Hybrid Logic Tango

Carlos Areces and Patrick Blackburn

INRIA Nancy Grand Est
Nancy, France

February 2008

Plan for Today

Plan for Today

- Inference tasks

Plan for Today

- Inference tasks
- Model checking FO

Plan for Today

- Inference tasks
- Model checking FO
- Model checking hybrid logic

Plan for Today

- Inference tasks
- Model checking FO
- Model checking hybrid logic
- Decidability and undecidability

Plan for Today

- Inference tasks
- Model checking FO
- Model checking hybrid logic
- Decidability and undecidability
- PSPACE complexity for the basic hybrid logic

Plan for Today

- Inference tasks
- Model checking FO
- Model checking hybrid logic
- Decidability and undecidability
- PSPACE complexity for the basic hybrid logic

So, let's get started...

Which are the problems in hybrid logics?

Which are the problems in hybrid logics?

Let $\mathcal{L}$ be a language and $\mathcal{K}$ a class of models

Which are the problems in hybrid logics?

Let $\mathcal{L}$ be a language and $\mathcal{K}$ a class of models

model comparison:	- Input: two models M_1 and M_2 in $\mathcal{K}$; - Question: are M_1 and M_2 "the same" w.r.t $\mathcal{L}$?; - Output: Yes/No.

Which are the problems in hybrid logics?

Let $\mathcal{L}$ be a language and $\mathcal{K}$ a class of models

model comparison:	- Input: two models M_1 and M_2 in $\mathcal{K}$; - Question: are M_1 and M_2 "the same" w.r.t $\mathcal{L}$?; - Output: Yes/No.
model checking:	- Input: a model M in $\mathcal{K}$ and a formula φ in $\mathcal{L}$; - Question: does M satisfy φ?; - Output: Yes/No.

Which are the problems in hybrid logics?

Let $\mathcal{L}$ be a language and $\mathcal{K}$ a class of models

model comparison:	- Input: two models M_1 and M_2 in $\mathcal{K}$; - Question: are M_1 and M_2 "the same" w.r.t $\mathcal{L}$?; - Output: Yes/No.
model checking:	- Input: a model M in $\mathcal{K}$ and a formula φ in $\mathcal{L}$; - Question: does M satisfy φ?; - Output: Yes/No.
satisfiability:	- Input: a formula φ in $\mathcal{L}$ - Question: is there a model in $\mathcal{K}$ that satisfies φ? - Output: Yes/No.

Which are the problems in hybrid logics?

Let $\mathcal{L}$ be a language and $\mathcal{K}$ a class of models

model comparison:	- Input: two models M_1 and M_2 in $\mathcal{K}$; - Question: are M_1 and M_2 "the same" w.r.t $\mathcal{L}$?; - Output: Yes/No.
model checking:	- Input: a model M in $\mathcal{K}$ and a formula φ in $\mathcal{L}$; - Question: does M satisfy φ?; - Output: Yes/No.
satisfiability:	- Input: a formula φ in $\mathcal{L}$ - Question: is there a model in $\mathcal{K}$ that satisfies φ? - Output: Yes/No.
model generation:	- Input: a formula φ in $\mathcal{L}$ - Question: is there a model M in $\mathcal{K}$ that satisfies φ? - Output: a representation of M.

Model Checking

Model Checking

- Given a formula φ, to solve the satisfiability problem for φ, we may have to inspect all models, and determine whether they satisfy φ
  - models represent "possible ways the world can be," and to check whether φ is satisfiable, we want to find a way in which the world can be that satisfies φ

Model Checking

- Given a formula φ, to solve the satisfiability problem for φ, we may have to inspect all models, and determine whether they satisfy φ
  - models represent "possible ways the world can be," and to check whether φ is satisfiable, we want to find a way in which the world can be that satisfies φ
- Instead of inspecting all models, inspect a fixed model: "the" model in which we are interested

Model Checking

- Given a formula φ, to solve the satisfiability problem for φ, we may have to inspect all models, and determine whether they satisfy φ
  - models represent "possible ways the world can be," and to check whether φ is satisfiable, we want to find a way in which the world can be that satisfies φ
- Instead of inspecting all models, inspect a fixed model: "the" model in which we are interested
- The (finite) model checking problem:
  Given a (finite) model M, and given a formula φ, does M satisfy φ?

Model Checking FO

Model Checking FO

- Thm: (Stockmeyer 1974, Vardi 1982) Model-checking for first-order logic (and also monadic second-order logic) is PSPACE-complete.
  - This is "expression-complexity", i.e., complexity in terms of the input formula.

Model Checking FO

- Thm: (Stockmeyer 1974, Vardi 1982) Model-checking for first-order logic (and also monadic second-order logic) is PSPACE-complete.
  - This is "expression-complexity", i.e., complexity in terms of the input formula.
  - And actually, the theorem holds on any class of structures that contains at least one structure with at least two elements.

Model Checking FO

- Thm: (Stockmeyer 1974, Vardi 1982) Model-checking for first-order logic (and also monadic second-order logic) is PSPACE-complete.
  - This is "expression-complexity", i.e., complexity in terms of the input formula.
  - And actually, the theorem holds on any class of structures that contains at least one structure with at least two elements.
- Inclusion: Easy!
  - Given a finite first-order structure and a first-order formula, just implement an algorithm that cycles through all possible assignments to the free variables, and substitutions for the quantified variables.

Model Checking FO

- Thm: (Stockmeyer 1974, Vardi 1982) Model-checking for first-order logic (and also monadic second-order logic) is PSPACE-complete.
  - This is "expression-complexity", i.e., complexity in terms of the input formula.
  - And actually, the theorem holds on any class of structures that contains at least one structure with at least two elements.
- Inclusion: Easy!
  - Given a finite first-order structure and a first-order formula, just implement an algorithm that cycles through all possible assignments to the free variables, and substitutions for the quantified variables.
  - In each cycle we only need to store polynomially many values. As the values in each cycle are independent of the previous one, we can "overwrite" them, and stay in PSPACE.

Model Checking FO

Model Checking FO

- Hardness: A bit harder. (We take advantage of a clever friend who already proved some complexity results for us).

Model Checking FO

- Hardness: A bit harder. (We take advantage of a clever friend who already proved some complexity results for us).
  - the proof is a reduction from the Quantified Boolean Formula (QBF) validity problem
  - essentially, evaluating a QBF is shown to be equivalent to model-checking a structure with just two distinguishable elements representing the Boolean values True and False

Model Checking FO

- Hardness: A bit harder. (We take advantage of a clever friend who already proved some complexity results for us).
  - the proof is a reduction from the Quantified Boolean Formula (QBF) validity problem
  - essentially, evaluating a QBF is shown to be equivalent to model-checking a structure with just two distinguishable elements representing the Boolean values True and False
- Thus, the complexity of model checking in FO comes purely from large and complicated input formulas

Model Checking FO

- Hardness: A bit harder. (We take advantage of a clever friend who already proved some complexity results for us).
  - the proof is a reduction from the Quantified Boolean Formula (QBF) validity problem
  - essentially, evaluating a QBF is shown to be equivalent to model-checking a structure with just two distinguishable elements representing the Boolean values True and False
- Thus, the complexity of model checking in FO comes purely from large and complicated input formulas
- The input structure only plays a very limited role, it can actually be a fixed two-element structure whose vocabulary only consists of one unary relation symbol

Model Checking Hybrid Logics

Model Checking Hybrid Logics

Given a finite model M and a formula φ, how can we determine if $M, w \models \varphi$?

Model Checking Hybrid Logics

Given a finite model M and a formula φ, how can we determine if $M, w \models \varphi$?

- Suppose that φ is a propositional symbol p_i then, we only need to check whether $w \in V(p_i)$.

Model Checking Hybrid Logics

Given a finite model M and a formula φ, how can we determine if $M, w \models \varphi$?

- Suppose that φ is a propositional symbol p_i then, we only need to check whether $w \in V(p_i)$.
  Graphically, think of the propositional symbols as labels (colors) for the states in W, if w is labeled p_i we are done.

Model Checking Hybrid Logics

Given a finite model M and a formula φ, how can we determine if $M, w \models \varphi$?

- Suppose that φ is a propositional symbol p_i then, we only need to check whether $w \in V(p_i)$.
  Graphically, think of the propositional symbols as labels (colors) for the states in W, if w is labeled p_i we are done.
  (The case for nominals is identical)

Model Checking Hybrid Logics

Given a finite model M and a formula φ, how can we determine if $M, w \models \varphi$?

- Suppose that φ is a propositional symbol p_i then, we only need to check whether $w \in V(p_i)$.
  Graphically, think of the propositional symbols as labels (colors) for the states in W, if w is labeled p_i we are done.
  (The case for nominals is identical)
- Suppose that we know which is the status of ψ in M and we want to know which is the status of $\neg\psi$:

Model Checking Hybrid Logics

Given a finite model M and a formula φ, how can we determine if $M, w \models \varphi$?

- Suppose that φ is a propositional symbol p_i then, we only need to check whether $w \in V(p_i)$.
  Graphically, think of the propositional symbols as labels (colors) for the states in W, if w is labeled p_i we are done.
  (The case for nominals is identical)
- Suppose that we know which is the status of ψ in M and we want to know which is the status of $\neg\psi$:
  for each state we know whether or not they are labeled with ψ, so label with $\neg\psi$ only those states which are not labeled by ψ.

Model Checking Hybrid Logics

Given a finite model M and a formula φ, how can we determine if $M, w \models \varphi$?

- Suppose that φ is a propositional symbol p_i then, we only need to check whether $w \in V(p_i)$.
  Graphically, think of the propositional symbols as labels (colors) for the states in W, if w is labeled p_i we are done.
  (The case for nominals is identical)
- Suppose that we know which is the status of ψ in M and we want to know which is the status of $\neg\psi$:
  for each state we know whether or not they are labeled with ψ, so label with $\neg\psi$ only those states which are not labeled by ψ.
- The interesting case is $\Box\psi$ (again assume we know the status of ψ).

Model Checking Hybrid Logics

Given a finite model M and a formula φ, how can we determine if $M, w \models \varphi$?

- Suppose that φ is a propositional symbol p_i then, we only need to check whether $w \in V(p_i)$.
  Graphically, think of the propositional symbols as labels (colors) for the states in W, if w is labeled p_i we are done.
  (The case for nominals is identical)
- Suppose that we know which is the status of ψ in M and we want to know which is the status of $\neg\psi$:
  for each state we know whether or not they are labeled with ψ, so label with $\neg\psi$ only those states which are not labeled by ψ.
- The interesting case is $\Box\psi$ (again assume we know the status of ψ).
  To know whether we have to label a certain states with $\Box\psi$ we only need to verify that all accessible states are already labeled by ψ.

Model Checking Hybrid Logics

Given a finite model M and a formula φ, how can we determine if $M, w \models \varphi$?

- Suppose that φ is a propositional symbol p_i then, we only need to check whether $w \in V(p_i)$.
  Graphically, think of the propositional symbols as labels (colors) for the states in W, if w is labeled p_i we are done.
  (The case for nominals is identical)
- Suppose that we know which is the status of ψ in M and we want to know which is the status of $\neg\psi$:
  for each state we know whether or not they are labeled with ψ, so label with $\neg\psi$ only those states which are not labeled by ψ.
- The interesting case is $\Box\psi$ (again assume we know the status of ψ).
  To know whether we have to label a certain states with $\Box\psi$ we only need to verify that all accessible states are already labeled by ψ.
  (You can do $@_i\psi$, it's really easy).

Model Checking Hybrid Logics

Model Checking Hybrid Logics

- Let $|\varphi|$ be the number of symbols in φ, and $|M|$ the sum of the number of states in M and the number of pairs in R.

Model Checking Hybrid Logics

- Let $|\varphi|$ be the number of symbols in φ, and $|M|$ the sum of the number of states in M and the number of pairs in R.
- Thm: Given a model M and a hybrid formula φ, determining whether $M \models \varphi$ can be done in time $\mathcal{O}(|M| \cdot |\varphi|)$.

Model Checking Hybrid Logics

- Let $|\varphi|$ be the number of symbols in φ, and $|M|$ the sum of the number of states in M and the number of pairs in R.
- Thm: Given a model M and a hybrid formula φ, determining whether $M \models \varphi$ can be done in time $\mathcal{O}(|M| \cdot |\varphi|)$.
- Proof: Let $\varphi_1, \ldots, \varphi_m$ be the subformulas of φ listed in order of length. There are at most $|\varphi|$ subformulas.

Model Checking Hybrid Logics

- Let $|\varphi|$ be the number of symbols in φ, and $|M|$ the sum of the number of states in M and the number of pairs in R.
- Thm: Given a model M and a hybrid formula φ, determining whether $M \models \varphi$ can be done in time $\mathcal{O}(|M| \cdot |\varphi|)$.
- Proof: Let $\varphi_1, \ldots, \varphi_m$ be the subformulas of φ listed in order of length. There are at most $|\varphi|$ subformulas.
  Start by labeling each state $w \in M$ as follows: if p_i appears in φ then label w with p_i iff $w \in V(p_i)$.

Model Checking Hybrid Logics

Model Checking Hybrid Logics

- Now proceed through the list of subformulas of φ

Model Checking Hybrid Logics

- Now proceed through the list of subformulas of φ
  1. If φ_j is a proposition simbol or a nominal p, then do nothing.

Model Checking Hybrid Logics

- Now proceed through the list of subformulas of φ
  1. If φ_j is a proposition simbol or a nominal p, then do nothing.
  2. If φ_j is $\neg\psi$ then ψ occurs before in the list and hence we have already decided which states are marked with ψ.
     Label with $\neg\psi$ all states not labeled with ψ.

Model Checking Hybrid Logics

- Now proceed through the list of subformulas of φ
  1. If φ_j is a proposition simbol or a nominal p, then do nothing.
  2. If φ_j is $\neg\psi$ then ψ occurs before in the list and hence we have already decided which states are marked with ψ.
     Label with $\neg\psi$ all states not labeled with ψ.
  3. If φ_j is $\psi \wedge \theta$ then both ψ and θ occur before in the list and we have determined which state are labeled by them.
     Label with φ_j all states which are labeled with both ψ and θ.

Model Checking Hybrid Logics

- Now proceed through the list of subformulas of φ
  1. If φ_j is a proposition simbol or a nominal p, then do nothing.
  2. If φ_j is $\neg\psi$ then ψ occurs before in the list and hence we have already decided which states are marked with ψ.
     Label with $\neg\psi$ all states not labeled with ψ.
  3. If φ_j is $\psi \wedge \theta$ then both ψ and θ occur before in the list and we have determined which state are labeled by them.
     Label with φ_j all states which are labeled with both ψ and θ.
  4. If φ_j is $\Box\psi$, we already decided which states are labeled ψ.
     Label a state w with $\Box\psi$ iff each state v such that Rwv is labeled with ψ.

Model Checking Hybrid Logics

- Now proceed through the list of subformulas of φ
  1. If φ_j is a proposition simbol or a nominal p, then do nothing.
  2. If φ_j is $\neg\psi$ then ψ occurs before in the list and hence we have already decided which states are marked with ψ.
     Label with $\neg\psi$ all states not labeled with ψ.
  3. If φ_j is $\psi \wedge \theta$ then both ψ and θ occur before in the list and we have determined which state are labeled by them.
     Label with φ_j all states which are labeled with both ψ and θ.
  4. If φ_j is $\Box\psi$, we already decided which states are labeled ψ.
     Label a state w with $\Box\psi$ iff each state v such that Rwv is labeled with ψ.
  5. You do the case for $\varphi_j = @_i\psi$.
- Steps 2 and 3 need only inspection of a single state, while step 4 needs inspection of at most $|M|$ states. Hence the whole algorithm runs in $\mathcal{O}(|M| \cdot |\varphi|)$.

Decidability and Complexity

Decidability and Complexity

- We are interested in determining the computational properties of the satisfiability problem.

Decidability and Complexity

- We are interested in determining the computational properties of the satisfiability problem.
  - if the problem is decidable, we can ask how complex it is
  - if the problem is undecidable, we might want to know at which "level of non-computability" it resides

Decidability and Complexity

- We are interested in determining the computational properties of the satisfiability problem.
  - if the problem is decidable, we can ask how complex it is
  - if the problem is undecidable, we might want to know at which "level of non-computability" it resides
- Methods for Proving Decidability/Undecidability:

Decidability and Complexity

- We are interested in determining the computational properties of the satisfiability problem.
  - if the problem is decidable, we can ask how complex it is
  - if the problem is undecidable, we might want to know at which "level of non-computability" it resides
- Methods for Proving Decidability/Undecidability:

Decidability:	– via finite models – via interpretations in monadic second order logic – quasi-models and mosaics

Decidability and Complexity

- We are interested in determining the computational properties of the satisfiability problem.
  - if the problem is decidable, we can ask how complex it is
  - if the problem is undecidable, we might want to know at which "level of non-computability" it resides
- Methods for Proving Decidability/Undecidability:

Decidability:	– via finite models – via interpretations in monadic second order logic – quasi-models and mosaics
Undecidability:	– via tiling problems – via the codification of another undecidable logic

Decidability of the Validity Problem

Decidability of the Validity Problem

- Decidability with Completeness.
  - "Theoremhood" in axiomatic terms. The logic K
    - A1 All tautologies of propositional logic
    - A2 $\Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi)$
    - R1 From φ and $\varphi \rightarrow \psi$, infer ψ
    - R2 From φ infer $\Box\varphi$

Decidability of the Validity Problem

- Decidability with Completeness.
  - "Theoremhood" in axiomatic terms. The logic K
    - A1 All tautologies of propositional logic
    - A2 $\Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi)$
    - R1 From φ and $\varphi \rightarrow \psi$, infer ψ
    - R2 From φ infer $\Box\varphi$
  - K is a sound and complete axiom system

Decidability of the Validity Problem

- Decidability with Completeness.
  - "Theoremhood" in axiomatic terms. The logic K
    - A1 All tautologies of propositional logic
    - A2 $\Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi)$
    - R1 From φ and $\varphi \rightarrow \psi$, infer ψ
    - R2 From φ infer $\Box\varphi$
  - K is a sound and complete axiom system
  - The finite model property: φ is valid in every model iff it is valid in all finite models. As K has fmp, it is decidabile... Why?

Decidability of the Validity Problem

- Decidability with Completeness.
  - "Theoremhood" in axiomatic terms. The logic K
    - A1 All tautologies of propositional logic
    - A2 $\Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi)$
    - R1 From φ and $\varphi \rightarrow \psi$, infer ψ
    - R2 From φ infer $\Box\varphi$
  - K is a sound and complete axiom system
  - The finite model property: φ is valid in every model iff it is valid in all finite models. As K has fmp, it is decidabile... Why?
- Decidability without Completeness.
  - If a formula is satisfiable, then it is satisfiable in a finite structure of bounded size (bounded model property)

Decidability of the Validity Problem

- Decidability with Completeness.
  - "Theoremhood" in axiomatic terms. The logic K
    - A1 All tautologies of propositional logic
    - A2 $\Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi)$
    - R1 From φ and $\varphi \rightarrow \psi$, infer ψ
    - R2 From φ infer $\Box\varphi$
  - K is a sound and complete axiom system
  - The finite model property: φ is valid in every model iff it is valid in all finite models. As K has fmp, it is decidabile... Why?
- Decidability without Completeness.
  - If a formula is satisfiable, then it is satisfiable in a finite structure of bounded size (bounded model property)
  - The basic hybrid logic has the bmp: if φ is satisfiable, then it is satisfiable in a model with at most $2^{|\varphi|}$ elements

Finite Models

Finite Models

- A logic L has the f-size model property with respect to a class of models M if:
  - whenever a formula φ is consistent, then
  - it can be satisfied in a model of M with less that $f(|\varphi|)$ states.

Finite Models

- A logic L has the f-size model property with respect to a class of models M if:
  - whenever a formula φ is consistent, then
  - it can be satisfied in a model of M with less that $f(|\varphi|)$ states.
- L has the strong finite model property with respect to a class of models M if:
  - there is a computable function f such that
  - L has the f-size model property with respect to M.

Finite Models

- A logic L has the f-size model property with respect to a class of models M if:
  - whenever a formula φ is consistent, then
  - it can be satisfied in a model of M with less that $f(|\varphi|)$ states.
- L has the strong finite model property with respect to a class of models M if:
  - there is a computable function f such that
  - L has the f-size model property with respect to M.
- Thm: If L is a modal logic with the strong finite model property with respect to a recursive set of models then L is decidable.

A Recipe for Small Models: Filtrations

A Recipe for Small Models: Filtrations

- Given $M = (W, R, V)$ a model and Σ a set of formulas closed under subformulas, define the relation $\equiv_\Sigma$ as:
  $w \equiv_\Sigma v$ iff for all φ in Σ: $(M, w \models \varphi$ iff $M, v \models \varphi)$.
  Clearly, $\equiv_\Sigma$ is an equivalence relation

A Recipe for Small Models: Filtrations

- Given $M = (W, R, V)$ a model and Σ a set of formulas closed under subformulas, define the relation $\equiv_\Sigma$ as:
  $w \equiv_\Sigma v$ iff for all φ in Σ: $(M, w \models \varphi$ iff $M, v \models \varphi)$.
  Clearly, $\equiv_\Sigma$ is an equivalence relation
- A filtration of M via Σ is any model $M^f_\Sigma = (W^f, R^f, V^f)$ that satisfies
  - $W^f = \{|w|_{/\Sigma} \mid w \in W\}$.

A Recipe for Small Models: Filtrations

- Given $M = (W, R, V)$ a model and Σ a set of formulas closed under subformulas, define the relation $\equiv_\Sigma$ as:
  $w \equiv_\Sigma v$ iff for all φ in Σ: $(M, w \models \varphi$ iff $M, v \models \varphi)$.
  Clearly, $\equiv_\Sigma$ is an equivalence relation
- A filtration of M via Σ is any model $M^f_\Sigma = (W^f, R^f, V^f)$ that satisfies
  - $W^f = \{|w|_{/\Sigma} \mid w \in W\}$.
  - if Rwv then $R^f|w||v|$.

A Recipe for Small Models: Filtrations

- Given $M = (W, R, V)$ a model and Σ a set of formulas closed under subformulas, define the relation $\equiv_\Sigma$ as:
  $w \equiv_\Sigma v$ iff for all φ in Σ: $(M, w \models \varphi$ iff $M, v \models \varphi)$.
  Clearly, $\equiv_\Sigma$ is an equivalence relation
- A filtration of M via Σ is any model $M^f_\Sigma = (W^f, R^f, V^f)$ that satisfies
  - $W^f = \{|w|_{/\Sigma} \mid w \in W\}$.
  - if Rwv then $R^f|w||v|$.
  - if $R^f|w||v|$ then for all $\Diamond\varphi \in \Sigma$, if $M, v \models \varphi$ then $M, w \models \Diamond\varphi$.

A Recipe for Small Models: Filtrations

- Given $M = (W, R, V)$ a model and Σ a set of formulas closed under subformulas, define the relation $\equiv_\Sigma$ as:
  $w \equiv_\Sigma v$ iff for all φ in Σ: $(M, w \models \varphi$ iff $M, v \models \varphi)$.
  Clearly, $\equiv_\Sigma$ is an equivalence relation
- A filtration of M via Σ is any model $M^f_\Sigma = (W^f, R^f, V^f)$ that satisfies
  - $W^f = \{|w|_{/\Sigma} \mid w \in W\}$.
  - if Rwv then $R^f|w||v|$.
  - if $R^f|w||v|$ then for all $\Diamond\varphi \in \Sigma$, if $M, v \models \varphi$ then $M, w \models \Diamond\varphi$.
  - $V^f(p) = \{|w| \mid M, w \models p\}$, for any p in Σ.

An Example

An Example

- Let M be the model $(\mathbb{N}, R, V)$, where $R = \{(0,1), (0,2), (1,3)\} \cup \{(n, n+1) \mid n \geq 2\}$, and V is $V(p) = \mathbb{N} \setminus \{0\}$ and $V(q) = \{2\}$.

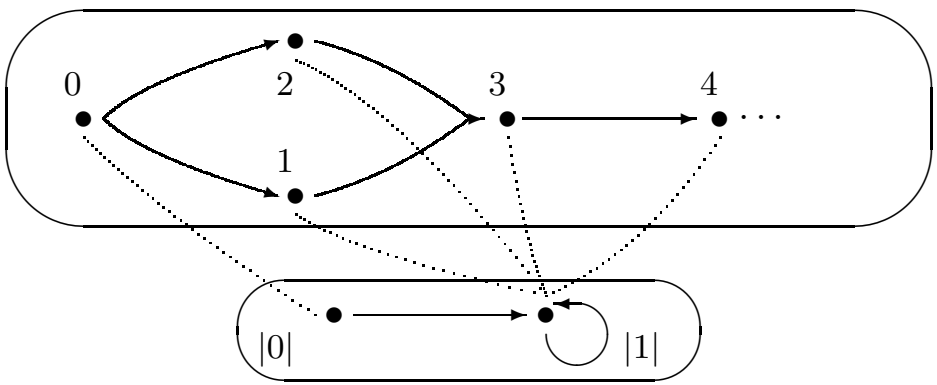
An Example

- Let M be the model $(\mathbb{N}, R, V)$, where $R = \{(0,1), (0,2), (1,3)\} \cup \{(n, n+1) \mid n \geq 2\}$, and V is $V(p) = \mathbb{N} \setminus \{0\}$ and $V(q) = \{2\}$.
- Let's take $\Sigma = \{\Diamond p, p\}$.

An Example

- Let M be the model $(\mathbb{N}, R, V)$, where $R = \{(0,1), (0,2), (1,3)\} \cup \{(n, n+1) \mid n \geq 2\}$, and V is $V(p) = \mathbb{N} \setminus \{0\}$ and $V(q) = \{2\}$.
- Let's take $\Sigma = \{\Diamond p, p\}$.
- The model $M^f_\Sigma = (W^f, R^f, V^f)$ with
  - $W^f = \{|0|, |1|\}$
  - $R^f = \{(|0|, |1|), (|1|, |1|)\}$
  - $V^f(p) = \{|1|\}$

  is a filtration of M via Σ.



A NEXPTIME Upper Bound

A NEXPTIME Upper Bound

- Thm: For any model M, if M^f is a filtration of M via Σ, then M^f has at most $2^{|\Sigma|}$ states

A NEXPTIME Upper Bound

- Thm: For any model M, if M^f is a filtration of M via Σ, then M^f has at most $2^{|\Sigma|}$ states
- Thm: Let M^f be a filtration of M via Σ. Then for any formula $\varphi \in \Sigma$, and for any node w in M, $M, w \models \varphi$ iff $M^f, |w| \models \varphi$.

A NEXPTIME Upper Bound

- Thm: For any model M, if M^f is a filtration of M via Σ, then M^f has at most $2^{|\Sigma|}$ states
- Thm: Let M^f be a filtration of M via Σ. Then for any formula $\varphi \in \Sigma$, and for any node w in M, $M, w \models \varphi$ iff $M^f, |w| \models \varphi$.
- These two results together give us a NEXPTIME algorithm for satisfiability:
  - guess the correct filtration
  - check that φ is satisfied in this structure

A NEXPTIME Upper Bound

- Thm: For any model M, if M^f is a filtration of M via Σ, then M^f has at most $2^{|\Sigma|}$ states
- Thm: Let M^f be a filtration of M via Σ. Then for any formula $\varphi \in \Sigma$, and for any node w in M, $M, w \models \varphi$ iff $M^f, |w| \models \varphi$.
- These two results together give us a NEXPTIME algorithm for satisfiability:
  - guess the correct filtration
  - check that φ is satisfied in this structure
- But for many modal logics we can obtain more effective algorithms: NP, PSPACE, EXPTIME

Undecidability

Undecidability

- One of the main problems of FO (from a computational point of view) is that deciding whether an FO formula is satisfiable is an undecidable problem.

Undecidability

- One of the main problems of FO (from a computational point of view) is that deciding whether an FO formula is satisfiable is an undecidable problem.
- How can we show that a given problem X is undecidable? One way:
  1. ask somebody (more intelligent than us) to show that some problem Y is undecidable
  2. prove that if X is decidable then Y would also be, providing a codification of Y in X

Undecidability

- One of the main problems of FO (from a computational point of view) is that deciding whether an FO formula is satisfiable is an undecidable problem.
- How can we show that a given problem X is undecidable? One way:
  1. ask somebody (more intelligent than us) to show that some problem Y is undecidable
  2. prove that if X is decidable then Y would also be, providing a codification of Y in X
- A class of problems called tiling problems are especially convenient for coding up satisfiability problems for certain logics

Tiling Problems

Tiling Problems

A tiling problem is a kind of jigsaw puzzle

Tiling Problems

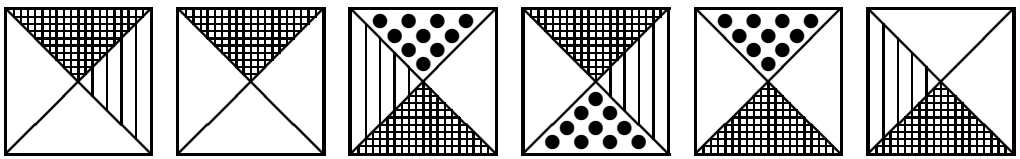
A tiling problem is a kind of jigsaw puzzle

- a tile T is a 1×1 square, fixed in orientation, with a fixed color in each side

Tiling Problems

A tiling problem is a kind of jigsaw puzzle

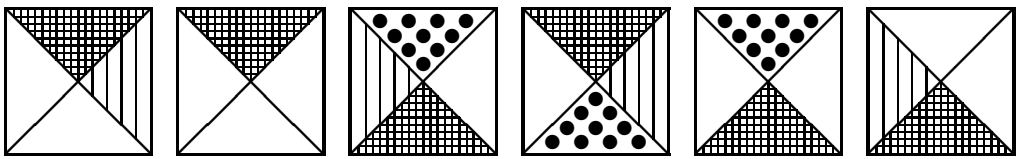
- a tile T is a 1×1 square, fixed in orientation, with a fixed color in each sideFor example, here we have six different kinds of tiles:



Tiling Problems

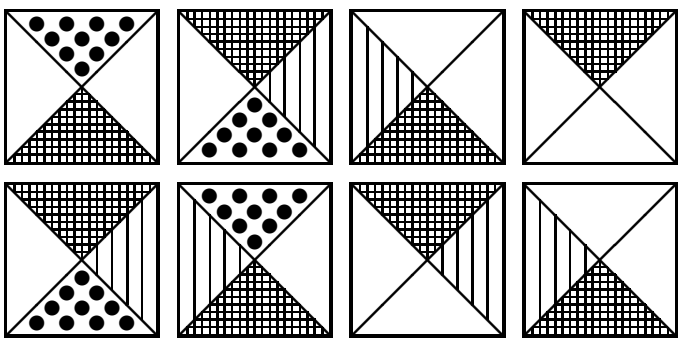
A tiling problem is a kind of jigsaw puzzle

- a tile T is a 1×1 square, fixed in orientation, with a fixed color in each sideFor example, here we have six different kinds of tiles:



- a simple tiling problem, could be:

  Is it possible to place tiles of the kind we show above on a grid of 2×4*, in such a way that we cover the entire grid and that adjacent tiles have the same color on neighboring sides?*



Tiling Problems

Tiling Problems

- The general form of a tiling problem

  Given a finite number of kind of tiles $\mathcal{T}$, can we cover a given part of $\mathbb{Z} \times \mathbb{Z}$ in such a way that adjacent tiles have the same color on the neighboring sides?

Tiling Problems

- The general form of a tiling problem

  Given a finite number of kind of tiles $\mathcal{T}$, can we cover a given part of $\mathbb{Z} \times \mathbb{Z}$ in such a way that adjacent tiles have the same color on the neighboring sides?

- In some cases, it is also possible to impose certain conditions on what is considered a correct tiling.

Tiling Problems

- The general form of a tiling problem

  Given a finite number of kind of tiles $\mathcal{T}$, can we cover a given part of $\mathbb{Z} \times \mathbb{Z}$ in such a way that adjacent tiles have the same color on the neighboring sides?

- In some cases, it is also possible to impose certain conditions on what is considered a correct tiling.
- Covering $\mathbb{N} \times \mathbb{N}$
  - tiling $\mathbb{N} \times \mathbb{N}$: Given a finite set of tiles $\mathcal{T}$, can $\mathcal{T}$ cover $\mathbb{N} \times \mathbb{N}$?
  - this problem is undecidable (It can be proved equivalent to the halting problem of Turing machines. See David Harel, Algorithms. The Essence of Computing for an excellent explanation.)

Coding a Tiling of the Grid in FO

Coding a Tiling of the Grid in FO

- Thm: The problem of deciding whether a given FO formula is satisfiable is undecidable.

Coding a Tiling of the Grid in FO

- Thm: The problem of deciding whether a given FO formula is satisfiable is undecidable.
  Proof: [Gurevich, 1976] For $h, v, t_1, \ldots, t_n$ unary function symbols. Let φ be the conjunction of the formulas:

$$h(v(x)) = v(h(x)),$$

Coding a Tiling of the Grid in FO

- Thm: The problem of deciding whether a given FO formula is satisfiable is undecidable.
  Proof: [Gurevich, 1976] For $h, v, t_1, \ldots, t_n$ unary function symbols. Let φ be the conjunction of the formulas:

$$h(v(x)) = v(h(x)),$$
$$\bigvee\{t_i(x) = x \mid 1 \leq i \leq n\},$$

Coding a Tiling of the Grid in FO

- Thm: The problem of deciding whether a given FO formula is satisfiable is undecidable.
  Proof: [Gurevich, 1976] For $h, v, t_1, \ldots, t_n$ unary function symbols. Let φ be the conjunction of the formulas:

$$h(v(x)) = v(h(x)),$$
$$\bigvee\{t_i(x) = x \mid 1 \leq i \leq n\},$$
$$\bigwedge\{t_i(x) = x \rightarrow \neg(t_j(x) = x) \mid 1 \leq i < j \leq n\},$$

Coding a Tiling of the Grid in FO

- Thm: The problem of deciding whether a given FO formula is satisfiable is undecidable.
  Proof: [Gurevich, 1976] For $h, v, t_1, \ldots, t_n$ unary function symbols. Let φ be the conjunction of the formulas:

$$h(v(x)) = v(h(x)),$$
$$\bigvee\{t_i(x) = x \mid 1 \leq i \leq n\},$$
$$\bigwedge\{t_i(x) = x \rightarrow \neg(t_j(x) = x) \mid 1 \leq i < j \leq n\},$$
$$\bigvee\{t_i(x) = x \wedge t_j(h(x)) = h(x) \mid H(t_i, t_j)\},$$

Coding a Tiling of the Grid in FO

- Thm: The problem of deciding whether a given FO formula is satisfiable is undecidable.
  Proof: [Gurevich, 1976] For $h, v, t_1, \ldots, t_n$ unary function symbols. Let φ be the conjunction of the formulas:

$$
\begin{gathered}
h(v(x)) = v(h(x)), \\
\bigvee\{t_i(x) = x \mid 1 \leq i \leq n\}, \\
\bigwedge\{t_i(x) = x \rightarrow \neg(t_j(x) = x) \mid 1 \leq i < j \leq n\}, \\
\bigvee\{t_i(x) = x \wedge t_j(h(x)) = h(x) \mid H(t_i, t_j)\}, \\
\bigvee\{t_i(x) = x \wedge t_j(v(x)) = v(x) \mid V(t_i, t_j)\}.
\end{gathered}
$$

Coding a Tiling of the Grid in FO

- Thm: The problem of deciding whether a given FO formula is satisfiable is undecidable.
  Proof: [Gurevich, 1976] For $h, v, t_1, \ldots, t_n$ unary function symbols. Let φ be the conjunction of the formulas:

$$
\begin{gathered}
h(v(x)) = v(h(x)), \\
\bigvee\{t_i(x) = x \mid 1 \leq i \leq n\}, \\
\bigwedge\{t_i(x) = x \rightarrow \neg(t_j(x) = x) \mid 1 \leq i < j \leq n\}, \\
\bigvee\{t_i(x) = x \wedge t_j(h(x)) = h(x) \mid H(t_i, t_j)\}, \\
\bigvee\{t_i(x) = x \wedge t_j(v(x)) = v(x) \mid V(t_i, t_j)\}.
\end{gathered}
$$

  Then $\forall x.\varphi(x)$ is satisfiable iff $\mathcal{T}$ covers $\mathbb{N} \times \mathbb{N}$.

Tiling for $\downarrow$

- We will show that we can do the same "trick" that we did for FO but using only the $\downarrow$ operator.

Tiling for $\downarrow$

- We will show that we can do the same "trick" that we did for FO but using only the $\downarrow$ operator.
- But remember that we proved that $\downarrow$ was 'local,' how are we going to code the whole $\mathbb{N} \times \mathbb{N}$???!!!!

Tiling for $\downarrow$

- We will show that we can do the same "trick" that we did for FO but using only the $\downarrow$ operator.
- But remember that we proved that $\downarrow$ was 'local,' how are we going to code the whole $\mathbb{N} \times \mathbb{N}$???!!!!
- The answer: Step by step, using a Spypoint Model

Tiling for $\downarrow$

- We will show that we can do the same "trick" that we did for FO but using only the $\downarrow$ operator.
- But remember that we proved that $\downarrow$ was 'local,' how are we going to code the whole $\mathbb{N} \times \mathbb{N}$???!!!!
- The answer: Step by step, using a Spypoint Model

  $S_1 \quad x \wedge \neg\langle S\rangle x \wedge \langle S\rangle\neg x \wedge [S]\langle S\rangle x \wedge [S][S]x$

Tiling for $\downarrow$

- We will show that we can do the same "trick" that we did for FO but using only the $\downarrow$ operator.
- But remember that we proved that $\downarrow$ was 'local,' how are we going to code the whole $\mathbb{N} \times \mathbb{N}$???!!!!
- The answer: Step by step, using a Spypoint Model

S_1 $\quad x \wedge \neg\langle S\rangle x \wedge \langle S\rangle\neg x \wedge [S]\langle S\rangle x \wedge [S][S]x$

S_2 $\quad [S][U]\neg x \wedge [S][R]\neg x$

Tiling for $\downarrow$

- We will show that we can do the same "trick" that we did for FO but using only the $\downarrow$ operator.
- But remember that we proved that $\downarrow$ was 'local,' how are we going to code the whole $\mathbb{N} \times \mathbb{N}$???!!!!
- The answer: Step by step, using a Spypoint Model

$$
\begin{array}{ll}
S_1 & x \wedge \neg\langle S\rangle x \wedge \langle S\rangle\neg x \wedge [S]\langle S\rangle x \wedge [S][S]x \\
S_2 & [S][U]\neg x \wedge [S][R]\neg x \\
D & [S]\langle U\rangle\top \wedge [S]\langle R\rangle\top
\end{array}
$$

Tiling for $\downarrow$

- We will show that we can do the same "trick" that we did for FO but using only the $\downarrow$ operator.
- But remember that we proved that $\downarrow$ was 'local,' how are we going to code the whole $\mathbb{N} \times \mathbb{N}$???!!!!
- The answer: Step by step, using a Spypoint Model

S_1	$x \wedge \neg\langle S\rangle x \wedge \langle S\rangle\neg x \wedge [S]\langle S\rangle x \wedge [S][S]x$	
S_2	$[S][U]\neg x \wedge [S][R]\neg x$	
D	$[S]\langle U\rangle\top \wedge [S]\langle R\rangle\top$	
$C^{\dagger}$	$[S][\dagger]{\downarrow}y.\langle S\rangle\langle S\rangle y$	for $\dagger \in \{U, R\}$

Tiling for $\downarrow$

- We will show that we can do the same "trick" that we did for FO but using only the $\downarrow$ operator.
- But remember that we proved that $\downarrow$ was 'local,' how are we going to code the whole $\mathbb{N} \times \mathbb{N}$???!!!!
- The answer: Step by step, using a Spypoint Model

S_1	$x \wedge \neg\langle S\rangle x \wedge \langle S\rangle\neg x \wedge [S]\langle S\rangle x \wedge [S][S]x$	
S_2	$[S][U]\neg x \wedge [S][R]\neg x$	
D	$[S]\langle U\rangle\top \wedge [S]\langle R\rangle\top$	
$C^{\dagger}$	$[S][\dagger]\downarrow y.\langle S\rangle\langle S\rangle y$	for $\dagger \in \{U, R\}$
$P^{\dagger}$	$[S]\downarrow y.\langle\dagger\rangle\downarrow z.\langle S\rangle\langle S\rangle(y \wedge [\dagger]z)$	for $\dagger \in \{U, R\}$

Tiling for $\downarrow$

- We will show that we can do the same "trick" that we did for FO but using only the $\downarrow$ operator.
- But remember that we proved that $\downarrow$ was 'local,' how are we going to code the whole $\mathbb{N} \times \mathbb{N}$???!!!!
- The answer: Step by step, using a Spypoint Model

S_1	$x \wedge \neg\langle S\rangle x \wedge \langle S\rangle\neg x \wedge [S]\langle S\rangle x \wedge [S][S]x$	
S_2	$[S][U]\neg x \wedge [S][R]\neg x$	
D	$[S]\langle U\rangle\top \wedge [S]\langle R\rangle\top$	
$C^{\dagger}$	$[S][\dagger]{\downarrow}y.\langle S\rangle\langle S\rangle y$	for $\dagger \in \{U, R\}$
$P^{\dagger}$	$[S]{\downarrow}y.\langle\dagger\rangle{\downarrow}z.\langle S\rangle\langle S\rangle(y \wedge [\dagger]z)$	for $\dagger \in \{U, R\}$
G	$[S]{\downarrow}y.\langle U\rangle\langle R\rangle{\downarrow}z.\langle S\rangle\langle S\rangle(y \wedge \langle R\rangle\langle U\rangle z)$	

Tiling for $\downarrow$

- We will show that we can do the same "trick" that we did for FO but using only the $\downarrow$ operator.
- But remember that we proved that $\downarrow$ was 'local,' how are we going to code the whole $\mathbb{N} \times \mathbb{N}$???!!!!
- The answer: Step by step, using a Spypoint Model

S_1	$x \wedge \neg\langle S\rangle x \wedge \langle S\rangle\neg x \wedge [S]\langle S\rangle x \wedge [S][S]x$	
S_2	$[S][U]\neg x \wedge [S][R]\neg x$	
D	$[S]\langle U\rangle\top \wedge [S]\langle R\rangle\top$	
$C^{\dagger}$	$[S][\dagger]{\downarrow}y.\langle S\rangle\langle S\rangle y$	for $\dagger \in \{U, R\}$
$P^{\dagger}$	$[S]{\downarrow}y.\langle\dagger\rangle{\downarrow}z.\langle S\rangle\langle S\rangle(y \wedge [\dagger]z)$	for $\dagger \in \{U, R\}$
G	$[S]{\downarrow}y.\langle U\rangle\langle R\rangle{\downarrow}z.\langle S\rangle\langle S\rangle(y \wedge \langle R\rangle\langle U\rangle z)$	
O	$[S]((\bigvee_{i=1}^{k} T_i) \wedge \bigwedge_{1\leq i<j\leq k}(T_i \rightarrow \neg T_j))$	

Tiling for $\downarrow$

- We will show that we can do the same "trick" that we did for FO but using only the $\downarrow$ operator.
- But remember that we proved that $\downarrow$ was 'local,' how are we going to code the whole $\mathbb{N} \times \mathbb{N}$???!!!!
- The answer: Step by step, using a Spypoint Model

S_1	$x \wedge \neg\langle S\rangle x \wedge \langle S\rangle\neg x \wedge [S]\langle S\rangle x \wedge [S][S]x$	
S_2	$[S][U]\neg x \wedge [S][R]\neg x$	
D	$[S]\langle U\rangle\top \wedge [S]\langle R\rangle\top$	
$C^{\dagger}$	$[S][\dagger]{\downarrow}y.\langle S\rangle\langle S\rangle y$	for $\dagger \in \{U, R\}$
$P^{\dagger}$	$[S]{\downarrow}y.\langle\dagger\rangle{\downarrow}z.\langle S\rangle\langle S\rangle(y \wedge [\dagger]z)$	for $\dagger \in \{U, R\}$
G	$[S]{\downarrow}y.\langle U\rangle\langle R\rangle{\downarrow}z.\langle S\rangle\langle S\rangle(y \wedge \langle R\rangle\langle U\rangle z)$	
O	$[S]((\bigvee_{i=1}^{k} T_i) \wedge \bigwedge_{1\leq i<j\leq k}(T_i \rightarrow \neg T_j))$	
V	$[S](\bigvee_{(t_i,t_j)\in V}(t_i \wedge \langle U\rangle T_j))$	

Tiling for $\downarrow$

- We will show that we can do the same "trick" that we did for FO but using only the $\downarrow$ operator.
- But remember that we proved that $\downarrow$ was 'local,' how are we going to code the whole $\mathbb{N} \times \mathbb{N}$???!!!!
- The answer: Step by step, using a Spypoint Model

S_1	$x \wedge \neg\langle S\rangle x \wedge \langle S\rangle\neg x \wedge [S]\langle S\rangle x \wedge [S][S]x$	
S_2	$[S][U]\neg x \wedge [S][R]\neg x$	
D	$[S]\langle U\rangle\top \wedge [S]\langle R\rangle\top$	
$C^{\dagger}$	$[S][\dagger]{\downarrow}y.\langle S\rangle\langle S\rangle y$	for $\dagger \in \{U, R\}$
$P^{\dagger}$	$[S]{\downarrow}y.\langle\dagger\rangle{\downarrow}z.\langle S\rangle\langle S\rangle(y \wedge [\dagger]z)$	for $\dagger \in \{U, R\}$
G	$[S]{\downarrow}y.\langle U\rangle\langle R\rangle{\downarrow}z.\langle S\rangle\langle S\rangle(y \wedge \langle R\rangle\langle U\rangle z)$	
O	$[S]((\bigvee_{i=1}^{k} T_i) \wedge \bigwedge_{1\leq i<j\leq k}(T_i \rightarrow \neg T_j))$	
V	$[S](\bigvee_{(t_i,t_j)\in V}(t_i \wedge \langle U\rangle T_j))$	
H	$[S](\bigvee_{(t_i,t_j)\in H}(t_i \wedge \langle R\rangle T_j))$	

Tiling for $\downarrow$

- We will show that we can do the same "trick" that we did for FO but using only the $\downarrow$ operator.
- But remember that we proved that $\downarrow$ was 'local,' how are we going to code the whole $\mathbb{N} \times \mathbb{N}$???!!!!
- The answer: Step by step, using a Spypoint Model

S_1	$x \wedge \neg\langle S\rangle x \wedge \langle S\rangle\neg x \wedge [S]\langle S\rangle x \wedge [S][S]x$	
S_2	$[S][U]\neg x \wedge [S][R]\neg x$	
D	$[S]\langle U\rangle\top \wedge [S]\langle R\rangle\top$	
$C^\dagger$	$[S][\dagger]\downarrow y.\langle S\rangle\langle S\rangle y$	for $\dagger \in \{U, R\}$
$P^\dagger$	$[S]\downarrow y.\langle\dagger\rangle\downarrow z.\langle S\rangle\langle S\rangle(y \wedge [\dagger]z)$	for $\dagger \in \{U, R\}$
G	$[S]\downarrow y.\langle U\rangle\langle R\rangle\downarrow z.\langle S\rangle\langle S\rangle(y \wedge \langle R\rangle\langle U\rangle z)$	
O	$[S]((\bigvee_{i=1}^{k} T_i) \wedge \bigwedge_{1\leq i<j\leq k}(T_i \rightarrow \neg T_j))$	
V	$[S](\bigvee_{(t_i,t_j)\in V}(t_i \wedge \langle U\rangle T_j))$	
H	$[S](\bigvee_{(t_i,t_j)\in H}(t_i \wedge \langle R\rangle T_j))$	

Let $\varphi^{\mathcal{T}}$ be $\downarrow x.(S_1 \wedge S_2 \wedge D \wedge C^\dagger \wedge P^\dagger \wedge G \wedge O \wedge V \wedge H)$.

Tiling for $\downarrow$

Tiling for $\downarrow$

Thm: $\mathcal{T}$ tiles $\mathbb{N} \times \mathbb{N}$ iff $\varphi^{\mathcal{T}}$ is satisfiable.

Tiling for $\downarrow$

Thm: $\mathcal{T}$ tiles $\mathbb{N} \times \mathbb{N}$ iff $\varphi^{\mathcal{T}}$ is satisfiable.

Proof [$\Rightarrow$] Quite easy, if we have a tiling for $\mathcal{T}$, we take the grid and add a point s which is properly connected using an S relation. Then $\varphi^{\mathcal{T}}$ will be satisfied at s.

Tiling for $\downarrow$

Thm: $\mathcal{T}$ tiles $\mathbb{N} \times \mathbb{N}$ iff $\varphi^{\mathcal{T}}$ is satisfiable.

Proof [$\Rightarrow$] Quite easy, if we have a tiling for $\mathcal{T}$, we take the grid and add a point s which is properly connected using an S relation. Then $\varphi^{\mathcal{T}}$ will be satisfied at s.

[$\Leftarrow$] A bit harder but not much. Take a model satisfying $\varphi^{\mathcal{T}}$. Let G be the set of elements $G = \{g \mid sSg\}$. G is going to be our 'grid' (i.e., an abstract representation of $\mathbb{N} \times \mathbb{N}$). It is easy to see that $G \neq \emptyset$, and that $s \notin G$.

Tiling for $\downarrow$

Thm: $\mathcal{T}$ tiles $\mathbb{N} \times \mathbb{N}$ iff $\varphi^{\mathcal{T}}$ is satisfiable.

Proof [$\Rightarrow$] Quite easy, if we have a tiling for $\mathcal{T}$, we take the grid and add a point s which is properly connected using an S relation. Then $\varphi^{\mathcal{T}}$ will be satisfied at s.

[$\Leftarrow$] A bit harder but not much. Take a model satisfying $\varphi^{\mathcal{T}}$. Let G be the set of elements $G = \{g \mid sSg\}$. G is going to be our 'grid' (i.e., an abstract representation of $\mathbb{N} \times \mathbb{N}$). It is easy to see that $G \neq \emptyset$, and that $s \notin G$.

Let g_0 be an arbitrary element of G. Let $f : \mathbb{N} \times \mathbb{N} \to M$ be such that

$$f(0,0) = g_0 \qquad U(f(n,m), f(n,m+1)) \qquad R(f(n,m), f(n+1,m))$$

Tiling for $\downarrow$

Thm: $\mathcal{T}$ tiles $\mathbb{N} \times \mathbb{N}$ iff $\varphi^{\mathcal{T}}$ is satisfiable.

Proof [$\Rightarrow$] Quite easy, if we have a tiling for $\mathcal{T}$, we take the grid and add a point s which is properly connected using an S relation. Then $\varphi^{\mathcal{T}}$ will be satisfied at s.

[$\Leftarrow$] A bit harder but not much. Take a model satisfying $\varphi^{\mathcal{T}}$. Let G be the set of elements $G = \{g \mid sSg\}$. G is going to be our 'grid' (i.e., an abstract representation of $\mathbb{N} \times \mathbb{N}$). It is easy to see that $G \neq \emptyset$, and that $s \notin G$.

Let g_0 be an arbitrary element of G. Let $f : \mathbb{N} \times \mathbb{N} \to M$ be such that

$$f(0,0) = g_0 \qquad U(f(n,m), f(n,m+1)) \qquad R(f(n,m), f(n+1,m))$$

Now define the tiling function $t : \mathbb{N} \times \mathbb{N} \to T$ as

$$f(n,m) = t_i \text{ iff } \mathcal{M}, f(n,m) \models T_i.$$

The Basic Hybrid Logic is in PSPACE

The Basic Hybrid Logic is in PSPACE

- We are going to sketch the proof of containment in PSPACE for the basic hybrid logic.

The Basic Hybrid Logic is in PSPACE

- We are going to sketch the proof of containment in PSPACE for the basic hybrid logic.
- Perhaps the details of the proof are scary (after all, it's past 8pm on a Thursday of a looooooooooong hybrid logic week).

The Basic Hybrid Logic is in PSPACE

- We are going to sketch the proof of containment in PSPACE for the basic hybrid logic.
- Perhaps the details of the proof are scary (after all, it's past 8pm on a Thursday of a looooooooooong hybrid logic week).
- But the intuitions are simple and beautiful!

The Basic Hybrid Logic is in PSPACE

- We are going to sketch the proof of containment in PSPACE for the basic hybrid logic.
- Perhaps the details of the proof are scary (after all, it's past 8pm on a Thursday of a looooooooooong hybrid logic week).
- But the intuitions are simple and beautiful!
- And the technique is very general and used a lot in the area.

The Basic Hybrid Logic is in PSPACE

- We are going to sketch the proof of containment in PSPACE for the basic hybrid logic.
- Perhaps the details of the proof are scary (after all, it's past 8pm on a Thursday of a looooooooooong hybrid logic week).
- But the intuitions are simple and beautiful!
- And the technique is very general and used a lot in the area.
- My advice: Take it easy. Try to follow what you can and understand how the trick works.

The Basic Hybrid Logic is in PSPACE

- We are going to sketch the proof of containment in PSPACE for the basic hybrid logic.
- Perhaps the details of the proof are scary (after all, it's past 8pm on a Thursday of a looooooooooong hybrid logic week).
- But the intuitions are simple and beautiful!
- And the technique is very general and used a lot in the area.
- My advice: Take it easy. Try to follow what you can and understand how the trick works.
  If you are interested you can later go for the details in the paper.

∀belard and ∃loise play games

- We will define a game between two players called ∃loise and ∀belard.

∀belard and ∃loise play games

- We will define a game between two players called ∃loise and ∀belard.
- You can also think of it as an exam: ∃loise is the student and ∀belard the strict professor of Hybrid Logic.

∀belard and ∃loise play games

- We will define a game between two players called ∃loise and ∀belard.
- You can also think of it as an exam: ∃loise is the student and ∀belard the strict professor of Hybrid Logic.
- The general structure of the game/exam is as follows.

∀belard and ∃loise play games

- We will define a game between two players called ∃loise and ∀belard.
- You can also think of it as an exam: ∃loise is the student and ∀belard the strict professor of Hybrid Logic.
- The general structure of the game/exam is as follows.
  - At the start, ∀belard choses a question: a satisfiable formula φ of the hybrid modal logic.

∀belard and ∃loise play games

- We will define a game between two players called ∃loise and ∀belard.
- You can also think of it as an exam: ∃loise is the student and ∀belard the strict professor of Hybrid Logic.
- The general structure of the game/exam is as follows.
  - At the start, ∀belard choses a question: a satisfiable formula φ of the hybrid modal logic.
  - The final task of ∃loise is to build a model for φ.

∀belard and ∃loise play games

- We will define a game between two players called ∃loise and ∀belard.
- You can also think of it as an exam: ∃loise is the student and ∀belard the strict professor of Hybrid Logic.
- The general structure of the game/exam is as follows.
  - At the start, ∀belard choses a question: a satisfiable formula φ of the hybrid modal logic.
  - The final task of ∃loise is to build a model for φ.
  - ∃loise can make mistakes, but after all ∀belard is a nice professor and he wants ∃loise to learn and aprove the exam.

∀belard and ∃loise play games

- We will define a game between two players called ∃loise and ∀belard.
- You can also think of it as an exam: ∃loise is the student and ∀belard the strict professor of Hybrid Logic.
- The general structure of the game/exam is as follows.
  - At the start, ∀belard choses a question: a satisfiable formula φ of the hybrid modal logic.
  - The final task of ∃loise is to build a model for φ.
  - ∃loise can make mistakes, but after all ∀belard is a nice professor and he wants ∃loise to learn and aprove the exam.
  - Hence, ∀belard will point possible mistakes and ∃loise will have to show that they are not mistakes, or repair it.

∀belard and ∃loise play games

- We will define a game between two players called ∃loise and ∀belard.
- You can also think of it as an exam: ∃loise is the student and ∀belard the strict professor of Hybrid Logic.
- The general structure of the game/exam is as follows.
  - At the start, ∀belard choses a question: a satisfiable formula φ of the hybrid modal logic.
  - The final task of ∃loise is to build a model for φ.
  - ∃loise can make mistakes, but after all ∀belard is a nice professor and he wants ∃loise to learn and aprove the exam.
  - Hence, ∀belard will point possible mistakes and ∃loise will have to show that they are not mistakes, or repair it.
  - If ∃loise can deal with any question, and correct all possible mistakes pointed by ∀belard, then whe will aprove the exam.

Alternating Turing Machines (ATM)

Alternating Turing Machines (ATM)

- What we are actually doing (in disguise) is to write the program of an Alternating Turing Machine.

Alternating Turing Machines (ATM)

- What we are actually doing (in disguise) is to write the program of an Alternating Turing Machine.
- We will show that the machine can decide satisfiability of hybrid logics in polynomial time (i.e., each game played by $\forall$belard and $\exists$loise will only take polynomial time).

Alternating Turing Machines (ATM)

- What we are actually doing (in disguise) is to write the program of an Alternating Turing Machine.
- We will show that the machine can decide satisfiability of hybrid logics in polynomial time (i.e., each game played by $\forall$belard and $\exists$loise will only take polynomial time).
- Because PTIME Alternating Turing Machines can be simulated by normal Turing Machines using only polynomial space, we end up proving the complexity result we want.

The basic hybrid logic is in PSPACE

The basic hybrid logic is in PSPACE

- Def: Given a formula φ, let $\mathsf{SF}^{+}(\varphi)$ be the set of subformulas of φ closed under single negation.

The basic hybrid logic is in PSPACE

- Def: Given a formula φ, let $\mathsf{SF}^+(\varphi)$ be the set of subformulas of φ closed under single negation.
- Def: A Hintikka set for φ is a maximal consisten subset of $\mathsf{SF}^+(\varphi)$.
- Given a formula φ, $\exists$loise will answer questions from $\forall$belard by placing Hintikka set on the board (notice that the size of any Hintikka set is only polynomial in he size of φ).
- From now one, we consider φ fixed.

The Rules of the Game

The Rules of the Game

∃loise starts by playing a collection $\{X_0, \ldots, X_k\}$ of Hintikka sets and specifying a relation R on them. ∃loise loses immediately if one of the following conditions is false:

The Rules of the Game

∃loise starts by playing a collection $\{X_0, \ldots, X_k\}$ of Hintikka sets and specifying a relation R on them. ∃loise loses immediately if one of the following conditions is false:

1. $\varphi \in X_0$ and all others X_l contain at least one of the nominals of φ.

The Rules of the Game

∃loise starts by playing a collection $\{X_0, \ldots, X_k\}$ of Hintikka sets and specifying a relation R on them. ∃loise loses immediately if one of the following conditions is false:

1. $\varphi \in X_0$ and all others X_l contain at least one of the nominals of φ.
2. no nominal occurs in two different Hintikka sets.

The Rules of the Game

∃loise starts by playing a collection $\{X_0, \ldots, X_k\}$ of Hintikka sets and specifying a relation R on them. ∃loise loses immediately if one of the following conditions is false:

1. $\varphi \in X_0$ and all others X_l contain at least one of the nominals of φ.
2. no nominal occurs in two different Hintikka sets.
3. for all X_l, $@_i\psi \in \mathrm{SF}^+(\varphi)$, $@_i\psi \in X_l$ iff $\{i, \psi\} \subseteq X_k$, for some k.

The Rules of the Game

∃loise starts by playing a collection $\{X_0, \ldots, X_k\}$ of Hintikka sets and specifying a relation R on them. ∃loise loses immediately if one of the following conditions is false:

1. $\varphi \in X_0$ and all others X_l contain at least one of the nominals of φ.
2. no nominal occurs in two different Hintikka sets.
3. for all X_l, $@_i\psi \in \mathsf{SF}^+(\varphi)$, $@_i\psi \in X_l$ iff $\{i, \psi\} \subseteq X_k$, for some k.
4. for all $\Diamond\psi \in \mathsf{SF}^+(\varphi)$, if RX_lX_k and $\Diamond\psi \notin X_l$, then $\psi \notin X_k$.

The Rules of the Game

∃loise starts by playing a collection $\{X_0, \ldots, X_k\}$ of Hintikka sets and specifying a relation R on them. ∃loise loses immediately if one of the following conditions is false:

1. $\varphi \in X_0$ and all others X_l contain at least one of the nominals of φ.
2. no nominal occurs in two different Hintikka sets.
3. for all X_l, $@_i\psi \in \mathsf{SF}^+(\varphi)$, $@_i\psi \in X_l$ iff $\{i, \psi\} \subseteq X_k$, for some k.
4. for all $\Diamond\psi \in \mathsf{SF}^+(\varphi)$, if RX_lX_k and $\Diamond\psi \notin X_l$, then $\psi \notin X_k$.

Now ∀belard may choose an X_l and a "mistake" $\Diamond\theta \in X_l$. ∃loise must respond with a Hintikka set Y such that

1. $\theta \in Y$ and for all $\Diamond\psi \in \mathsf{SF}^+(\varphi)$, $\Diamond\psi \notin X_l$ implies that $\psi \notin Y$.
2. for all $@_i\psi \in \mathsf{SF}^+(\varphi)$, $@_i\psi \in Y$ iff $\{i, \psi\} \subseteq X_k$, for some k.
3. if $i \in Y$ for some nominal i, then Y is one of the Hintikka sets she played at the start. In this case the game stops and ∃loise wins.

Terminating Conditions

Terminating Conditions

- We just saw a case in which the game finished (when $\exists$loise plays a set with a nominal).

Terminating Conditions

- We just saw a case in which the game finished (when ∃loise plays a set with a nominal).
- If ∃loise cannot find a suitable Y, the game stops and she fails the exam.

Terminating Conditions

- We just saw a case in which the game finished (when $\exists$loise plays a set with a nominal).
- If $\exists$loise cannot find a suitable Y, the game stops and she fails the exam.
- If $\exists$loise does find a suitable Y (and Y doesn't have anominal) then Y is added to the list of played sets, and play continues.

Terminating Conditions

- We just saw a case in which the game finished (when $\exists$loise plays a set with a nominal).
- If $\exists$loise cannot find a suitable Y, the game stops and she fails the exam.
- If $\exists$loise does find a suitable Y (and Y doesn't have anominal) then Y is added to the list of played sets, and play continues.
- $\forall$belard must now choose a defect $\Diamond\theta$ from the last played Hintikka set with the following restriction:

Terminating Conditions

- We just saw a case in which the game finished (when ∃loise plays a set with a nominal).
- If ∃loise cannot find a suitable Y, the game stops and she fails the exam.
- If ∃loise does find a suitable Y (and Y doesn't have anominal) then Y is added to the list of played sets, and play continues.
- ∀belard must now choose a defect $\Diamond\theta$ from the last played Hintikka set with the following restriction:
  ∀belard should always chose a formula of lower modal depth than in the previous round

Terminating Conditions

- We just saw a case in which the game finished (when ∃loise plays a set with a nominal).
- If ∃loise cannot find a suitable Y, the game stops and she fails the exam.
- If ∃loise does find a suitable Y (and Y doesn't have anominal) then Y is added to the list of played sets, and play continues.
- ∀belard must now choose a defect $\Diamond\theta$ from the last played Hintikka set with the following restriction:
  ∀belard should always chose a formula of lower modal depth than in the previous round
  (We said that ∀belard was nice, the questions he asks ∃loise becomes easier and easier).

The PSPACE Theorem

Thm: φ is satisfiable if and only if $\exists$loise has a winning strategy.

The PSPACE Theorem

Thm: φ is satisfiable if and only if $\exists$loise has a winning strategy.

Proof: [$\Rightarrow$] Very easy. If φ is satisfiable, then $\exists$loise can use any model of φ to guide her moves and she will surely answer correctly any questin posed by $\forall$belard.

The PSPACE Theorem

Thm: φ is satisfiable if and only if $\exists$loise has a winning strategy.

Proof: [$\Rightarrow$] Very easy. If φ is satisfiable, then $\exists$loise can use any model of φ to guide her moves and she will surely answer correctly any questin posed by $\forall$belard.
[$\Leftarrow$] A model constructin argument. We use $\exists$loise winning strategy to build a model for φ.

The PSPACE Theorem

Thm: φ is satisfiable if and only if $\exists$loise has a winning strategy.

Proof: [$\Rightarrow$] Very easy. If φ is satisfiable, then $\exists$loise can use any model of φ to guide her moves and she will surely answer correctly any questin posed by $\forall$belard.
[$\Leftarrow$] A model constructin argument. We use $\exists$loise winning strategy to build a model for φ.

- Notice that we only use polynomial time in our game (there are only $|\varphi|$ rounds, and in each round $\exists$loise has to put in the board information of polynomial size)

The PSPACE Theorem

Thm: φ is satisfiable if and only if $\exists$loise has a winning strategy.

Proof: [$\Rightarrow$] Very easy. If φ is satisfiable, then $\exists$loise can use any model of φ to guide her moves and she will surely answer correctly any questin posed by $\forall$belard.
[$\Leftarrow$] A model constructin argument. We use $\exists$loise winning strategy to build a model for φ.

- Notice that we only use polynomial time in our game (there are only $|\varphi|$ rounds, and in each round $\exists$loise has to put in the board information of polynomial size)
- Hence, the satisfiability problem of the basic hybrid logic is in PSPACE.