the operator $\mathrm{AG}_{\approx v}(\varphi)$, and showed that the satisfiability problem for this new logic is equivalent to the control-state reachability for VASS. We also introduced MVASS and proved that the bottom-up coverability (and control-state reachability) problem for MVASS is in 3ExpTime. Finally, we showed that the satisfiability for LRV on k-ranked data trees can be reduced to the control-state reachability for MVASS_k, yielding a decision procedure for this full logic.

Future work

Although so far we have mostly focused our research on the theoretical aspects of bisimulation, we also want to apply our results and insights into actual optimization of database query languages. Since bisimulations coincide with logical indistinguishability over finite models, finding a maximal autobisimulation can be used as a tool to avoid unnecessary computations when calculating the answer to a query, although there are technical aspects that mean that over data-aware logics this idea is not as easily implemented as it is in the data-oblivious case. There are many interesting problems that would be part of this area of study, such as the trade-off between the time consumed in finding the autobisimulation and the expected time saved via the subsequent speed-up to the queries, the available strategies for an incremental construction of a maximal autobisimulation, the pitfalls and advantages of calculating autobisimulations for dynamic databases, et cetera. Beyond using different benchmarks to make various analyses of performance, and fine-tuning our algorithms depending on the target framework, we can also vary our models (data trees, data graphs, and other unexplored possibilities) and the syntactical fragments of our logics (some syntactical restrictions may encompass most of the queries that are actually used in databases, and at the same time they may have bisimilarity problems of greatly reduced complexity). The possibilities of this avenue of research are manifold, and we look forward to working on it.

Regarding research of a more theoretical nature, we would like to study the model theory and proof theory of different fragments of $\mathrm{XPath}_{=}$, or even of LRV, over data trees or over other models such as data graphs. As we did in Chapter 1, this may include the development of bisimulation notions that coincide with logical indistinguishability, the statement of van Benthem-style characterizations theorems (if possible), and the proof of definability and separation theorems; all this will probably require theoretical scaffolding of independent interest, such as the adequate design of normal forms, saturation and quasi-ultraproduct notions, et cetera. While the proofs of Chapter 2 were laborious, we may obtain new axiomatizations following the ideas that were presented there; alternatively,

we may start working with other types of axiomatizations, such as sequent calculus in the vein of [10], where finding a complete axiomatization gives rise to low-complexity decision procedures for the satisfiability problem. While for many of these logics tight bounds for the satisfiability problem are already known [46, 47, 43, 50, 65], working with axiomatizations provides alternative demonstrations that are more pliable to extensions and modifications, as long as the sets of axioms are modular enough. We are also interested in the possible application of axiomatizations as a tool for query optimization. This study would involve the search for rewriting strategies that are effective in practical cases in transforming queries into equivalent but less computationally intensive forms.

After developing the proper notions of bisimulation for various fragments of $\mathrm{XPath}_{=}$ and LRV, we want to delve in the analysis of computational aspects such as the complexity of the bisimilarity problem, in the vein of what we did in Chapter 3. Once again, we could modify different parameters to observe what are the corresponding changes in complexity; this may involve finding particular data structures with properties that behave well with respect to our definition of bisimulation, or finding adequate notions of local bisimulation depending on the fragment in question. This area of research ties in naturally with our previously stated goal of using bisimulations as a tool for optimizing database querying.

We also intend to deepen our study into the connection between logics and branching counter systems. While in Chapter 4 the focus has been put on *ranked* data trees, we also envisage working on unranked trees in the future. In particular, we remark that while LRV naturally functions over unranked trees, there are no well-known models of branching counter systems with *unbounded* branching. This may lead to new natural models featuring some sort of unbounded parallel computations with good computational properties.

We believe that SAT_k-$\mathrm{LRV}(\mathrm{AG}^{+}_{\approx})$ is equivalent to the control-state reachability problem for MVASS_k, but we did not prove it. Additionally, we want to determine the precise complexity of $\mathrm{CSReach}(\mathrm{MVASS}_k)$, which lies between 2ExpTime and 3ExpTime. We intend to answer both questions in future works.

We are also interested in considering other modalities in our logics, with branching (and more $\mathrm{XPath}_{=}$-like) tests such as $\mathrm{EX}^i v \star \mathrm{EF} u$ and $\mathrm{EF} u \approx \mathrm{EF} v$, or tests including past such as $u \approx \mathrm{EF}^{-1} v$ and $\mathrm{EF}^{-1} u \approx \mathrm{EF} v$.

Resúmenes en español

Introducción y preliminares

¿Qué es una lógica? De cierta manera, una lógica es una forma de razonar sobre un cierto dominio de discurso, es un lenguaje de símbolos y reglas que nos permite hablar acerca de verdad y consecuencia, acerca de lo que *puede ser* y lo que *debe ser* cuando uno parte de ciertas premisas. Más formalmente, podemos pensar que una lógica está compuesta de dos partes interconectadas: la sintáctica y la semántica. Básicamente hablando, la parte *sintáctica* establece un conjunto de símbolos utilizables y un conjunto de fórmulas construidas con esos símbolos, y puede incluir reglas de deducción que conectan a las fórmulas y dictan qué es consecuencia de qué. Pero, por sí solos, estos símbolos y fórmulas carecen de significado explícito: este es dado por la *semántica* de la lógica, la interpretación de sus fórmulas dentro de cierto marco de referencia. Dada una interpretación para los símbolos y fórmulas de la lógica, una fórmula adquiere un significado, volviéndose una declaración acerca de un objeto o modelo particular, y como tal puede resultar verdadera o falsa. Ahora bien, hay diversas maneras de juzgar qué tan adecuada es una lógica. Uno podría, por ejemplo, tratar de medir su *expresividad*, el grado en el cual podemos expresar diversas propiedades (que podemos definir en términos informales o meta-lógicos) usando únicamente fórmulas de nuestra lógica. Aunque mayor expresividad sería, manteniendo todo lo demás igual, algo mejor, usualmente tiene un costo asociado; en lógicas más expresivas puede ser más difícil determinar algunas características de fórmulas, como ser si dos fórmulas son semánticamente equivalentes, si una fórmula es verdadera en un módelo particular, si una fórmula puede ser satisfecha en algún modelo, si una fórmula puede ser deducida a partir de cierto conjunto de axiomas, et cetera. Efectivamente, otras dimensiones sobre las cuales se puede juzgar una lógica son aquellas relacionadas con la *decibilidad* y *complejidad algorítmica* de tales problemas: ¿son siquiera solubles por medios efectivos? de serlo, ¿qué tan trabajoso es para un algoritmo resolverlos? Además, hacer a una lógica más expresiva puede tener otros costos asociados, como ser el sacrificio de legibilidad o de concisión notacional. Este es el tipo de concesiones y balances que explican la abundancia existente de lógicas, y justifican entonces por qué podríamos usar una lógica a pesar que existan otras más expresivas. Este tipo de asuntos serán visibles cuando analicemos las lógicas principales de esta tesis: diversos fragmentos de $\mathrm{XPath}_{=}$ y otras lógicas con datos.

Un lenguaje de consulta de bases de datos es un lenguaje computacional que es usado para responder preguntas acerca de una base de datos, como ser si una entrada tiene

una propiedad particular. Los lenguajes de consulta pueden ser estudiados formalmente como lógicas, haciendo abstracciones matemáticas adecuadas de las estructuras de datos que son usadas e identificando a las *consultas* con sus correspondientes *fórmulas*. Hecho esto, obtenemos una equivalencia entre las respuestas a las consultas en la base de datos y la semántica de su traducción a fórmulas de la lógica (sobre los modelos correspondientes a esas bases de datos). XPath (XML Path Language) es un lenguaje de consulta diseñado para trabajar sobre documentos XML, y Data XPath, aquí llamado $\text{XPath}_=$, es la lógica correspondiente diseñada para trabajar sobre árboles con datos (data trees), una abstracción de los documentos XML que consiste de un árbol con raíz donde cada nodo posee una única etiqueta (label) y un único valor de dato (data value). $\text{XPath}_=$ es cercano en varios aspectos a la lógica modal básica (BML; basic modal logic): tiene modalidades de navegación (como ser $\uparrow$ o $\downarrow$) y es local (las fórmulas son analizadas en nodos particulares de los árboles, y, dependiendo del fragmento de la lógica, la profundidad hasta la cual pueden 'ver' está acotada por el tamaño de la fórmula). $\text{XPath}_=$ tiene dos tipos de fórmulas: *expresiones de nodo*, las cuales, como las fórmulas de BML, son evaluadas en nodos, y *expresiones de camino*, las cuales son analizadas en pares de nodos. A diferencia de BML, donde los nodos ni siquiera tienen data values, $\text{XPath}_=$ es capaz de hacer *comparaciones de datos* entre nodos, esto es, puede verificar si dos nodos tienen el mismo valor de dato o no, pero no puede consultar el valor concreto de un nodo. Además, como veremos, esta diferencia en capacidades es fundamental, y BML no puede expresar apropiadamente el concepto de comparación de datos aún si sus nodos son enriquecidos con valores de dato. Mientras que la teoría de BML está bien desarrollada, este no es el caso para $\text{XPath}_=$. En esta tesis avanzamos en los siguientes aspectos teóricos de $\text{XPath}_=$:

Teoría de modelos. Estudiamos problemas como *definibilidad*, que pregunta si una clase de modelos puede ser caracterizada por fórmulas de $\text{XPath}_=$; y *separación*, que pregunta si dos clases pueden ser separadas por fórmulas. Nuestro trabajo en definibilidad y separación para $\text{XPath}_=$ está inspirado en los correspondientes resultados para lógica modal básica; para poder ajustar las demostraciones a nuestro marco, desarrollamos nociones apropiadas de saturación y ultraproducto, y demostramos varios resultados técnicos antes de llegar a los teoremas buscados. También estudiamos problemas relacionados con bisimulación, un concepto central que aproxima equivalencia lógica mediante una noción más estructural. Creamos nociones adecuadas de bisimulación para expresiones de camino, y obtuvimos varios resultados clave para este marco. También observamos que diversos resultados de $\text{XPath}_=$ sobre árboles con datos pueden ser extendidos al contexto más amplio de grafos con datos.

Teoría de prueba. Obtuvimos *axiomatizaciones* correctas y completas para dos fragmentos de $\text{XPath}_=$, esto es, axiomas que solo permiten probar equivalencias universalmente verdaderas, y tales que todas las equivalencias universalmente verdaderas sobre árboles con datos pueden ser probadas a partir de esos axiomas. Al hacer esto, extendimos un trabajo previo hecho en el caso más simple de XPath sin comparación de datos. Demostramos la

completitud de nuestras axiomatizaciones a través de teoremas de forma normal, y haciendo las construcciones algo intrincadas que prueban que todas las expresiones consistentes de nodo que están en forma normal son satisfacibles.

Aspectos computacionales. Expandimos nuestro estudio desde el universo de árboles con datos al caso más general de *grafos con datos*, los cuales son ampliamente utilizados como una abstracción para bases de datos en forma de grafos. En esta área, extendimos resultados de $\text{XPath}_{=}$ que originalmente se basaban en árboles con datos, y estudiamos los problemas de similaridad y bisimilaridad entre grafos con datos desde una perspectiva computacional. Clasificamos a estos problemas de forma ajustada en varias clases de complejidad, dependiendo de la noción de (bi)simulación elegida y de las restricciones que aplicamos a la clase de modelos.

Adicionalmente, continuamos nuestro estudio de lógicas para razonar sobre árboles con datos, expandiéndonos hacia las *lógicas de valores repetidos* (LRV; logics of repeating values) sobre árboles con datos múltiples. Comenzamos generalizando un trabajo previo hecho sobre palabras con datos múltiples, probando la inter-reducibilidad entre el problema de satisfabilidad de LRV disjunto sobre árboles de rango finito y el problema de cubrimiento para VASS (branching vector addition systems with states). Luego presentamos una extensión de BVASS llamada MVASS (merging VASS), y demostramos que el problema de satisfabilidad de LRV sobre árboles de rango finito se puede reducir al problema de control-state reachability para MVASS.

Definibilidad y bisimulación binaria

En este capítulo nos enfocamos en estudiar la teoría de modelos y el poder expresivo de $\text{XPath}_{=}(\downarrow)$ y $\text{XPath}_{=}(\uparrow\downarrow)$, tanto para expresiones de nodo como para expresiones de camino. Nuestro objetivo principal es dar *teoremas de definibilidad* para estas lógicas: por un lado, condiciones necesarias y suficientes bajo las cuales podemos asegurar qué clases de pointed data trees pueden ser definidas mediante el uso de una única expresión de nodo o un conjunto de ellas; por otro lado, condiciones necesarias y suficientes bajo las cuales clases de two-pointed data trees resultan definibles por una única expresión de camino o un conjunto de ellas. Como consecuencia de estos resultados, obtenemos *teoremas de separación*, los cuales indican condiciones necesarias y suficientes para que dos clases de pointed (o two-pointed) data trees sean separables por una tercera clase que a su vez es definible por una única expresión de nodo (respectivamente, de camino) o un conjunto de ellas.

Aunque nuestra investigación en $\text{XPath}_{=}$ toma como motivación la relevancia actual de documentos XML (los cuales por supuesto son finitos) y las lógicas para razonar sobre ellos, no nos restringimos al caso finito. Efectivamente, un conjunto infinito de expresiones de nodo o camino puede forzar a todos sus modelos a ser infinitos. Por lo tanto, ya que buscamos trabajar con conjuntos arbitrarios de expresiones de nodo o camino, debemos considerar árboles de tamaño arbitrario (i.e. finito o infinito).

En el contexto de BML, los teoremas de definibilidad usan dos herramientas básicas: ultraproductos y bisimulaciones. Como un primer paso para nuestra adaptación de estos teoremas a XPath$_=$, necesitamos modificar el concepto de ultraproducto para que su aplicación permanezca en el universo de data trees. Y aunque la noción de bisimulación ya fue desarrollada y estudiada para el caso de pointed data trees, necesitamos desarrollar nociones apropiadas de *bisimulación binaria* para el caso de two-pointed data trees y expresiones de camino de XPath$_=(\downarrow)$ y XPath$_=(\uparrow\downarrow)$, que capturen la noción de indistinguibilidad lógica en los fragmentos respectivos y sobre árboles de ramificación finita. Nuestras definiciones de bisimulación binaria son más complejas que aquellas de bisimulación unaria, y de hecho la noción de bisimulación binaria subsume a aquella de bisimulación unaria.

Para este marco binario, también demostramos un teorema de caracterización al estilo van Benthem que es análogo al de bisimulación unaria, mostrando que una fórmula de primer orden con dos variables libres es expresable en XPath$_=(\downarrow)$ si y solo si es invariante por bisimulaciones binarias y representa una 'forward property'. Así como en el caso unario, la caracterización falla para XPath$_=(\uparrow\downarrow)$.

Los resultados de este capítulo pueden encontrarse publicados en [2] y, más completamente, en [5].

Axiomatizaciones

En este capítulo desarrollamos la teoría de prueba de XPath$_=(\downarrow)$, diseñando un sistema axiomático ecuacional que solo prueba verdades semánticas de XPath$_=(\downarrow)$, y tal que todas esas verdades de XPath$_=(\downarrow)$ pueden ser probadas con el sistema. Esto es, obtenemos una axiomatización *correcta* y *completa* de XPath$_=(\downarrow)$. El estudio de axiomatizaciones completas nos puede proporcionar un método alternativo para resolver el problema de validez, el cual es indecidible para la lógica entera Core-Data-XPath [54], pero es decidible para algunos fragmentos, como ser cuando el único eje presente en el lenguaje es el de 'hijo' [42]. Adicionalmente, obtener una axiomatización completa tiene aplicaciones en la optimización de consultas a través de reescritura de queries. La idea principal es mirar a los axiomas de equivalencia (que son de la forma $\varphi \equiv \psi$) como reglas de reescritura de queries; en este contexto, la completitud del sistema axiomático significa que una equivalencia semántica entre dos expresiones de nodo o camino debe tener una correspondiente cadena de reglas de reescritura que permite transformar a la primera expresión en la segunda. Por lo tanto, obtener una axiomatización de XPath$_=(\downarrow)$, junto con todas las pruebas involucradas en la demostración de su completitud, tiene potencial de utilización como un primer paso para encontrar estrategias efectivas para reescribir queries de XPath a formas que sean equivalentes pero a su vez menos complejas.

Para demostrar correctitud y completitud para axiomatizaciones al estilo Hilbert, uno quiere ver que una fórmula puede ser probada en el sistema axiomático si y solo si es válida (eso es, resulta satisfecha en todo modelo posible). Más relevante para el marco que elegimos para este capítulo, cuando uno usa las reglas de inferencia de la lógica ecuacional uno quiere ver que una equivalencia entre dos fórmulas puede ser probada (notado $\vdash \varphi \equiv \psi$)

si y solo si el valor de verdad de ambas fórmulas coincide sobre todos los modelos (notado $\models \varphi \equiv \psi$). Por la (finite) tree model property de BML, la validez de una fórmula con respecto a la clase de todos los modelos de Kripke es equivalente a la validez sobre la clase de modelos de Kripke con estructura de árbol (finito). Como existen traducciones preservadoras de verdad desde y hacia la lógica ciega a los valores Core-XPath, no es sorprendente que existan axiomatizaciones del fragmento de expresiones de nodo de Core-XPath con 'hijo' como el único operador de accesibilidad. Es más, también existen axiomatizaciones (ecuacionales) para todos los fragmentos de Core-XPath con un único eje (aquellos donde la única relación de accesibilidad es 'hijo', 'descendiente', 'hermano', o etcetera), y también para todo el lenguaje Core-XPath entero [102].

Para el caso de $\mathrm{XPath}_{=}$, que puede expresar comparación de datos, el parecido con lenguajes modales es ahora más distante, ya que, como indicamos brevemente en §I.1.2, los modelos de $\mathrm{XPath}_{=}$ no pueden ser representados por modelos de Kripke. Efectivamente, encontrar una axiomatización en este caso se vuelve más complejo que para el caso puramente de navegación. Nuestro procedimiento involucra el diseño de un teorema de forma normal que muestra que dentro de nuestro sistema axiomático ecuacional todas las expresiones de nodo (o camino) consistentes se pueden probar equivalentes a la disyunción (unión) de expresiones de nodo (camino) en forma normal. Después damos un método para construir, para toda expresión de nodo consistente, un árbol finito con datos donde la expresión se satisface en la raíz. La construcción de este árbol es bastante intrincada, así que comenzamos dando una axiomatización para un caso más simple, el de un fragmento sintáctico de $\mathrm{XPath}_{=}(\downarrow)$ al cual llamamos $\mathrm{XPath}_{=}(\downarrow)^{-}$.

Los resultados de este capítulo pueden encontrarse publicados en [3].

Bisimulaciones en grafos con datos

En este capítulo, nos trasladamos desde el dominio de data trees potencialmente infinitos hacia el estudio de bisimulaciones en el dominio de *finite* data *graphs*. Nuestro foco principal es el de calcular la complejidad algorítmica de encontrar bisimilaridades, de modo que nuestra restricción a estructuras finitas es algo natural. Respecto a la expansión hacia grafos, está parcialmente motivada por modelos de datos que se han vuelto cada vez más importantes con el crecimiento continuo de la Web y de aplicaciones relacionadas con Internet. Es cierto que, por un lado, la información accesible en la Web usualmente es guardada en estructuras jerárquicas, como ser el formato XML, que pueden ser modeladas como bases de datos estructuradas con árboles. Pero por otro lado, vastas cantidades de información están asociadas a nuevas aplicaciones cuyo modelo de datos subyacente está descrito por bases de datos estructuradas con grafos (finitos), como ser en los casos de redes sociales, la Web Semántica, sistemas biológicos, tareas de análisis de redes, o aplicaciones de detección de crímenes.

Las bases de datos semiestructuradas son usualmente vistas como árboles o grafos con etiquetas en las aristas, y los nodos pueden ser vistos como 'entidades', conteniendo los datos (por ejemplo, el nombre y dirección en una red social), mientras que las aristas

representan 'relaciones' entre estas entidades (como ser 'befriends' o 'likes'). Muchas de las aplicaciones que hacen uso de este modelo de datos tienen dos características en común: por un lado, el modelo de datos subyacente puede ser descrito por un grafo o un árbol, y, por otro lado, al consultar tales estructuras la *topología* del grafo es tan importante como los propios *datos*.

Cuando L es la lógica modal basica, la noción de indistinguibilidad es capturada por la relación de bisimulación [104], y las clases de equivalencia correspondientes a esta relación pueden ser computadas de forma eficiente [8]. Consultar bases de datos sobre grafos, en general, requiere la habilidad de testear propiedades relativas a la topología. Un lenguaje de consulta básico que puede testear este tipo de propiedades es RPQ (Regular Path Query), el cual selecciona nodos conectados por un camino que es descrito por un lenguaje regular sobre el alfabeto de las etiquetas [26]. Extensiones de este lenguaje de consultas básico, como ser Propositional Dynamic Logic, tienen una noción de bisimulación similar a la de la lógica modal básica y están por lo tanto sujetas a una computación eficiente de la relación de indistinguibilidad.

Sin embargo, en varios escenarios, estos lenguajes de consulta se quedan cortos en poder expresivo, ya que los datos contenidos en sus nodos desaparecen en sus representaciones abstractas. Una forma estándar de añadir datos es a través del uso de una lógica como XPath. XPath fue concebido originalmente para seleccionar nodos de documentos XML (que esencialmente son árboles), pero su simplicidad y comportamiento modal se adaptan perfectamente a grafos con datos, y efectivamente ya ha sido estudiado [78] y usado [22] en este escenario.

En este capítulo expandemos nuestro foco desde el universo de árboles con datos al universo de grafos con datos. También hacemos una transición desde nuestra lógica $\mathrm{XPath}_{=}(\downarrow)$ con etiquetas en los nodos a la lógica $\mathrm{XPath}_{=}(\downarrow_{\mathbf{a}})$ con etiquetas en las aristas, pero destacamos que se puede traducir entre ambos formalismos, y por lo tanto la elección de etiquetas en los nodos o en las aristas no es esencial. Mostramos que resultados previos para $\mathrm{XPath}_{=}(\downarrow)$ sobre árboles con datos y etiquetas en los nodos se extienden a $\mathrm{XPath}_{=}(\downarrow_{\mathbf{a}})$ sobre grafos con datos y etiquetas en las aristas. Estudiamos la complejidad computacional de la noción de bisimulación de $\mathrm{XPath}_{=}$ sobre bases de datos (finitas) semiestructuradas; a lo largo de este capítulo, nos restringimos al dominio de grafos *finitos* con datos. Para que este estudio sea completo, variamos, por un lado, los tipos de estructuras finitas que analizamos: grafo, árbol, o DAG (directed acyclic graph). Por otro lado, estudiamos dos modalidades diferentes que puede tener la lógica, enfocándonos en el fragmento $\mathrm{XPath}_{=}(\downarrow_{\mathbf{a}})$ y luego expandiendo nuestros resultados a $\mathrm{XPath}_{=}(\uparrow^{\mathbf{a}}\downarrow_{\mathbf{a}})$, que añade la posibilidad de realizar navegación inversa. Finalmente, también consideramos algunas restricciones sintácticas sobre las fórmulas, que resultan en mejores resultados computacionales.

Los resultados de este capítulo pueden encontrarse publicados en [1].

Lógicas de valores repetidos sobre árboles con datos

En este capítulo trabajamos con una definición extendida de data trees, donde permitimos que cada nodo lleve no solo un único valor de datos, sino una colección finita de ellos (ordenada de cierta manera). Esta estructura de (multi)data trees ha sido considerada en el ámbito de datos semiestructurados como otra abstracción de documentos XML, pero también de automatas temporizados, para verificación de programas, y en general en sistemas para manipular valores de datos. Encontrar lógicas decidibles o modelos de autómatas sobre data trees es un objetivo importante a la hora de estudiar sistemas enfocados en datos.

Hay una gran abundancia de formalismos de especificación sobre estas estructuras (ya sea data trees o su versión de 'palabras', data words), con orígenes en automatas [88, 99], lógica de primer orden [17, 65, 48, 19], XPath [66, 50, 44, 43, 47], o lógicas temporales [38, 79, 70, 46, 36, 67]. En su mayor generalidad, estos formalismos suelen llevar a problemas indecidibles, y un tópico de investigación muy conocido es aquel que busca un buen balance entre expresividad y decibilidad.

Resultados llamativos y sorprendentes han sido exhibidos sobre la relación entre lógicas para data trees y counter automata [65, 50, 66], indicando que las lógicas para data trees no son solo interesantes por sí mismas, sino también por sus profundas conexiones con sistemas con contadores.

En este capítulo estudiamos el mecanismo básico de "repetición de datos" que es común a muchas lógicas sobre data trees. Para esto, investigamos una lógica básica que puede navegar la estructura del árbol a través del uso de modalidades como las de CTL (computation tree logic), y que además puede hacer "tests de datos" al preguntar si un valor de dato es repetido en un subárbol. Más concretamente, los data tests son fórmulas del tipo $u \approx \mathrm{EF}v$, lo cual indica que el valor almacenado en el atributo u del nodo actual es igual al valor almacenado en el atributo v de algún descendiente. Este tipo de *lógica de valores que se repiten*, o LRV (logic of repeating values), ha sido el centro de una línea de investigación sobre data words estudiada en [35, 36], mostrando ajustadas correspondencias entre los problemas de satisfacibilidad y los de accesibilidad (reachability) para Vector Addition Systems. El capítulo actual extiende esta investigación, exhibiendo conexiones entre el problema de satisfabilidad de LRV *sobre data trees* y el problema de coverability para branching counter systems. Con el propósito de obtener conexiones con branching Vector Addition Systems with States (branching VASS) [105], también introducimos una restricción en la cual los data tests están limitados a usar una sola variable, esto es, son de la forma $v \approx \mathrm{EF}v$. Denotamos esta restricción con $\mathrm{LRV}^{\mathrm{D}}$. Esta relación simbiótica entre sistemas con contadores y lógicas nos conduce a considerar algunas extensiones naturales de tanto la lógica como de los branching counter systems. Particularmente, introducimos un nuevo modelo de branching counter system que es de interés independiente, y tiene problemas decidibles de coverability y de control-state reachability.

Los resultados de este capítulo pueden encontrarse publicados en [4].

Bibliography

> Naresh had once told us that, to men like himself, physical books were like trophies of slain animals and coats of arms rolled into one.
>
> *Crystal Society*
> Max Harms

> We report a method for estimating what percentage of people who cited a paper had actually read it [...] Our estimate is that only about 20% of citers read the original.
>
> *Read Before You Cite![100]*
> Simkin and Roychowdhury

[1] S. Abriola, P. Barceló, D. Figueira, and S. Figueira. Bisimulations on data graphs. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR*, pages 309–318, 2016.

[2] S. Abriola, M. E. Descotte, and S. Figueira. Definability for downward and vertical XPath on data trees. In *21th Workshop on Logic, Language, Information and Computation*, volume 6642 of *Lecture Notes in Computer Science*, pages 20–34, 2014.

[3] S. Abriola, M. Descotte, R. Fervari, and S. Figueira. Axiomatizations for downward XPath on Data Trees. Submitted.

[4] S. Abriola, D. Figueira, and S. Figueira. Logics of repeating values on data trees and branching counter systems. In *FoSSaCS 2017: 20th International Conference on Foundations of Software Science and Computation Structures*, 2017.

[5] S. Abriola, M. E. Descotte, and S. Figueira. Model theory of XPath on data trees. Part II: Binary bisimulation and definability. *Information and Computation*. In press, http://www.glyc.dc.uba.ar/santiago/papers/xpath-part2.pdf.

[6] C. Areces, F. Carreiro, and S. Figueira. Characterization, definability and separation via saturated models. *Theoretical Computer Science*, 537:72–86, 2014.

[7] C. Areces, A. Koller, and K. Striegnitz. Referring expressions as formulas of description logic. In *Proc. of the 5th INLG*, Salt Fork, OH, USA, 2008.

[8] C. Areces, S. Figueira, and D. Gorín. Using logic in the generation of referring expressions. In *Logical Aspects of Computational Linguistics*, pages 17–32. Springer, 2011.

[9] S. N. Artëmov and V. Krupski. Data storage interpretation of labeled modal logic. *Annals of Pure and Applied Logic*, 78(1-3):57–71, 1996.

[10] D. Baelde, S. Lunel, and S. Schmitz. A sequent calculus for a modal logic on finite data trees. In *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 - September 1, 2016, Marseille, France*, pages 32:1–32:16, 2016.

[11] C. Baier and J. Katoen. *Principles of model checking*. MIT Press, 2008.

[12] J. Balcázar, J. Gabarro, and M. Santha. Deciding bisimilarity is P-complete. *Formal aspects of computing*, 4(1):638–648, 1992.

[13] M. Benedikt, W. Fan, and G. M. Kuper. Structural properties of XPath fragments. *Theoretical Computer Science*, 336(1):3–31, 2005.

[14] P. Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.

[15] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.

[16] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.

[17] M. Bojańczyk, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data trees and XML reasoning. *Journal of the ACM*, 56(3):1–48, 2009.

[18] M. Bojańczyk, C. David, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data words. *ACM Trans. Comput. Log.*, 2010.

[19] B. Bollig, A. Cyriac, P. Gastin, and K. N. Kumar. Model checking languages of data words. In *Foundations of Software Science and Computational Structures*, pages 391–405. Springer, 2012.

[20] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Containment of conjunctive regular path queries with inverse. In *KR*, pages 176–185, 2000.

[21] L. Cardelli and G. Ghelli. TQL: a query language for semistructured data based on the ambient logic. *Mathematical Structures in Computer Science*, 14(3):285–327, 2004.

[22] S. Cassidy. Generalizing XPath for directed graphs. In *Extreme Markup Languages*, 2003.

[23] C. Chang and H. Keisler. *Model theory*. Studies in logic and the foundations of mathematics. North-Holland, 1990.

[24] J. Clark and S. DeRose. XML path language (XPath). Website, 1999. W3C Recommendation. http://www.w3.org/TR/xpath.

[25] E. M. Clarke, O. Grumberg, and D. Peled. *Model checking*. MIT Press, 2001.

[26] I. F. Cruz, A. O. Mendelzon, and P. T. Wood. A graphical query language supporting recursion. In *ACM SIGMOD Record*, volume 16, pages 323–330. ACM, 1987.

[27] R. Dale and E. Reiter. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science*, 19, 1995.

[28] V. Dalmau, P. G. Kolaitis, and M. Y. Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In *CP*, pages 310–326, 2002.

[29] A. Dawar and M. Otto. Modal characterisation theorems over special classes of frames. *Annals of Pure and Applied Logic*, 161(1):1–42, 2009.

[30] M. de Rijke. Modal model theory. Technical Report CS-R9517, CWI, Amsterdam, 1995.

[31] M. de Rijke and H. Sturm. Global definability in basic modal logic. *Essays on non-classical logic*, 1:111–135, 2001.

[32] R. Dechter. From local to global consistency. *Artif. Intell.*, 55(1):87–108, 1992.

[33] R. Dechter. *Constraint processing*. Elsevier Morgan Kaufmann, 2003.

[34] S. Demri, D. D'Souza, and R. Gascon. Decidable temporal logic with repeating values. In *Symposium on Logical Foundations of Computer Science*, volume 4514 of *LNCS*, pages 180–194. Springer, 2007.

[35] S. Demri, D. D'Souza, and R. Gascon. Temporal logics of repeating values. *J. Log. Comput.*, 22(5):1059–1096, 2012.

[36] S. Demri, D. Figueira, and M. Praveen. Reasoning about data repetitions with counter systems. In *LICS*, pages 33–42. IEEE Press, 2013.

[37] S. Demri, M. Jurdziński, O. Lachish, and R. Lazić. The covering and boundedness problems for branching vector addition systems. *J. Comput. Syst. Sci.*, 79(1):23–38, 2013.

[38] S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3), 2009.

[39] A. Dovier, C. Piazza, and A. Policriti. An efficient algorithm for computing bisimulation equivalence. *Theor. Comput. Sci*, 311:221–256, 2004.

[40] E. A. Emerson and J. Y. Halpern. sometimes and not never revisited: on branching versus linear time temporal logic. *Journal of the ACM (JACM)*, 33(1):151–178, 1986.

[41] W. Fan, J. Li, X. Wang, and Y. Wu. Query preserving graph compression. In *SIGMOD*, pages 157–168, 2012.

[42] D. Figueira. Decidability of downward XPath. *ACM Transactions on Computational Logic*, 13(4):34, 2012.

[43] D. Figueira. On XPath with transitive axes and data tests. In W. Fan, editor, *Proceedings of the 31st Annual ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'13)*, pages 249–260, New York, NY, USA, June 2013. ACM Press.

[44] D. Figueira, S. Figueira, and C. Areces. Basic model theory of XPath on data trees. In *International Conference on Database Theory*, pages 50–60, 2014.

[45] D. Figueira, S. Figueira, and C. Areces. Model theory of XPath on data trees. Part I: Bisimulation and characterization. *Journal of Artificial Intelligence Research*, 53:271–314, 2015.

[46] D. Figueira. Forward-XPath and extended register automata on data-trees. In *ICDT*. ACM, 2010.

[47] D. Figueira. Decidability of downward XPath. *ACM Trans. Comput. Log.*, 13(4), 2012.

[48] D. Figueira and L. Libkin. Pattern logics and auxiliary relations. In *LICS*, pages 40:1–40:10, 2014.

[49] D. Figueira and L. Segoufin. Future-looking logics on data words and trees. In *International Symposium on Mathematical Foundations of Computer Science*, pages 331–343. Springer, 2009.

[50] D. Figueira and L. Segoufin. Bottom-up automata on data trees and vertical XPath. In *STACS*, volume 9 of *LIPIcs*, pages 93–104. LZI, 2011.

[51] S. Figueira and D. Gorín. On the size of shortest modal descriptions. In *Advances in Modal Logic*, volume 8, pages 114–132, 2010.

[52] G. Fletcher, M. Gyssens, D. Leinders, J. Van den Bussche, D. Van Gucht, and S. Vansummeren. Similarity and bisimilarity notions appropriate for characterizing indistinguishability in fragments of the calculus of relations. *Journal of Logic and Computation*, 25(3):549–580, 2014. Published online.

[53] M. Forti and F. Honsell. Set theory with free construction principles. *Annali Scuola Normale Superiore, Pisa*, X(3):493–522, 1983.

[54] F. Geerts and W. Fan. Satisfiability of XPath queries with sibling axes. In *Database Programming Languages, 10th International Symposium, DBPL 2005, Trondheim, Norway, August 28-29, 2005, Revised Selected Papers*, pages 122–137, 2005.

[55] L. Getoor and C. P. Diehl. Link mining: a survey. volume 7, pages 3–12. ACM, 2005.

[56] R. Givan, T. L. Dean, and M. Greig. Equivalence notions and model minimization in Markov decision processes. *Artif. Intell.*, 147(1-2):163–223, 2003.

[57] G. Gottlob, C. Koch, and R. Pichler. Efficient algorithms for processing XPath queries. *ACM Trans. Database Syst.*, 30(2):444–491, 2005.

[58] A. Grinberg. Algoritmos incrementales de actualización para aproximaciones de bisimulación en XPath con datos. MSc thesis, Universidad de Buenos Aires, Argentina, 2016.

[59] M. Gyssens, J. Paredaens, D. Van Gucht, and G. Fletcher. Structural characterizations of the semantics of XPath as navigation tool on a document. In *PODS*, pages 318–327. ACM, 2006.

[60] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, Cambridge, MA, 2000.

[61] M. R. Henzinger, T. A. Henzinger, and P. W. Kopke. Computing simulations on finite and infinite graphs. In *Proc. of 36th Annual Symposium on Foundations of Computer Science*, pages 453–462. IEEE Computer Society Press, 1995.

[62] J. Hopcroft. *An $n\log(n)$ algorithm for minimizing states in a finite automaton*. In Z. Kohave, editor, *Theory of Machines and Computations*. Academic Press, 1971.

[63] E. V. Huntington. Boolean algebra. A correction to: New sets of independent postulates for the algebra of logic, with special reference to Whitehead and Russells Principia mathematica. *Transactions of the American Mathematical Society*, 35(2):557–558, 1933.

[64] E. V. Huntington. New sets of independent postulates for the algebra of logic, with special reference to Whitehead and Russells Principia Mathematica. *Transactions of the American Mathematical Society*, 35(1):274–304, 1933.

[65] F. Jacquemard, L. Segoufin, and J. Dimino. $\mathrm{FO}^2(<, +1, \sim)$ on data trees, data tree automata and branching vector addition systems. *arXiv preprint arXiv:1601.01579*, 2016.

[66] M. Jurdziński and R. Lazić. Alternating automata on data trees and XPath satisfiability. *ACM Trans. Comput. Log.*, 12(3):19, 2011.

[67] A. Kara, T. Schwentick, and T. Zeume. Temporal logics on words with multiple data values. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, 2010.

[68] P. G. Kolaitis and M. Y. Vardi. A game-theoretic approach to constraint satisfaction. In *AAAI*, pages 175–181, 2000.

[69] E. Krahmer, S. van Erk, and A. Verleg. Graph-based generation of referring expressions. *Computational Linguistics*, 29(1), 2003.

[70] O. Kupferman and M. Vardi. Memoryful Branching-Time Logic. In *LICS'06*, pages 265–274. IEEE, 2006.

[71] O. Kupferman and M. Y. Vardi. Verification of fair transition systems. *Chicago J. Theor. Comput. Sci.*, 1998, 1998.

[72] N. Kurtonina and M. de Rijke. Bisimulations for temporal logic. *Journal of Logic, Language and Information*, 6:403–425, 1997.

[73] N. Kurtonina and M. de Rijke. Simulating without negation. *Journal of Logic and Computation*, 7:503–524, 1997.

[74] N. Kurtonina and M. de Rijke. Expressiveness of concept expressions in first-order description logics. *Artif. Intell.*, 107(2):303–333, 1999.

[75] R. E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM journal on computing*, 6(3):467–480, 1977.

[76] R. Lazić and S. Schmitz. Nonelementary complexities for branching VASS, MELL, and extensions. *ACM Transactions on Computational Logic (TOCL)*, 16(3):20, 2015.

[77] L. Libkin and D. Vrgoč. Regular path queries on graphs with data. In *International Conference on Database Theory*, pages 74–85, 2012.

[78] L. Libkin, W. Martens, and D. Vrgoč. Querying graph databases with XPath. In *ICDT*, pages 129–140. ACM, 2013.

[79] A. Lisitsa and I. Potapov. Temporal logic with predicate λ-abstraction. In *TIME'05*, pages 147–155. IEEE, 2005.

[80] Y. Luo, G. H. L. Fletcher, J. Hidders, P. D. Bra, and Y. Wu. Regularities and dynamics in bisimulation reductions of big graphs. In *GRADES 2013*, page 13, 2013.

[81] Y. Luo, G. H. L. Fletcher, J. Hidders, Y. Wu, and P. D. Bra. External memory k-bisimulation reduction of big graphs. In *22nd ACM CIKM'13*, pages 919–928, 2013.

[82] M. Marx and M. de Rijke. Semantic characterizations of navigational XPath. *SIGMOD Record*, 34(2):41–46, 2005.

[83] A. R. Meyer and L. J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *SWAT (FOCS)*, pages 125–129, 1972.

[84] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer, 1980.

[85] R. Milner. An algebraic definition of simulation between programs. In *Proceedings of the 2nd International Joint Conference on Artificial Intelligence. London, UK, September 1971.*, pages 481–489, 1971.

[86] R. Milner. *Communicating and mobile systems - the Pi-calculus*. Cambridge University Press, 1999.

[87] T. Milo and D. Suciu. Index structures for path expressions. In *ICDT*, pages 277–295, 1999.

[88] F. Neven, T. Schwentick, and V. Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Log.*, 5(3):403–435, 2004.

[89] M. Otto. Elementary proof of the van Benthem-Rosen characterisation theorem. Technical Report 2342, Fachbereich Mathematik, Technische Universität Darmstadt, 2004.

[90] M. Otto. Bisimulation invariance and finite models. In *Logic Colloquium'02*, volume 27 of *Lecture Notes in Logic*, pages 276–298, 2006.

[91] R. Paige and R. Tarjan. Three partition refinement algorithms. *SIAM J. Comput.*, 16(6):973–989, 1987.

[92] D. Park. Concurrency and Automata on Infinite Sequences. In *Theoret. Comput. Sci.*, volume 104 of *LNCS*, pages 167–183. Springer, 1981.

[93] A. Pnueli. The temporal logic of programs. In *Foundations of Computer Science, 1977., 18th Annual Symposium on*, pages 46–57. IEEE, 1977.

[94] C. Rackoff. The covering and boundedness problems for vector addition systems. *Theoret. Comput. Sci.*, 6(2):223–231, 1978.

[95] E. Rosen. Modal logic over finite structures. *Journal of Logic, Language and Information*, 6(4):427–439, 1997.

[96] D. Sangiorgi. On the origins of bisimulation and coinduction. *ACM Trans. Program. Lang. Syst.*, 31(4):1–41, 2009.

[97] D. Sangiorgi. On the origins of bisimulation and coinduction. *ACM Transactions on Programming Languages and Systems*, 31(4), 2009.

[98] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970.

[99] L. Segoufin. Automata and logics for words and trees over an infinite alphabet. In *CSL*, pages 41–57. Springer, 2006.

[100] M. V. Simkin and V. P. Roychowdhury. Read before you cite! *arXiv preprint cond-mat/0212043*, 2002.

[101] B. ten Cate. The expressivity of XPath with transitive closure. In S. Vansummeren, editor, *PODS*, pages 328–337. ACM, 2006.

[102] B. ten Cate, T. Litak, and M. Marx. Complete axiomatizations for XPath fragments. *Journal of Applied Logic*, 8(2):153–172, 2010.

[103] B. ten Cate and M. Marx. Axiomatizing the logical core of XPath 2.0. *Theory of Computing Systems*, 44(4):561–589, 2009.

[104] J. van Benthem. *Modal Correspondence Theory*. PhD thesis, Universiteit van Amsterdam, 1976.

[105] K. N. Verma and J. Goubault-Larrecq. Karp-Miller trees for a branching extension of VASS. *Discrete Mathematics & Theoretical Computer Science*, 7(1):217–230, 2005.

Index of notation

Index