Bisimulations on Data Graphs*

Sergio Abriola
Dept. Computación
Universidad de Buenos Aires

Pablo Barceló
Center for Semantic Web Research &
Dept. Computer Science
University of Chile

Diego Figueira
CNRS, LaBRI, France

Santiago Figueira
Dept. Computación
Universidad de Buenos Aires
& CONICET, Argentina

Abstract

Bisimulation provides structural conditions to characterize indistinguishability between nodes on graph-like structures from an external observer. It is a fundamental notion used in many areas. However, many applications use graphs where nodes have data, and where observers can test for equality or inequality of data values (e.g., asking the attribute 'name' of a node to be different from that of all its neighbors).

The present work constitutes a first investigation of "data aware" bisimulations on data graphs. We study the problem of computing such bisimulations, based on the observational indistinguishability for XPath —a language that extends modal logic with tests for data equality. We show that in general the problem is PSPACE-complete, but identify several restrictions that yield better complexity bounds (CO-NP, PTIME) by controlling suitable parameters of the problem; namely, the amount of *non-locality* allowed, and the class of models considered (graph, DAG, tree). In particular, this analysis yields a hierarchy of tractable fragments.

1 Introduction

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 (Sangiorgi 2009) for a thorough historical revision of the notion of bisimulation. In both contexts, bisimulation (and its "half" 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 by Milner (and refined by Park) in the context of concurrency theory as a way to study the behavior of programs (Milner 1971; Park 1981). 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 (van Benthem 1976).

*This work was partially supported by the Laboratoire International Associé "INFINIS" and by grant ANPCyT-PICT-2013-2011. Barceló is funded by the Millenium Nucleus Center for Semantic Web Research under grant NC120004.

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 (Milner 1999); in model checking to tackle the state-explosion problem (Clarke, Grumberg, and Peled 2001); in databases as a method for indexing and compressing semi-structured data (Milo and Suciu 1999; Fan et al. 2012); in stochastic planning to solve Markov decision processes efficiently (Givan, Dean, and Greig 2003); in description logics to understand the expressiveness of some languages (Kurtonina and de Rijke 1999); and in natural language generation to define semantic counterparts to the notion of referring expression (Areces, Figueira, and Gorín 2011). Also, in constraint satisfaction the closely related notion of arc consistency is used as an approximation of satisfiability (Dechter 1992; 2003) and as a method for finding tractable instances of SAT (Kolaitis and Vardi 2000; Dalmau, Kolaitis, and Vardi 2002).

Let us quickly recall the notion of bisimulation for the basic modal logic (ML). Let $\mathcal{G} = \langle G, E\rangle$ and $\mathcal{G}' = \langle G', E'\rangle$ be two directed graphs whose edges are labeled over a finite alphabet Σ. A bisimulation between $\mathcal{G}$ and $\mathcal{G}'$ is a relation $Z \subseteq G \times G'$ such that:

- **(Zig)** For each pair $(u, u') \in Z$, if $\mathcal{G}$ contains an edge $u \xrightarrow{a} v$ (for $a \in \Sigma$) then $\mathcal{G}'$ contains an edge $u' \xrightarrow{a} v'$ such that $(v, v') \in Z$.
- **(Zag)** For each pair $(u, u') \in Z$, if $\mathcal{G}'$ contains an edge $u' \xrightarrow{a} v'$ (for $a \in \Sigma$) then $\mathcal{G}$ contains an edge $u \xrightarrow{a} v$ such that $(v, v') \in Z$.

The nodes u in $\mathcal{G}$ and u' in $\mathcal{G}'$ are *bisimilar* if there is a bisimulation between $\mathcal{G}$ and $\mathcal{G}'$ that contains the pair (u, u').

The following are two important properties of this notion:

- First, bisimulation can be restated in terms of greatest fixed points, which in turn yields a simple polynomial time algorithm for checking if u and u' are bisimilar (more specifically, for computing the *maximal* bisimulation between $\mathcal{G}$ and $\mathcal{G}'$).
- Second, the notion of bisimulation captures, in a precise sense, the expressiveness of ML on finite models. Formally, Hennesy-Milner's Theorem establishes that nodes u and u' are bisimilar if and only if they cannot be distinguished by ML formulas (see, e.g., (Blackburn, de Rijke, and Venema 2001)). This result is robust, as it continues

to hold if we replace ML by more expressive navigational logics used in the analysis of programs (e.g., PDL (Fischer and Ladner 1979)) and model checking (e.g., CTL* (Clarke, Grumberg, and Peled 2001)).

Data-awareness. An important feature of bisimulations is that they are defined in terms of the *topology* of the graph structure, i.e., the way in which nodes are linked by labeled edges. This is good enough for applications on which this topology of labeled edges is the only relevant feature in their model. However, it is not sufficient for other applications that impose higher demands on such models and query languages. We are thinking here, in particular, of "data-aware" models such as *data* or *property graphs*, which have become *de-facto* standard in the area of graph databases (see, e.g., (Angles and Gutiérrez 2008; Robinson, Webber, and Eifrem 2013; Libkin and Vrgoč 2012)), or XML documents (i.e., data trees). In addition to the topology defined by labeled edges, such graph-based models allow nodes to be *attributed*, i.e., to be associated with a set of property/value pairs, and languages over these models are endowed with the capability of testing for equality of such data values.

Example 1. Consider a data graph representation of a movie database, in which (a) nodes represent actors, directors, and movies, (b) edges establish relationships between such nodes, e.g. movie m is directed by director d and casts actor a, and (c) nodes contain attributes, such as the name of the actor and its age, or the title of the movie, its duration, and the company who produced it. □

An important feature of the query languages for data graphs is that they combine topology and data to express relevant properties. An example is the query which asks whether a director has two movies produced by different companies. This query cannot be expressed in a purely navigational language such as ML, PDL or CTL* (simply because they cannot compare the attribute values of two nodes), but can in turn be expressed in the "data-aware" language XPath$_=$ (Libkin, Martens, and Vrgoč 2013). This language extends the navigational core of XPath with data comparison formulas of the form $\langle \alpha_1 = \alpha_2 \rangle$ and $\langle \alpha_1 \neq \alpha_2 \rangle$. Intuitively, when evaluated on a node u these formulas ask whether there are paths π_1 and π_2 starting in u such that π_i satisfies the condition given by path expression α_i (for $i = 1, 2$) and the final nodes of π_1 and π_2 have the same (resp., different) data value (we assume for the sake of simplicity that each node is attributed with a single data value, given by function $data$).

As it has been recently shown in the context of XML/data trees, XPath$_=$ allows for a Hennessy-Milner's style characterization in terms of a natural class of "data-aware" bisimulations (Figueira, Figueira, and Areces 2015). We notice in this article that such characterization extends in a straightforward way to the class of data graphs. Let us explain intuitively how such "data-aware" bisimulation Z (called XPath$_=$-bisimulation in the paper) between data graphs $\mathcal{G} = \langle G, E, data \rangle$ and $\mathcal{G}' = \langle G', E', data' \rangle$ is defined. The **(Zig$_=$)** property establishes that for each pair (u, u') in Z and paths π_1 and π_2 in $\mathcal{G}$ starting from u, there must be paths π'_1 and π'_2 in $\mathcal{G}'$ starting from u' such that:

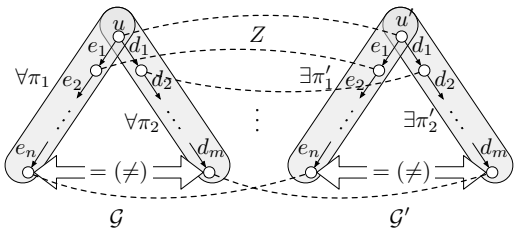

Figure 1: The **(Zig$_=$)** clause for XPath$_=$ on data graphs. In the picture, $(e_i)_{i \leq n}$ and $(d_j)_{j \leq m}$ are labels.

- **Topology-preserving property:** The labels of π_i and π'_i (for $i = 1, 2$) are the same and the jth node of π_i is in the Z-relation with the jth node of π'_i, for every j.
- **Data-awareness property:** If the data values of the final nodes of π_1 and π_2 are equal (resp., different), so is the case for the data values of the final nodes of π'_1 and π'_2.

This is graphically depicted in Figure 1. The **(Zag$_=$)** property is, of course, symmetric.

It is worth remarking here that, in general, languages for data graphs, such as XPath$_=$ and others —e.g., (Libkin and Vrgoč 2012; Bojańczyk et al. 2009; David et al. 2013; Figueira 2010)— allow to test for (in)equality of data values only, abstracting away the concrete data. This is because meaningful properties of the graph topology are naturally closed under renaming of data values through bijections. While the use of constants may be essential for data retrieval, from an observational perspective the infinite domain of data values is merely a source of unique names to relate nodes. This is why we work with languages and bisimulation notions that are closed under bijections of data values and, therefore, domain-independent.

The logics we study are related to description logics with concrete domains (Lutz 2003) —a family of modal logics designed for the representation of conceptual knowledge, equipped with means that allow to describe "concrete qualities" of real-world objects such as their weight, temperature, and spatial extension. One can draw a connection between, on the one hand, $\mathcal{ALC}$ when restricted to having binary predicates $=, \neq$ on a concrete domain, and, on the other hand, data-aware XPath without transitive axes on data graphs.

Potential applications. Data-aware bisimulations have been used to study the expressive power of XPath$_=$ on data trees. We foresee several other potential applications of them when interpreted over data graphs:

- **Indexing:** Finding bisimilar nodes over graph-structured data is the first step in many approaches to building indexing data structures for efficient evaluation of navigational languages (Milo and Suciu 1999; Fan et al. 2012). These approaches are based on the following idea: If x and y are bisimilar and x is in the output of a query, also y is in the output. Extending this to "data-aware" bisimulations might then serve as a building-block over which index structures for XPath$_=$ expressions can be constructed.

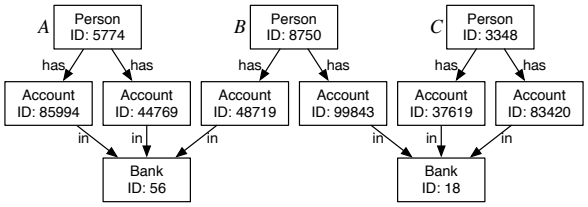

Figure 2: A scene with people, accounts and banks.

- **Clustering:** Another motivation stems from the task of *clustering* in graph data mining (Getoor and Diehl 2005), i.e.,, the division of data into groups of similar objects. One way to define similarity on data graphs is based on *observational indistinguishability*, that is, grouping together elements x, y that cannot be distinguished with a data aware logic L: $x \equiv_L y$. For the logic XPath$_=$ 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 paper).
- **Referring expressions generation:** A basic and active task in natural language generation is *referring expressions generation* (REG), 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 language that uniquely represents the element. Krahmer, van Erk, and Verleg (2003) propose to use labeled directed graphs for representing the scene, and Areces, Koller, and Striegnitz (2008) resort to description logics (DLs) as a formalism for representing a RE. Areces, Figueira, and Gorín (2011) show that this approach can be efficiently implemented using bisimulations.

  In some cases, though, a scene for the REG problem is better modeled as a data graph. Imagine, e.g., a scene modeling clients, accounts, and banks (Figure 2). Each object has an ID. Suppose we look for a RE for target B. It is impossible to distinguish nodes A and B using ML 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 XPath$_=$. Extending (Areces, Figueira, and Gorín 2011), "data-aware" bisimulations might then be an efficient tool for REG in cases when REs are expressed in the language of XPath$_=$.

Computing "data-aware" bisimulations. In any of the previous cases one is faced with the fundamental problem of determining whether two nodes are "data-aware" bisimilar (more in general, checking if there is a "data-aware" bisimulation relating two data graphs). Recall that this problem can be solved in PTIME for usual (i.e., purely topological) bisimulations. One of the reasons that explains this is that such bisimulations are *local*, in the sense that the **(Zig)** and **(Zag)** conditions for a pair (u, u') are defined in terms of nodes which are adjacent to u and u', resp. But this no longer holds for "data-aware" bisimulations, as the **(Zig$_=$)** and **(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 "data-aware" 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 (Kupferman and Vardi 1998). 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.

Contributions. 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 full generality, checking whether two data graphs are "data-aware" bisimilar is PSPACE-complete. This is obtained by showing that the problem is polynomially equivalent to *equivalence* of nondeterministic finite automata, which is PSPACE-complete (Meyer and Stockmeyer 1972). In particular, 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 XPath 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 (but does not restrict the navigational expressions of the form $\langle \alpha \rangle$).
- We then study how the underlying graphs affect the complexity of the problem (restriction (b) above), and look at the two most important classes of acyclic graphs: trees and DAGs. We show that checking "data-aware" bisimilarity is tractable for the former and CO-NP-complete for the latter.
- Finally, we look at *two-way* XPath$_=$, which allows to traverse edges in both directions. The problem of checking "data aware" bisimilarity in this context remains PSPACE-complete. The upper bound follows easily, but the lower bound needs 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.

Organization of the paper. We present basic notions in §2. XPath$_=$-bisimulations are introduced in §3. The complex-

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

Figure 3: Semantics of XPath$_=$ for a data graph $\mathcal{G} = \langle G, E, data\rangle$.

ity of XPath$_=$-bisimulations is studied in §4. Restrictions on paths are presented in §5, and those on data models in §6. The two-way version of XPath$_=$ is studied in §7, while §8 is devoted to conclusions.

2 Data graphs and XPath

Data graphs. As is customary in graph-structured data, we work with edge-labeled data graphs, i.e., finite graphs whose edges are labeled with an element of a finite alphabet $\mathbb{A}$ and whose nodes contain a single value of an infinite domain $\mathbb{D}$ (Libkin and Vrgoč 2012; Barceló 2013). Formally, a *data graph* $\mathcal{G}$ over $\mathbb{A}$ is a tuple $\langle G, E, data\rangle$, where G is a finite set of nodes, $E \subseteq G \times \mathbb{A} \times G$, and $data : G \to \mathbb{D}$ assigns values to nodes. Intuitively, an edge $(x,a,y) \in E$ (for x, y nodes in G and a a symbol in $\mathbb{A}$) represents that there is an a-labeled edge from x to y. Also, $data(x) = d$ iff the data value of node x is d.

By convention, the set of nodes of a data graph $\mathcal{G}$ will be denoted by G, the set of nodes of a data graph $\mathcal{G}'$ by G', and so on. When E is clear from the context, we write $x \xrightarrow{a} y$ instead of $(x,a,y) \in E$.

XPath. We work with the language XPath$_=$, a simplification of XPath, stripped of its syntactic sugar, and adapted to reasoning on data graphs (see, e.g., (Libkin, Martens, and Vrgoč 2013)). XPath$_=$ is a two-sorted language, with *path expressions* (denoted α, β, γ) representing binary relations on nodes, and *node expressions* (denoted φ, ψ, η) representing unary relations, or properties. Its syntax is defined by mutual recursion as follows:

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

A node expression is *positive* if it contains no negation.

We formally define the semantics of XPath$_=$ in Figure 3. For a data graph $\mathcal{G}$ and $u \in G$, we write $\mathcal{G}, u \models \varphi$ to denote $u \in [\![\varphi]\!]^{\mathcal{G}}$, and we say that $\mathcal{G}, u$ *satisfies* φ.

Definition 2 (Indistinguishability)**.** We write $\mathcal{G}, u \equiv \mathcal{G}', u'$ if $\mathcal{G}, u \models \varphi \Leftrightarrow \mathcal{G}', u' \models \varphi$ for every node expression φ of XPath$_=$. Further, we write $\mathcal{G}, u \Rrightarrow \mathcal{G}', u'$ if $\mathcal{G}, u \models \varphi \Rightarrow \mathcal{G}', u' \models \varphi$ for every positive node expression φ of XPath$_=$. □

In the next section we introduce a notion of (bi)simulation that characterizes, in a precise sense, logical *indistinguishability* (i.e., the relations $\equiv$ and $\Rrightarrow$) for XPath$_=$.

3 Bisimulations on data graphs

The notion of (bi)simulation for XPath$_=$ over *data trees* was developed in (Figueira, Figueira, and Areces 2014). As we observe in this section, this notion is robust and extends in a straightforward way to data graphs.

Definition 3 (XPath$_=$-bisimulations)**.** Let $\mathcal{G}$ and $\mathcal{G}'$ be data graphs over $\mathbb{A}$. An XPath$_=$*-bisimulation* between $u \in G$ and $u' \in G'$ (written $\mathcal{G}, u \leftrightarrows \mathcal{G}', u'$) is a relation $Z \subseteq G \times G'$ such that uZu' and for all $(x,x') \in G \times G'$ such that xZx' we have:

- **(Zig$_=$)** If there are paths $\pi_1 = x \xrightarrow{e_1} x_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} x_n$ and $\pi_2 = x \xrightarrow{d_1} y_1 \xrightarrow{d_2} \ldots \xrightarrow{d_m} y_m$, (for $e_i, d_j \in \mathbb{A}$) in $\mathcal{G}$, then there are paths $\pi_1' = x' \xrightarrow{e_1} x_1' \xrightarrow{e_2} \ldots \xrightarrow{e_n} x_n'$ and $\pi_2' = x' \xrightarrow{d_1} y_1' \xrightarrow{d_2} \ldots \xrightarrow{d_m} y_m'$, in $\mathcal{G}'$ such that:
  1. $x_i Z x_i'$ for all $i \in [1,n]$, and $y_j Z y_j'$ for all $j \in [1,m]$.
  2. $data(x_n) = data(y_m) \Leftrightarrow data(x_n') = data(y_m')$.
- **(Zag$_=$)** If there are paths $\pi_1' = x' \xrightarrow{e_1} x_1' \xrightarrow{e_2} \ldots \xrightarrow{e_n} x_n'$ and $\pi_2' = x' \xrightarrow{d_1} y_1' \xrightarrow{d_2} \ldots \xrightarrow{d_m} y_m'$, (for $e_i, d_j \in \mathbb{A}$) in $\mathcal{G}'$, then there are paths: $\pi_1 = x \xrightarrow{e_1} x_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} x_n$ and $\pi_2 = x \xrightarrow{d_1} y_1 \xrightarrow{d_2} \ldots \xrightarrow{d_m} y_m$, in $\mathcal{G}$ such that conditions 1 and 2 above are verified.

Further, an XPath$_=$*-simulation* from $u \in G$ to $u' \in G'$ (denoted $\mathcal{G}, u \rightrightarrows \mathcal{G}', u'$) is a relation $Z \subseteq G \times G'$ such that uZu' and for all $x \in G$ and $x' \in G'$ the condition **(Zig$_=$)** above is verified.[1] □

It is worth comparing our XPath$_=$-bisimulation with the classical bisimulation notion for ML. There are two simple ways of transforming a data graph into a Kripke structure:

1. We erase the data values, obtaining a Kripke structure with empty valuation for the propositional variables. Under this interpretation, it is clear that if two nodes are XPath$_=$-bisimilar then they are ML-bisimilar, while the converse implication is not true in general (see Figure 4).
2. We assign a propositional variable p_d to every node of the data graph with data value d in the data graph. In contrast with the previous case, now the existence of an ML-bisimulation over the Kripke structure implies the existence of a data-aware bisimulation in the original data

[1] Both notions are slightly different from the ones in (Figueira, Figueira, and Areces 2014). This is because the data trees studied in such article are *node-labeled* while our data graphs are *edge-labeled*. The difference is, of course, inessential to the results.

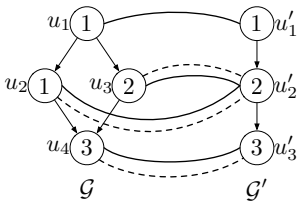

Figure 4: The dotted lines represent the largest bisimulation for XPath$_=$. Observe that, in particular, $\mathcal{G}, u_1 \not\leftrightarrows \mathcal{G}', u'_1$ and $\mathcal{G}, u_2 \leftrightarrows \mathcal{G}', u'_2$. If $\mathcal{G}$ and $\mathcal{G}'$ are regarded as Kripke models without propositional variables, the solid lines represent an ML-bisimulation. In particular, u_1 and u'_1 are bisimilar for this logic. If $\mathcal{G}$ and $\mathcal{G}'$ are regarded as Kripke models with one propositional variable for each data value, then u_2 and u'_2 are ML-bisimilar.

graph, but the converse does not hold (see Figure 4). This is because XPath$_=$ (and, therefore, the data-aware bisimulations studied in the paper) cannot speak about a particular data value (it cannot express "data value d holds in this node"), but can only check whether two paths finish in nodes with the same (or different) data value.

It is not hard to verify that XPath$_=$-(bi)simulations are closed under union; i.e., if $Z_1 \subseteq G \times G'$ and $Z_2 \subseteq G \times G'$ are XPath$_=$-(bi)simulations between $u \in G$ and $u' \in G$, then so is $Z_1 \cup Z_2$. This immediately implies the following:

Proposition 4. *If there is an XPath$_=$-(bi)simulation between $u \in G$ and $u' \in G$, then there is a maximal such XPath$_=$-(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 characterization of the notion of logical indistinguishability for XPath$_=$ in terms of XPath$_=$-(bi)simulations:

Theorem 5. *(Figueira, Figueira, and Areces 2015) Let $\mathcal{G}$ and $\mathcal{G}'$ be data graphs over the same alphabet $\mathbb{A}$, and u and u' nodes in G and G', respectively. Then $\mathcal{G}, u \equiv \mathcal{G}', u'$ iff $\mathcal{G}, u \leftrightarrows \mathcal{G}', u'$, and $\mathcal{G}, u \Rrightarrow \mathcal{G}', u'$ iff $\mathcal{G}, u \rightrightarrows \mathcal{G}', u'$.*

In the example of Figure 4, $\mathcal{G}, u_1 \not\equiv \mathcal{G}', u'_1$ since $\mathcal{G}, u_1 \models \langle \downarrow \neq \downarrow \rangle$ but $\mathcal{G}', u'_1 \not\models \langle \downarrow \neq \downarrow \rangle$. Another distinguishing node expression could be $\langle \varepsilon = \downarrow \rangle$. Notice that u_2 and u'_2 cannot be distinguished in XPath$_=$ though they have different data values (as this cannot be expressed in the logic).

Notice that if we add the transitive reflexive closure $\downarrow_a^*$ of $\downarrow_a$ to our language, the notion of (bi)simulation does not change, and Theorem 5 still holds (when $\Rrightarrow$ and $\equiv$ are replaced with the corresponding indistinguishability notion).

4 Computing XPath$_=$-bisimulations

As mentioned in the Introduction, a fundamental problem when dealing with (bi)simulations is checking whether a pair of nodes is (bi)similar. In this section we study the complexity of such problem for XPath$_=$-(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 efficiently.

Formally, we study the following problem:

XPATH$_=$-(BI)SIMILARITY
INPUT : Data graphs $\mathcal{G}$ and $\mathcal{G}'$, nodes $u \in G$ and $u' \in G'$. QUESTION : $\mathcal{G}, u \rightrightarrows \mathcal{G}', u'$? ($\mathcal{G}, u \leftrightarrows \mathcal{G}', u'$?, resp.)

Our main result establishes the following:

Theorem 6. *The problem* XPATH$_=$-(BI)SIMILARITY *is* PSPACE*-complete.*

From Theorem 5 we obtain:

Corollary 7. *The problem of checking $\mathcal{G}, u \equiv \mathcal{G}', u'$ or $\mathcal{G}, u \Rrightarrow \mathcal{G}', u'$, given data graphs $\mathcal{G}$ and $\mathcal{G}'$ and nodes $u \in G$ and $u' \in G'$, is* PSPACE*-complete.*

Further, the following holds by direct inspection of the proof of Theorem 6:

Corollary 8. *Checking $\mathcal{G}, u \equiv \mathcal{G}', u'$ or $\mathcal{G}, u \Rrightarrow \mathcal{G}', u'$ is* PSPACE*-complete, even if $\equiv$ and $\Rrightarrow$ are defined only with respect to formulas of the form $\langle \varepsilon = \downarrow_{e_1} \ldots \downarrow_{e_n} \rangle$, for $e_1 \ldots e_n \in \mathbb{A}$.*

To prove Theorem 6 we show that XPATH$_=$-(BI)SIMILARITY is polynomially equivalent to the *containment problem* (resp., *equivalence problem*) for nondeterministic finite automata (NFAs). Recall that this is defined as follows: Given NFAs $\mathcal{A}_1$ and $\mathcal{A}_2$, is $L(\mathcal{A}_1)$ contained in $L(\mathcal{A}_2)$ (resp., are both languages the same)? Both containment and equivalence of NFAs are PSPACE-complete (Meyer and Stockmeyer 1972).

4.1 Upper bound

We prove a PSPACE upper bound for XPATH$_=$-SIMILARITY. The proof for bisimilarity is analogous but using equivalence instead of containment of NFAs. The algorithm first guesses the candidate relation $Z \subseteq G \times G'$, which contains (u, u'), and then checks that it satisfies the (**Zig**$_=$) property. Since Z is of polynomial size and PSPACE = NPSPACE, 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 the latter 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 $x \in G$, it is not hard to construct in polynomial time an NFA $\mathcal{A}_{\mathcal{G},x}$ over alphabet $\mathbb{A} \times G \cup \{=, \neq\}$ that accepts precisely those words of the form:

$$(e_1, x_1) \ldots (e_n, x_n) \star (d_1, y_1) \ldots (d_m, y_m), \qquad (1)$$

for $\star \in \{=, \neq\}$, such that $\mathcal{G}$ contains paths:

$$x \xrightarrow{e_1} x_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} x_n \quad \text{and} \quad x \xrightarrow{d_1} y_1 \xrightarrow{d_2} \ldots \xrightarrow{d_m} y_m$$

for which $data(x_n) \star data(y_m)$. In the same way, it is possible to construct in polynomial time an NFA $\mathcal{A}_{\mathcal{G}',x',Z}$ over alphabet $\mathbb{A} \times G \cup \{=, \neq\}$ that accepts precisely those words of the form (1) 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:

- $G_i = Q_i \cup \{u_i, v_i, w_i\}$
- $(x, a, y) \in E_i$ iff 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 $y \in \{u_i, v_i, w_i\}$
- $data_i(q) =$
$$\begin{cases} 1 & \text{if } q \in \{q_i\} \cup F_i \\ 2 & \text{if } q \in \{u_i\} \cup Q_i \setminus \{q_i, f_i\} \\ 3 & \text{if } q = v_i \\ 4 & \text{if } q = w_i \end{cases}$$

- xZy iff one of the following holds:
  - $x = q_1$ and $y = q_2$.
  - $x \in \{u_1\} \cup Q_1 \setminus \{q_1, f_1\}$ and $y \in \{u_2\} \cup Q_2 \setminus \{q_2, f_2\}$.
  - $x \in \{v_1, w_1, f_1\}$ and $y \in \{v_2, w_2, f_2\}$.

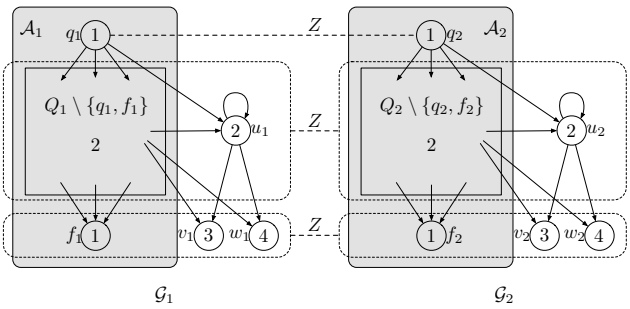

Figure 5: The data graphs $\mathcal{G}_i = (G_i, E_i, data_i)$ $(i = 1, 2)$, constructed from NFAs $\mathcal{A}_1$ and $\mathcal{A}_2$, and the bisimulation $Z \subseteq G_1 \times G_2$. All nodes inside a dotted area on $\mathcal{G}_1$ are related to all nodes inside a dotted on $\mathcal{G}_2$ area via Z.

- $x_i Z x'_i$ for $i \in [1, n]$, and $y_j Z y'_j$ for $j \in [1, m]$.
- $data(x'_n) \star data(y'_m)$.

It is clear that Z satisfies **(Zig$_=$)** if and only if $\mathcal{A}_{\mathcal{G},x} \subseteq \mathcal{A}_{\mathcal{G}',x',Z}$ for each $(x, x') \in Z$. Since Z is of polynomial size, we only need to check containment for a polynomial number of pairs of NFAs. This establishes the upper bound.

4.2 Lower bound

Again, we only show the lower bound for XPATH$_=$-SIMILARITY. The proof for bisimilarity is analogous. We proceed by constructing a polynomial time reduction from containment of NFAs to XPATH$_=$-SIMILARITY. Let $\mathcal{A}_i = (Q_i, \delta_i, q_i, F_i)$ be NFAs over alphabet $\mathbb{A}$, for $i = 1, 2$. Here, (a) Q_i is the finite set of states of $\mathcal{A}_i$, (b) $\delta_i \subseteq Q_i \times \mathbb{A} \times Q_i$ is its transition relation, (c) q_i is the initial state, and (d) F_i is the set of final states. We assume, without loss of generality, that q_i has no incoming transitions and F_i consists of a single state f_i without outgoing transitions. Furthermore, we assume that f_i (for $i = 1, 2$) is reachable from every state in Q_i and that $Q_1 \cap Q_2 = \emptyset$.

Let $u_1, u_2, v_1, v_2, w_1, w_2$ be 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 in Figure 5. We show that $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$ iff $\mathcal{G}_1, q_1 \rightrightarrows \mathcal{G}_2, q_2$. The right-to-left direction is straightforward, since for $e_1 \dots e_n \in \mathbb{A}$ $(n > 0)$ and $\varphi = \langle \varepsilon = \downarrow_{e_1} \dots \downarrow_{e_n} \rangle$ we have that: $e_1 \dots e_n \in L(\mathcal{A}_1)$ then by construction $\mathcal{G}_1, q_1 \models \varphi$ then (since $\mathcal{G}_1, q_1 \rightrightarrows \mathcal{G}_2, q_2$) $\mathcal{G}_2, q_2 \models \varphi$, and then by construction $e_1 \dots e_n \in L(\mathcal{A}_2)$. For the left-to-right direction, let us define $Z \subseteq G_1 \times G_2$ as in Figure 5. We show next that Z satisfies the **(Zig$_=$)** clause for any pair $x_1 \in G_1, x_2 \in G_2$ such that $x_1 Z x_2$.

If $x_1 \in \{f_1, v_1, w_1\}$ and $x_2 \in \{f_2, v_2, w_2\}$, **(Zig$_=$)** holds trivially as there are no outgoing paths from f_1, v_1 or w_1. 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 $\alpha = x_1$ and $\beta = x_1 y_2 \dots y_m z$ (resp., $x_1 y_2 \dots y_m$), 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 $x_2 u_2 \dots u_2 v_2$ (resp., $x_2 u_2 \dots u_2$), 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 $\{1, 3, 4\}$.

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 $\mathcal{G}_2$, assuming they are of length greater than 0 (when one of the paths is of length 0 the analysis is easier). If any of these paths ends in f_1, its inner nodes (save for the beginning and the end), must lay in $Q_1 \setminus \{q_1, f_1\}$; thus the path corresponds with a word of $L(\mathcal{A}_1)$. But $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$, and thus this path can also be replicated starting from q_2. On the other hand, if any of these paths ends in a node different from f_1, say $\alpha = q_1 y_2 \dots y_m$, then $\alpha' = q_2 u_2 \dots u_2 t$, with $m - 2$ occurrences of u_2, and where $t = u_2$ if $y_m \in Q_1 \setminus \{q_1, f_1\} \cup \{u_1\}$, $t = v_2$ if $y_m = v_1$, and $t = w_2$ if $y_m = w_1$.

5 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 (Meyer and Stockmeyer 1972). As a corollary to the proof of Theorem 6, we obtain then that the smallest witness to the fact that a given relation $Z \subseteq G \times G'$ does not satisfy the **(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$_=$-(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$_=$. 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$_=$-(bi)similarity based on user-understandable formulas.
- In practice, algorithms for computing (bi)simulations in the absence of data stop after a few iterations (Luo et al. 2013b; 2013a). This tells us that when nodes in real-world graphs are distinguishable by ML formulas, they are distinguishable by some small formula. One might expect a similar behavior for XPath$_=$, 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$_=$-(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$_=$; namely, the one formed by expressions of *bounded length*. This fragment only restricts formulas of the form $\langle \alpha \star \beta \rangle$, for $\star \in \{=, \neq\}$.

5.1 Bounded bisimulation and equivalence

Let $f : \mathbb{N} \to \mathbb{N}$ be a positive, non-decreasing function. We define the notion of f-XPath$_=$*-(bi)simulation* as in Definition 3, but now in the **(Zig$_=$)** (resp., **(Zag$_=$)**) condition we only consider paths π_1 and π_2 (resp., π_1' and π_2') of length at most $f(\max(|G|, |G'|))$. We call the new conditions **(Zig$^f_=$)** and **(Zag$^f_=$)**, respectively. We write $\mathcal{G}, u \leftrightarrows_f \mathcal{G}', u'$ if there is an f-XPath$_=$-bisimulation between $\mathcal{G}$ and $\mathcal{G}'$ that contains the pair (u, u'). Similarly, $\mathcal{G}, u \rightrightarrows_f \mathcal{G}', u'$ if there is an f-XPath$_=$-simulation from u in $\mathcal{G}$ to u' in $\mathcal{G}'$.

We define the logical counterpart of this refined notion of bisimulation. We aim at an analog of Theorem 5 relative to the adequate restriction of indistinguishability (*cf.* Def. 2). As we show below, this restriction is defined by the fragment of XPath$_=$ 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.

The *length* of a path expression α, denoted $\mathsf{len}(\alpha)$, was defined in (Abriola, Descotte, and Figueira 2015). It corresponds 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 9 (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:

$$\begin{array}{rcl} \mathsf{ml}(\lambda) & = & 0 \\ \mathsf{ml}(\varepsilon\alpha) & = & \mathsf{ml}(\alpha) \\ \mathsf{ml}([\varphi]\alpha) & = & \max\{\mathsf{ml}(\varphi), \mathsf{ml}(\alpha)\} \\ \mathsf{ml}(\downarrow_a \alpha) & = & \mathsf{ml}(\alpha) \\ \mathsf{ml}(\varphi \wedge \psi) & = & \max\{\mathsf{ml}(\varphi), \mathsf{ml}(\psi)\} \\ \mathsf{ml}(\neg\varphi) & = & \mathsf{ml}(\varphi) \\ \mathsf{ml}(\langle\alpha\rangle) & = & \mathsf{ml}(\alpha) \\ \mathsf{ml}(\langle \alpha \star \beta \rangle) & = & \max\{\mathsf{len}(\alpha), \mathsf{len}(\beta), \mathsf{ml}(\alpha), \mathsf{ml}(\beta)\}, \end{array}$$

where α is any path expression or the empty string λ. For $c \geq 0$, we call XPath$_=(c)$ the syntactical fragment of XPath$_=$ of ml bounded by c. □

E.g., $\mathsf{ml}(\langle \downarrow_a [\langle \downarrow_b \downarrow_a \downarrow_c \rangle] \downarrow_b = \downarrow_b [\langle \downarrow_a \downarrow_b \downarrow_a \downarrow_b \rangle] \rangle) = 2$.

Notice that XPath$_=(0)$ corresponds of the fragment of the logic without data value comparisons $\langle \alpha \star \beta \rangle$ ($\star \in \{=, \neq\}$). Further, fragments of the form XPath$_=(c)$ (for $c \geq 1$) extend multi-modal logic, which in turn coincides with XPath$_=(0)$:

Proposition 10. *XPath$_=(0)$ is semantically equivalent to multi-modal logic with no propositions and only atom* $\top$.

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 XPath$_=(0)$ is also of the form $\langle [\varphi_1] \dots [\varphi_n] \star [\psi_1] \dots [\psi_m] \rangle$, 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 XPath$_=(0)$ 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 XPath$_=(0)$ is obvious. □

We now introduce the notion of f-XPath$_=$-indistinguishability which matches f-XPath$_=$-bisimulation (*cf.* Definition 2). We write $\mathcal{G}, u \equiv_f \mathcal{G}', u'$ (resp. $\mathcal{G}, u \Rrightarrow_f \mathcal{G}', u'$) if $\mathcal{G}, u \models \varphi \Leftrightarrow \mathcal{G}', u' \models \varphi$ (resp. $\mathcal{G}, u \models \varphi \Rightarrow \mathcal{G}', u' \models \varphi$) for every (positive) node expression φ of XPath$_=$ such that $\mathsf{ml}(\varphi) \leq f(\max(|G|, |G'|))$.

As in Theorem 5 we obtain the following characterization:

Proposition 11. *$\mathcal{G}, u \equiv_f \mathcal{G}', u'$ iff $\mathcal{G}, u \leftrightarrows_f \mathcal{G}', u'$, and $\mathcal{G}, u \Rrightarrow_f \mathcal{G}', u'$ iff $\mathcal{G}, u \rightrightarrows_f \mathcal{G}', u'$.*

5.2 Computing f-XPath$_=$-bisimulations

Here we study the complexity of computing f-XPath$_=$-(bi)simulations:

f-XPath$_=$-(Bi)Similarity
INPUT : Data graphs $\mathcal{G}$ and $\mathcal{G}'$, nodes $u \in G$ and $u' \in G'$.
QUESTION : $\mathcal{G}, u \rightrightarrows_f \mathcal{G}', u'$? ($\mathcal{G}, u \leftrightarrows_f \mathcal{G}', u'$?, 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 12. *The following holds:*

1. *The problem p-XPath$_=$-(Bi)Similarity is in* Co-NP *for every non-decreasing polynomial $p : \mathbb{N} \to \mathbb{N}$.*
2. *For every $k \geq 1$ there exists a non-decreasing polynomial $p : \mathbb{N} \to \mathbb{N}$ of degree k such that p-XPath$_=$-(Bi)Similarity is* Co-NP*-hard.*

Proof. We only prove the claim for p-XPath$_=$-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$_=$-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 **(Zig$^p_=$)** condition of XPath$_=$-simulations.

The algorithm computes the maximal XPath$_=$-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 **(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$_=$-simulation from u to u'.

Thus, in order to check that there is *no* XPath$_=$-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 $(x_1, x_1'), \dots, (x_m, x_m')$; (b) relations $Z_0, \dots, Z_m$ 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 **(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 **(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 6) 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—in particular, to check that the word representing (π_1, π_2) does not belong to the language of $\mathcal{A}_{\mathcal{G}', x'_i, Z_{i-1}}$. It follows that checking whether there is *no* XPath$_=$-simulation from u to u' is in NP (and, thus, that our problem is in CO-NP).

For item 2 we use the following fact: For every $k \geq 1$ there exists a non-decreasing polynomial $p : \mathbb{N} \to \mathbb{N}$ of degree k such that the problem of checking containment of NFA $\mathcal{A}_1$ in $\mathcal{A}_2$, restricted to words of length at most $p(\max(|\mathcal{A}_1|, |\mathcal{A}_2|))$, is CO-NP-hard. This is obtained by simply mimicking the proof of PSPACE-hardness for containment of NFAs, but this time restricted to words of polynomial length only. Once this is established, we reduce such restricted containment problem to p-XPATH$_=$-SIMILARITY by applying the construction in the proof of the lower bound of Theorem 6. □

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.

Proposition 13. *The problem* c-XPATH$_=$-(BI)SIMULATION *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 **(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 $\mathcal{G}, u \rightrightarrows_c \mathcal{G}', u'$ is in PTIME. The same holds for $\mathcal{G}, u \leftrightarrows_c \mathcal{G}', u'$. The lower bound follows from PTIME-hardness for bisimulations (Balcázar, Gabarro, and Santha 1992). □

Recall that Proposition 11 establishes that c-XPath$_=$-simulations characterize the expressive power of the fragment of XPath$_=$ defined by formulas of ml bounded by the constant c. The following corollary to the proof of Proposition 13 states that when two nodes are not c-XPath$_=$-bisimilar, it is possible to compute in polynomial time a node expression in this fragment that distinguishes them.

Corollary 14. *There is a polynomial time algorithm which given* $\mathcal{G}, u \not\rightrightarrows_c \mathcal{G}', u'$ *(resp.,* $\mathcal{G}, u \not\leftrightarrows_c \mathcal{G}', u'$*), constructs*[2] *a (positive) node expression* φ *of XPath*$_=(c)$ *such that* $\mathcal{G}, u \models \varphi$ *and* $\mathcal{G}', u' \not\models \varphi$.

[2]Provided an adequate representation for such formula is chosen (Figueira and Gorín 2010).

6 Restricting the models

Here we follow a different approach from the one in §5: 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 data trees are (essentially) XML documents, and the study of XPath$_=$-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 **(Zig$_=$)** and **(Zag$_=$)** conditions are polynomial. This implies that the problem of computing XPath$_=$-(bi)simulations over data trees is tractable:

Theorem 15. *The problem* XPATH$_=$-(BI)SIMULATION *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 **(Zig$_=$)** and **(Zag$_=$)** conditions is polynomial. The first observation helps lowering the complexity of computing XPath$_=$-(bi)simulations in this context from PSPACE to CO-NP, while the second one prevents us from obtaining tractability.

Proposition 16. *The problem* XPATH$_=$-(BI)SIMULATION *for data DAGs is* CO-NP*-complete.*

Proof. The upper bound is a consequence of Proposition 12, since paths in DAGs are of linear size. For the lower bound we use the following observation: Checking containment/equivalence of DAG NFAs (i.e., NFAs whose underlying graph is a DAG) is CO-NP-complete. This can be proved by an easy reduction from the complement of 3SAT. Once this is established, we can adapt the proof of Theorem 6. Given $\mathcal{A}_1, \mathcal{A}_2$ DAG NFAs, we can build data DAGs $\mathcal{G}_1, \mathcal{G}_2$ reducing from the problem of equivalence/containment. In order to obtain data DAG, the nodes u_1 and u_2 of Figure 5 are replaced with DAGs of size n and data value 2, where n is the maximum number of states of the NFAs. Let Q^j_i 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 not of length $> j$ (note that $Q_i = \bigcup_{j \leq n} Q^j_i$). Each u_i is replaced with n fresh nodes $u^1_i, \ldots, u^n_i$ with every possible edge from Q^j_i to u^{j-1}_i, from u^j_i to v_i, w_i, and from u^j_i to $u^{j'}_i$ for every $j' < j$. It is easy to check that the resulting graphs are DAGs, and that the reductions are preserved. This time, the maximal bisimulation relates all nodes with equal maximum distance from $\{f_i, v_i, w_i\}$. □

7 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., (Calvanese et al. 2000; Libkin, Martens, and Vrgoč 2013)). We call XPath$^{\updownarrow}_=$ the extension of XPath$_=$ 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 $\mathcal{G}, u \equiv^{\updownarrow} \mathcal{G}', u'$ (resp. $\mathcal{G}, u \Rrightarrow^{\updownarrow} \mathcal{G}', u'$) if $\mathcal{G}, u \models \varphi \Leftrightarrow \mathcal{G}', u' \models \varphi$ (resp. $\mathcal{G}, u \models \varphi \Rightarrow \mathcal{G}', u' \models \varphi$) for every (positive) node expression φ of XPath$^{\updownarrow}_{=}$.

A notion of (bi)simulation for XPath$^{\updownarrow}_{=}$ *over data trees* was introduced in (Figueira, Figueira, and Areces 2015), 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$_{=}$-bisimulations to the case of two-way XPath$_{=}$.

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 XPath$_{=}$ over $\mathbb{A}$ to node expressions of XPath$^{\updownarrow}_{=}$ 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 17. *$\mathcal{G}, u \models \varphi$ iff $\mathcal{G}_c, u \models \varphi_c$.*

We say that there is is an *XPath$^{\updownarrow}_{=}$-bisimulation* between $u \in G$ and $u' \in G'$ (denoted $\mathcal{G}, u \leftrightarrows^{\updownarrow} \mathcal{G}', u'$) if $\mathcal{G}_c, u \leftrightarrows \mathcal{G}'_c, u'$ (over the extended alphabet $\mathbb{A} \cup \mathbb{A}^-$). Similarly, we define XPath$^{\updownarrow}_{=}$-simulations $\rightrightarrows^{\updownarrow}$. An analog of Theorem 5 can be shown for the case of $\equiv^{\updownarrow}$ (resp. $\Rrightarrow^{\updownarrow}$) and $\leftrightarrows^{\updownarrow}$ (resp. $\rightrightarrows^{\updownarrow}$)

We study the complexity of the following problem:

XPath$^{\updownarrow}_{=}$-(Bi)Similarity
INPUT : Data graphs $\mathcal{G}$ and $\mathcal{G}'$, nodes $u \in G$ and $u' \in G'$. QUESTION : $\mathcal{G}, u \rightrightarrows^{\updownarrow} \mathcal{G}', u'$? ($\mathcal{G}, u \leftrightarrows^{\updownarrow} \mathcal{G}', u'$?, resp.)

The bounded notions of bisimulation introduced in §5 are defined over XPath$^{\updownarrow}_{=}$ and alphabet $\mathbb{A}$ in the expected way: reducing to XPath$_{=}$ over the signature $\mathbb{A} \cup \mathbb{A}^-$ and the corresponding completion of the data graphs. We use symbols $\rightrightarrows^{\updownarrow}_f$ (resp. $\leftrightarrows^{\updownarrow}_f$) for referring to the f-XPath$^{\updownarrow}_{=}$ (bi)simulations. We then study:

f-XPath$^{\updownarrow}_{=}$-(Bi)Similarity
INPUT : Data graphs $\mathcal{G}$ and $\mathcal{G}'$, nodes $u \in G$ and $u' \in G'$. QUESTION : $\mathcal{G}, u \rightrightarrows^{\updownarrow}_f \mathcal{G}', u'$? ($\mathcal{G}, u \leftrightarrows^{\updownarrow}_f \mathcal{G}', u'$?, resp.)

The identification of XPath$^{\updownarrow}_{=}$ over $\mathbb{A}$ with XPath$_{=}$ over $\mathbb{A} \cup \mathbb{A}^-$ and the corresponding completions of graphs allows us to straightforwardly transfer some upper bounds:

- XPath$^{\updownarrow}_{=}$-(Bi)Similarity is in PSpace (§4.1).
- p-XPath$^{\updownarrow}_{=}$-(Bi)Similarity, for p a non-decreasing polynomial, is in Co-NP (item 1 of Proposition 12).
- c-XPath$^{\updownarrow}_{=}$-(Bi)Similarity, for c a constant function, is in PTime (Proposition 13)

Regarding the lower bounds, we focus here on the general case of XPath$^{\updownarrow}_{=}$. One can check that the proof given in §4.2 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:

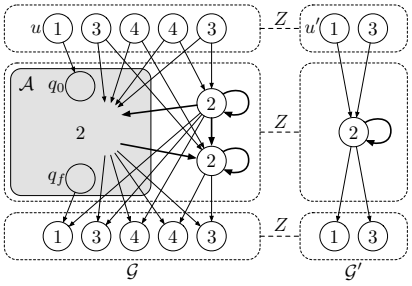

Figure 6: Definition of the data graphs $\mathcal{G}$ and $\mathcal{G}'$ based on an NFA $\mathcal{A}$ over alphabet $\mathbb{A}$. Boldface arrows have, as label, all symbols from $\mathbb{A}$. Lightface arrows have all the same label $e \notin \mathbb{A}$ for some e. All nodes of $\mathcal{A}$ (the 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.

Theorem 18. XPath$^{\updownarrow}_{=}$-(Bi)Similarity *is* PSpace*-hard.*

Proof. We reduce 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 6. We claim that $\mathcal{G}, u \leftrightarrows^{\updownarrow} \mathcal{G}, u'$ iff $L(\mathcal{A}) = \Sigma^*$. The left-to-right direction is straightforward, and for the right-to-left direction, one can check that the relation Z depicted in the figure is an XPath$^{\updownarrow}_{=}$-bisimulation. Due to space constraints we omit details. □

8 Conclusions

As we have seen, while in general computing (bi)simulations on data graphs is PSpace-complete, tractability can be regained by either restricting the topology of the graph, or by relaxing the conditions for bisimulation. Further, several upper bounds continue to hold when inverses are added (the only case left open is for DAGs). The following table summarizes our results:

Model	Problem	Logic	
		XPath$_{=}$	XPath$^{\updownarrow}_{=}$
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 the future we would like to consider XPath with reflexive-transitive axes. Instead of having $\downarrow_a$, we then have $\downarrow^*_A$ denoting pairs of nodes that can be reached through paths with labels from a set A. Although having both $\downarrow_a$ and $\downarrow^*_A$ does not change the indistinguishability (nor bisimulation) relation, having only $\downarrow^*_A$ in the absence of $\downarrow_a$ gives rise to a different bisimulation relation, somewhat akin to ML-bisimulation over transitive frames (Dawar and Otto 2009).

References

Abriola, S.; Descotte, M. E.; and Figueira, S. 2015. Model theory of XPath on data trees. Part II: Binary bisimulation and definability. *Information and Computation*. To appear.

Angles, R., and Gutiérrez, C. 2008. Survey of graph database models. *ACM Comput. Surv.* 40(1).

Areces, C.; Figueira, S.; and Gorín, D. 2011. Using logic in the generation of referring expressions. In *Logical Aspects of Computational Linguistics*. Springer. 17–32.

Areces, C.; Koller, A.; and Striegnitz, K. 2008. Referring expressions as formulas of description logic. In *Proc. of the 5th INLG*.

Balcázar, J.; Gabarro, J.; and Santha, M. 1992. Deciding bisimilarity is P-complete. *Formal aspects of computing* 4(1):638–648.

Barceló, P. 2013. Querying graph databases. In *PODS*, 175–188.

Blackburn, P.; de Rijke, M.; and Venema, Y. 2001. *Modal Logic*. Cambridge University Press.

Bojańczyk, M.; Muscholl, A.; Schwentick, T.; and Segoufin, L. 2009. Two-variable logic on data trees and XML reasoning. *J. ACM* 56(3).

Calvanese, D.; De Giacomo, G.; Lenzerini, M.; and Vardi, M. Y. 2000. Containment of conjunctive regular path queries with inverse. In *KR*, 176–185.

Clarke, E. M.; Grumberg, O.; and Peled, D. 2001. *Model checking*. MIT Press.

Dalmau, V.; Kolaitis, P. G.; and Vardi, M. Y. 2002. Constraint satisfaction, bounded treewidth, and finite-variable logics. In *CP*, 310–326.

David, C.; Gheerbrant, A.; Libkin, L.; and Martens, W. 2013. Containment of pattern-based queries over data trees. In *ICDT*, 201–212. ACM.

Dawar, A., and Otto, M. 2009. Modal characterisation theorems over special classes of frames. *Annals of Pure and Applied Logic* 161(1):1–42.

Dechter, R. 1992. From local to global consistency. *Artif. Intell.* 55(1):87–108.

Dechter, R. 2003. *Constraint processing*. Elsevier Morgan Kaufmann.

Fan, W.; Li, J.; Wang, X.; and Wu, Y. 2012. Query preserving graph compression. In *SIGMOD*, 157–168.

Figueira, S., and Gorín, D. 2010. On the size of shortest modal descriptions. In *Advances in Modal Logic*, volume 8, 114–132.

Figueira, D.; Figueira, S.; and Areces, C. 2014. Basic model theory of xpath on data trees. In *ICDT*, 50–60.

Figueira, D.; Figueira, S.; and Areces, C. 2015. Model theory of XPath on data trees. Part I: Bisimulation and characterization. *Journal of Artificial Intelligence Research* 53:271–314.

Figueira, D. 2010. *Reasoning on Words and Trees with Data*. PhD thesis, Laboratoire Spécification et Vérification, ENS Cachan, France.

Fischer, M. J., and Ladner, R. E. 1979. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.* 18(2):194–211.

Getoor, L., and Diehl, C. P. 2005. Link mining: a survey. *ACM SIGKDD Explorations Newsletter* 7(2):3–12.

Givan, R.; Dean, T. L.; and Greig, M. 2003. Equivalence notions and model minimization in markov decision processes. *Artif. Intell.* 147(1-2):163–223.

Kolaitis, P. G., and Vardi, M. Y. 2000. A game-theoretic approach to constraint satisfaction. In *AAAI*, 175–181.

Krahmer, E.; van Erk, S.; and Verleg, A. 2003. Graph-based generation of referring expressions. *Computational Linguistics* 29(1).

Kupferman, O., and Vardi, M. Y. 1998. Verification of fair transition systems. *Chicago J. Theor. Comput. Sci.* 1998.

Kurtonina, N., and de Rijke, M. 1999. Expressiveness of concept expressions in first-order description logics. *Artif. Intell.* 107(2):303–333.

Libkin, L., and Vrgoč, D. 2012. Regular path queries on graphs with data. In *ICDT*, 74–85.

Libkin, L.; Martens, W.; and Vrgoč, D. 2013. Querying graph databases with XPath. In *ICDT*, 129–140. ACM.

Luo, Y.; Fletcher, G. H. L.; Hidders, J.; Bra, P. D.; and Wu, Y. 2013a. Regularities and dynamics in bisimulation reductions of big graphs. In *GRADES 2013*, 13.

Luo, Y.; Fletcher, G. H. L.; Hidders, J.; Wu, Y.; and Bra, P. D. 2013b. External memory k-bisimulation reduction of big graphs. In *22nd ACM CIKM'13*, 919–928.

Lutz, C. 2003. Description logics with concrete domains—a survey. In *Advances in Modal Logics Volume 4*. King's College Publications.

Meyer, A. R., and Stockmeyer, L. J. 1972. The equivalence problem for regular expressions with squaring requires exponential space. In *SWAT (FOCS)*, 125–129.

Milner, R. 1971. An algebraic definition of simulation between programs. In *Proceedings of the 2nd International Joint Conference on Artificial Intelligence*, 481–489.

Milner, R. 1999. *Communicating and mobile systems - the Pi-calculus*. Cambridge University Press.

Milo, T., and Suciu, D. 1999. Index structures for path expressions. In *ICDT*, 277–295.

Park, D. M. R. 1981. Concurrency and automata on infinite sequences. In *Theoretical Computer Science, 5th GI-Conference, Karlsruhe, Germany, March 23-25, 1981, Proceedings*, 167–183.

Robinson, I.; Webber, J.; and Eifrem, E. 2013. *Graph Databases*. O'Reilly Media, Inc.

Sangiorgi, D. 2009. On the origins of bisimulation and coinduction. *ACM Trans. Program. Lang. Syst.* 31(4):1–41.

van Benthem, J. 1976. *Modal Correspondence Theory*. PhD thesis, Universiteit van Amsterdam.