# LogicS

## Lecture #8: Putting Tiles in an Infinite Bathroom

Carlos Areces and Patrick Blackburn

{carlos.areces,patrick.blackburn}@loria.fr

INRIA Nancy Grand Est
Nancy, France

NASSLLI 2010 - Bloomington - USA

# The Story so Far

- We put together various bits-and pieces we have played with (names, :, diamonds) added the $\langle\langle x\rangle\rangle$ and $[\![x]\!]$ operators, and reached the most expressive language we have seen so far.
- We reached "first-order logic", the logic often considered to be "classical logic", and one of the most distinctive and important spots on the logical landscape. For our $\langle\langle x\rangle\rangle$ and $[\![x]\!]$ are really just the familiar $\exists$ and $\forall$ quantifiers.
- We've come a long way. Think that on Monday we were in propositional logic!

# What do we do Today

## What do we do Today

- We'll talk about the satisfiability problem of first-order logic.

## What do we do Today

- We'll talk about the satisfiability problem of first-order logic.
- We will argue that the problem is undecidable

# What do we do Today

- We'll talk about the satisfiability problem of first-order logic.
- We will argue that the problem is undecidable
  - That is, there is no algorithm that can answer for an arbitrary formula of first order logic, whether the formula has a model or not.

## What do we do Today

- We'll talk about the satisfiability problem of first-order logic.
- We will argue that the problem is undecidable
  - That is, there is no algorithm that can answer for an arbitrary formula of first order logic, whether the formula has a model or not.
- Actually, we are going to show how to tile an infinite bathroom.

# (Un)Decidability

# (Un)Decidability

How can we prove that problem X is undecidable?

# (Un)Decidability

How can we prove that problem X is undecidable?
One way is

- Ask somebody (more intelligent than us) to prove that some problem Y is undecidable

# (Un)Decidability

How can we prove that problem X is undecidable?
One way is

- Ask somebody (more intelligent than us) to prove that some problem Y is undecidable
- Prove that if X would be decidable then Y would be decidable, giving a codification of Y into X.

# (Un)Decidability

How can we prove that problem X is undecidable?
One way is

- Ask somebody (more intelligent than us) to prove that some problem Y is undecidable
- Prove that if X would be decidable then Y would be decidable, giving a codification of Y into X.

The halting problem of Turing machines is the standard example of an undecidable problem. The behaviour of a Turing machine, and the predicate that says that a given turing machine stops on all inputs can be expressed in FO.

# Tiling Problems

## Tiling Problems

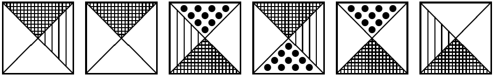
- A tiling problem is a kind of jigsaw puzzle

# Tiling Problems

- A tiling problem is a kind of jigsaw puzzle
- a tile $T$ is a $1 \times 1$ square, fixed in orientation, with a fixed color in each side

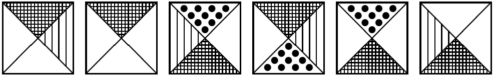
# Tiling Problems

- A tiling problem is a kind of jigsaw puzzle
- a tile $T$ is a $1 \times 1$ square, fixed in orientation, with a fixed color in each side
- for 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 $1 \times 1$ square, fixed in orientation, with a fixed color in each side
- for example, here we have six different kinds of tiles:

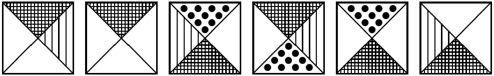


- a simple tiling problem, could be:

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

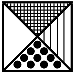
# Tiling Problems

- A tiling problem is a kind of jigsaw puzzle
- a tile $T$ is a $1 \times 1$ square, fixed in orientation, with a fixed color in each side
- for example, here we have six different kinds of tiles:



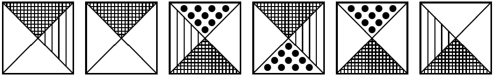
- a simple tiling problem, could be:

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



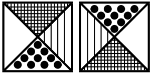
## Tiling Problems

- A tiling problem is a kind of jigsaw puzzle
- a tile $T$ is a $1 \times 1$ square, fixed in orientation, with a fixed color in each side
- for example, here we have six different kinds of tiles:



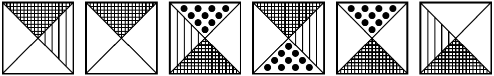
- a simple tiling problem, could be:

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



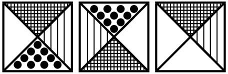
# Tiling Problems

- A tiling problem is a kind of jigsaw puzzle
- a tile $T$ is a $1 \times 1$ square, fixed in orientation, with a fixed color in each side
- for example, here we have six different kinds of tiles:



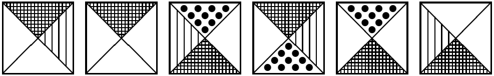
- a simple tiling problem, could be:

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



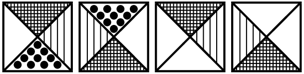
## Tiling Problems

- A tiling problem is a kind of jigsaw puzzle
- a tile $T$ is a $1 \times 1$ square, fixed in orientation, with a fixed color in each side
- for example, here we have six different kinds of tiles:



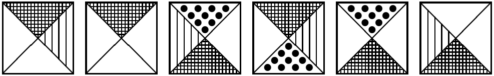
- a simple tiling problem, could be:

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



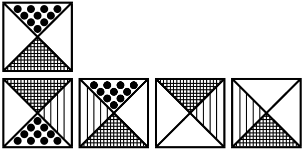
# Tiling Problems

- A tiling problem is a kind of jigsaw puzzle
- a tile $T$ is a $1 \times 1$ square, fixed in orientation, with a fixed color in each side
- for example, here we have six different kinds of tiles:



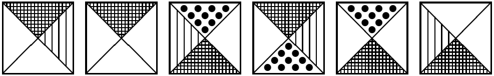
- a simple tiling problem, could be:

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



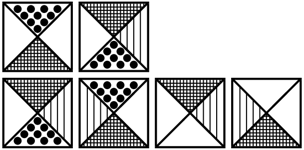
# Tiling Problems

- A tiling problem is a kind of jigsaw puzzle
- a tile $T$ is a $1 \times 1$ square, fixed in orientation, with a fixed color in each side
- for example, here we have six different kinds of tiles:



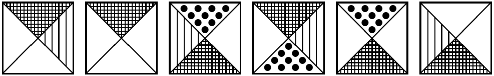
- a simple tiling problem, could be:

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



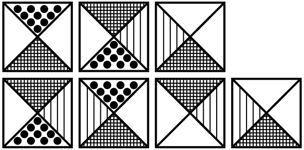
## Tiling Problems

- A tiling problem is a kind of jigsaw puzzle
- a tile $T$ is a $1 \times 1$ square, fixed in orientation, with a fixed color in each side
- for example, here we have six different kinds of tiles:



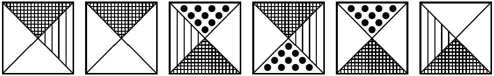
- a simple tiling problem, could be:

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



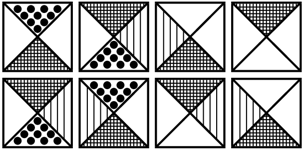
# Tiling Problems

- A tiling problem is a kind of jigsaw puzzle
- a tile $T$ is a $1 \times 1$ square, fixed in orientation, with a fixed color in each side
- for 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 $2 \times 4$, 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?*

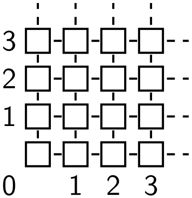
# Tiling Problems

- 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 is equivalent to the halting problem of Turing machines)



# Representing Tiling Problems

# Representing Tiling Problems

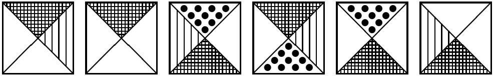
- Notice that every finite set of tiles $\mathcal{T} = \{t_1, \ldots, t_k\}$ can be represented as two binary relations $H$, and $V$.

## Representing Tiling Problems

- Notice that every finite set of tiles $\mathcal{T} = \{t_1, \ldots, t_k\}$ can be represented as two binary relations $H$, and $V$.
  We put $H(t_i, t_j)$ when the right side of $t_i$ coincide with the left side of $t_j$, and similarly for $V$.

# Representing Tiling Problems

- Notice that every finite set of tiles $\mathcal{T} = \{t_1, \ldots, t_k\}$ can be represented as two binary relations $H$, and $V$.
  We put $H(t_i, t_j)$ when the right side of $t_i$ coincide with the left side of $t_j$, and similarly for $V$.
- For example, for



$t_1$ $t_2$ $t_3$ $t_4$ $t_5$ $t_6$

we will have

$$\begin{array}{lcl} H & = & \{(t_1, t_3), (t_1, t_6), (t_2, t_4), (t_2, t_5), (t_2, t_1), \ldots\} \\ V & = & \{(t_1, t_3), (t_1, t_5), (t_1, t_6), (t_2, t_3), (t_2, t_5), \ldots\} \end{array}$$

# Coding a Tiling of the Grid in FO

# Coding a Tiling of the Grid in FO

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

## Coding a Tiling of the Grid in FO

- Theorem: The problem of deciding whether a given FO formula is satisfiable is undecidable.
  Let $\rightarrow$ and $\uparrow$ be relations, and $t_1, \ldots, t_n$ be propositinal symbols.
  Let $\varphi$ be the conjunction of the formulas:

# Coding a Tiling of the Grid in FO

- Theorem: The problem of deciding whether a given FO formula is satisfiable is undecidable.
  Let $\rightarrow$ and $\uparrow$ be relations, and $t_1, \ldots, t_n$ be propositinal symbols.
  Let $\varphi$ be the conjunction of the formulas:

  1) Total: $[\![x]\!]\langle\!\langle y\rangle\!\rangle(x{:}\langle R\rangle y)$ for $R \in \{\rightarrow, \uparrow\}$

## Coding a Tiling of the Grid in FO

- Theorem: The problem of deciding whether a given FO formula is satisfiable is undecidable.
  Let $\rightarrow$ and $\uparrow$ be relations, and $t_1, \ldots, t_n$ be propositinal symbols.
  Let $\varphi$ be the conjunction of the formulas:

  1) Total: $[\![x]\!]\langle\!\langle y\rangle\!\rangle(x{:}\langle R\rangle y)$ for $R \in \{\rightarrow, \uparrow\}$
  2) Functional: $[\![x]\!][\![y]\!][\![z]\!](x{:}\langle R\rangle y \wedge x{:}\langle R\rangle z \rightarrow y{:}z)$ for $R \in \{\rightarrow, \uparrow\}$

## Coding a Tiling of the Grid in FO

- Theorem: The problem of deciding whether a given FO formula is satisfiable is undecidable.
  Let $\rightarrow$ and $\uparrow$ be relations, and $t_1, \ldots, t_n$ be propositinal symbols.
  Let $\varphi$ be the conjunction of the formulas:

  1) Total: $[\![x]\!]\langle\langle y\rangle\rangle(x{:}\langle R\rangle y)$ for $R \in \{\rightarrow, \uparrow\}$
  2) Functional: $[\![x]\!][\![y]\!][\![z]\!](x{:}\langle R\rangle y \wedge x{:}\langle R\rangle z \rightarrow y{:}z)$ for $R \in \{\rightarrow, \uparrow\}$
  3) Commuting: $[\![x]\!][\![y]\!](x{:}\langle\rightarrow\rangle\langle\uparrow\rangle y \leftrightarrow x{:}\langle\uparrow\rangle\langle\rightarrow\rangle y)$

# Coding a Tiling of the Grid in FO

- Theorem: The problem of deciding whether a given FO formula is satisfiable is undecidable.
  Let $\rightarrow$ and $\uparrow$ be relations, and $t_1, \ldots, t_n$ be propositinal symbols.
  Let $\varphi$ be the conjunction of the formulas:

  1) Total: $[\![x]\!]\langle\!\langle y\rangle\!\rangle(x{:}\langle R\rangle y)$ for $R \in \{\rightarrow, \uparrow\}$
  2) Functional: $[\![x]\!][\![y]\!][\![z]\!](x{:}\langle R\rangle y \wedge x{:}\langle R\rangle z \rightarrow y{:}z)$ for $R \in \{\rightarrow, \uparrow\}$
  3) Commuting: $[\![x]\!][\![y]\!](x{:}\langle\rightarrow\rangle\langle\uparrow\rangle y \leftrightarrow x{:}\langle\uparrow\rangle\langle\rightarrow\rangle y)$
  4) Tiled: $[\![x]\!](x{:}t_1 \vee \ldots \vee x{:}t_n)$,

# Coding a Tiling of the Grid in FO

- Theorem: The problem of deciding whether a given FO formula is satisfiable is undecidable.
  Let $\rightarrow$ and $\uparrow$ be relations, and $t_1, \ldots, t_n$ be propositinal symbols.
  Let $\varphi$ be the conjunction of the formulas:

  1) Total: $[\![x]\!]\langle\!\langle y\rangle\!\rangle(x{:}\langle R\rangle y)$ for $R \in \{\rightarrow, \uparrow\}$
  2) Functional: $[\![x]\!][\![y]\!][\![z]\!](x{:}\langle R\rangle y \wedge x{:}\langle R\rangle z \rightarrow y{:}z)$ for $R \in \{\rightarrow, \uparrow\}$
  3) Commuting: $[\![x]\!][\![y]\!](x{:}\langle\rightarrow\rangle\langle\uparrow\rangle y \leftrightarrow x{:}\langle\uparrow\rangle\langle\rightarrow\rangle y)$
  4) Tiled: $[\![x]\!](x{:}t_1 \vee \ldots \vee x{:}t_n)$,
  5) But not Twice: $[\![x]\!](x{:}t_i \rightarrow x{:}\neg t_j)$ for $i \neq j$ ,

## Coding a Tiling of the Grid in FO

- Theorem: The problem of deciding whether a given FO formula is satisfiable is undecidable.
  Let $\rightarrow$ and $\uparrow$ be relations, and $t_1, \ldots, t_n$ be propositinal symbols.
  Let $\varphi$ be the conjunction of the formulas:

  1) Total: $[\![x]\!]\langle\!\langle y\rangle\!\rangle(x{:}\langle R\rangle y)$ for $R \in \{\rightarrow, \uparrow\}$
  2) Functional: $[\![x]\!][\![y]\!][\![z]\!](x{:}\langle R\rangle y \wedge x{:}\langle R\rangle z \rightarrow y{:}z)$ for $R \in \{\rightarrow, \uparrow\}$
  3) Commuting: $[\![x]\!][\![y]\!](x{:}\langle\rightarrow\rangle\langle\uparrow\rangle y \leftrightarrow x{:}\langle\uparrow\rangle\langle\rightarrow\rangle y)$
  4) Tiled: $[\![x]\!](x{:}t_1 \vee \ldots \vee x{:}t_n)$,
  5) But not Twice: $[\![x]\!](x{:}t_i \rightarrow x{:}\neg t_j)$ for $i \neq j$ ,
  6) Horizontal Match: $[\![x]\!][\![y]\!]((x{:}t_i \wedge x{:}\langle\rightarrow\rangle y) \rightarrow (\bigvee_{H(t_i,t_j)} y{:}t_j))$,

# Coding a Tiling of the Grid in FO

- Theorem: The problem of deciding whether a given FO formula is satisfiable is undecidable.
  Let $\rightarrow$ and $\uparrow$ be relations, and $t_1, \ldots, t_n$ be propositinal symbols.
  Let $\varphi$ be the conjunction of the formulas:

  1) Total: $[\![x]\!]\langle\!\langle y\rangle\!\rangle(x{:}\langle R\rangle y)$ for $R \in \{\rightarrow, \uparrow\}$
  2) Functional: $[\![x]\!][\![y]\!][\![z]\!](x{:}\langle R\rangle y \wedge x{:}\langle R\rangle z \rightarrow y{:}z)$ for $R \in \{\rightarrow, \uparrow\}$
  3) Commuting: $[\![x]\!][\![y]\!](x{:}\langle\rightarrow\rangle\langle\uparrow\rangle y \leftrightarrow x{:}\langle\uparrow\rangle\langle\rightarrow\rangle y)$
  4) Tiled: $[\![x]\!](x{:}t_1 \vee \ldots \vee x{:}t_n)$,
  5) But not Twice: $[\![x]\!](x{:}t_i \rightarrow x{:}\neg t_j)$ for $i \neq j$ ,
  6) Horizontal Match: $[\![x]\!][\![y]\!]((x{:}t_i \wedge x{:}\langle\rightarrow\rangle y) \rightarrow (\bigvee_{H(t_i,t_j)} y{:}t_j))$,
  7) Vertical Match: $[\![x]\!][\![y]\!]((x{:}t_i \wedge x{:}\langle\uparrow\rangle y) \rightarrow (\bigvee_{V(t_i,t_j)} y{:}t_j))$

# Coding a Tiling of the Grid in FO

- Theorem: The problem of deciding whether a given FO formula is satisfiable is undecidable.
  Let $\rightarrow$ and $\uparrow$ be relations, and $t_1, \ldots, t_n$ be propositinal symbols.
  Let $\varphi$ be the conjunction of the formulas:

  | | |
  |---|---|
  | 1) Total: | $[\![x]\!]\langle\!\langle y \rangle\!\rangle(x{:}\langle R\rangle y)$ for $R \in \{\rightarrow, \uparrow\}$ |
  | 2) Functional: | $[\![x]\!][\![y]\!][\![z]\!](x{:}\langle R\rangle y \wedge x{:}\langle R\rangle z \rightarrow y{:}z)$ for $R \in \{\rightarrow, \uparrow\}$ |
  | 3) Commuting: | $[\![x]\!][\![y]\!](x{:}\langle\rightarrow\rangle\langle\uparrow\rangle y \leftrightarrow x{:}\langle\uparrow\rangle\langle\rightarrow\rangle y)$ |
  | 4) Tiled: | $[\![x]\!](x{:}t_1 \vee \ldots \vee x{:}t_n)$, |
  | 5) But not Twice: | $[\![x]\!](x{:}t_i \rightarrow x{:}\neg t_j)$ for $i \neq j$ , |
  | 6) Horizontal Match: | $[\![x]\!][\![y]\!]((x{:}t_i \wedge x{:}\langle\rightarrow\rangle y) \rightarrow (\bigvee_{H(t_i,t_j)} y{:}t_j))$, |
  | 7) Vertical Match: | $[\![x]\!][\![y]\!]((x{:}t_i \wedge x{:}\langle\uparrow\rangle y) \rightarrow (\bigvee_{V(t_i,t_j)} y{:}t_j))$ |

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

# Relevant Bibliography I

### Undecidable Problems

- Alonzo Church invented lambda calculus and proposed it as a model for computation.

# Relevant Bibliography I

## Undecidable Problems

- Alonzo Church invented lambda calculus and proposed it as a model for computation.
- He showed then that the problem of satisfiability of first-order logic could not be coded in the lambda calculus. Hence it was uncomputable.

# Relevant Bibliography I

## Undecidable Problems

- Alonzo Church invented lambda calculus and proposed it as a model for computation.
- He showed then that the problem of satisfiability of first-order logic could not be coded in the lambda calculus. Hence it was uncomputable.
- Here is a list of some of Church's doctoral students: A. Anderson, P. Andrews, M. Davis, L. Henkin, S. Kleene, M. Rabin, B. Rosser, D. Scott, R. Smullyan, and A. Turing.

# Relevant Bibliography I

## Undecidable Problems

- Alonzo Church invented lambda calculus and proposed it as a model for computation.
- He showed then that the problem of satisfiability of first-order logic could not be coded in the lambda calculus. Hence it was uncomputable.
- Here is a list of some of Church's doctoral students: A. Anderson, P. Andrews, M. Davis, L. Henkin, S. Kleene, M. Rabin, B. Rosser, D. Scott, R. Smullyan, and A. Turing.

Church, Alonzo (1956). *Introduction to Mathematical Logic*. The Princeton University Press.

# Relevant Bibliography II

### Undecidable Problems

- Alan Turing invented Turing Machines and Computer Science.

# Relevant Bibliography II

### Undecidable Problems

- Alan Turing invented Turing Machines and Computer Science.
- He showed that the halting problem could not be decided by a Turing Machine.

# Relevant Bibliography II

## Undecidable Problems

- Alan Turing invented Turing Machines and Computer Science.
- He showed that the halting problem could not be decided by a Turing Machine.
- And that the behavior of a Turing Machine can easily be described in first-order logic, providing an alternative proof that the satisfiability problem of first-order logic is undecidable.

# Relevant Bibliography II

## Undecidable Problems

- Alan Turing invented Turing Machines and Computer Science.
- He showed that the halting problem could not be decided by a Turing Machine.
- And that the behavior of a Turing Machine can easily be described in first-order logic, providing an alternative proof that the satisfiability problem of first-order logic is undecidable.

Turing, Alan (1936), *On Computable Numbers, with an Application to the Entscheidungsproblem*. Proceedings of the London Mathematical Society, Series 2, Vol.42, pp 230–265, http://www.thocp.net/biographies/papers/turing_oncomputablenumbers_1936.pdf