On the Parallel Complexity of Bisimulation over Finite Systems

CSL 2016

Moses Ganardi

University of Siegen

joint work with Stefan Göller and Markus Lohrey

Hereditarily finite sets

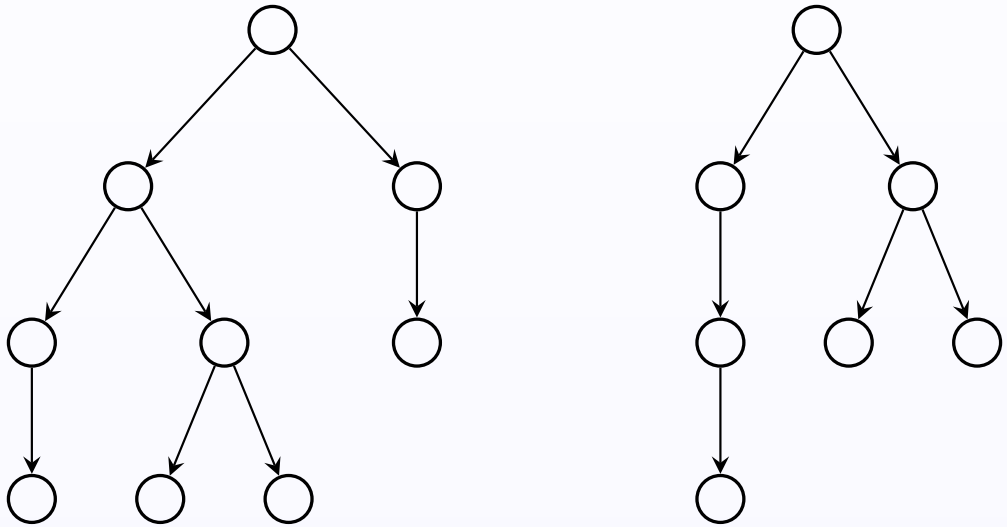
Are the following two sets equal?

$$\{\{\{\}\},\{\{\},\{\}\},\{\}\} \stackrel{?}{=} \{\{\{\}\}\},\{\{\},\{\}\}\}$$

Hereditarily finite sets

Are the following two sets equal?

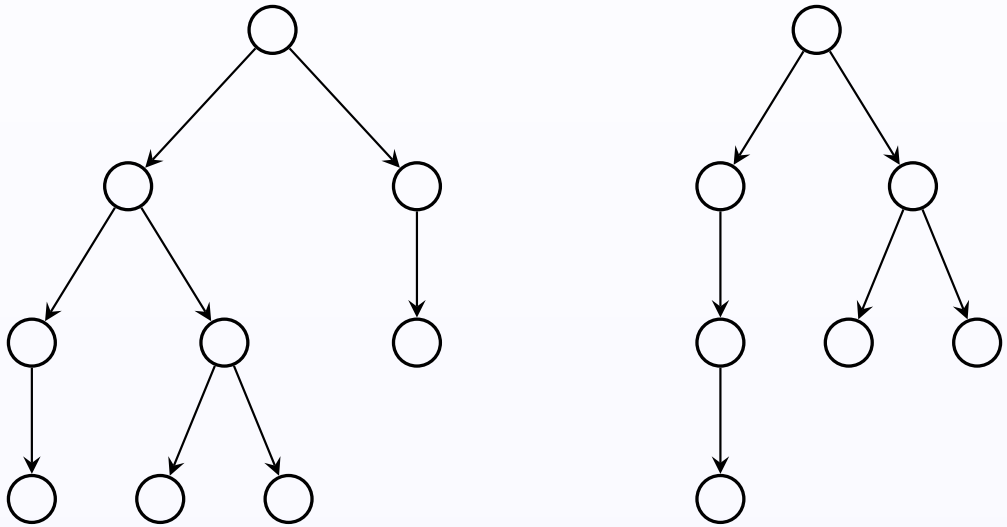
$$\{\{\{\}\},\{\{\},\{\}\},\{\}\} \stackrel{?}{=} \{\{\{\}\}\},\{\{\},\{\}\}\}$$



Hereditarily finite sets

Are the following two sets equal?

$$\{\{\{\}\},\{\{\},\{\}\},\{\}\} \stackrel{?}{=} \{\{\{\}\}\},\{\{\},\{\}\}\}$$



Two sets are equal iff the corresponding trees are bisimilar.

Starting point: Courcelle's theorem

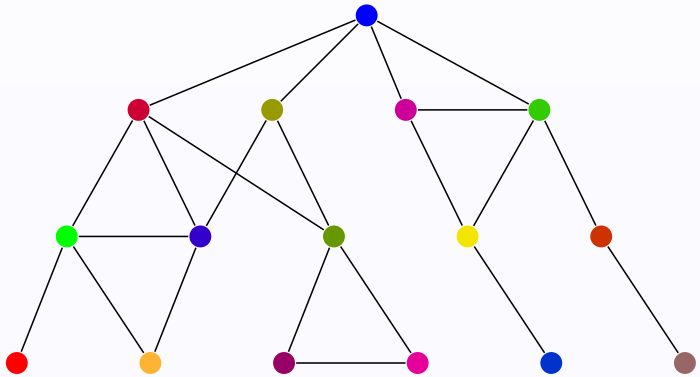
Theorem [Courcelle, 1990]

For every MSO-sentence ψ and every $k \in \mathbb{N}$, one can decide in linear time whether a given structure $\mathcal{A}$ of tree-width $\leq k$ satisfies ψ.

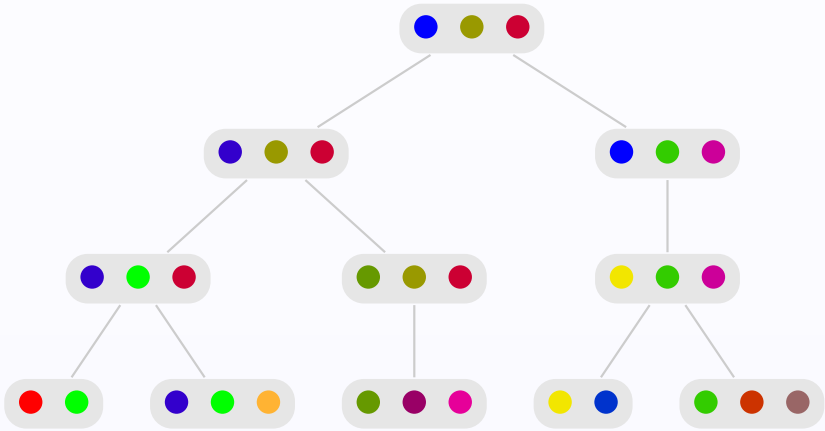
Starting point: Courcelle's theorem

Theorem [Courcelle, 1990]

For every MSO-sentence ψ and every $k \in \mathbb{N}$, one can decide in linear time whether a given structure $\mathcal{A}$ of tree-width $\leq k$ satisfies ψ.



graph G

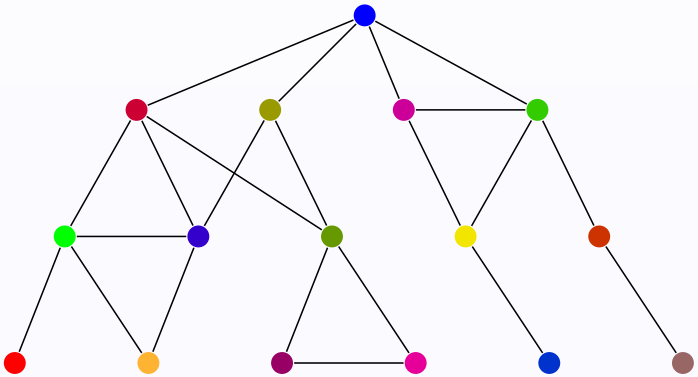


tree decomposition of width 2 for G

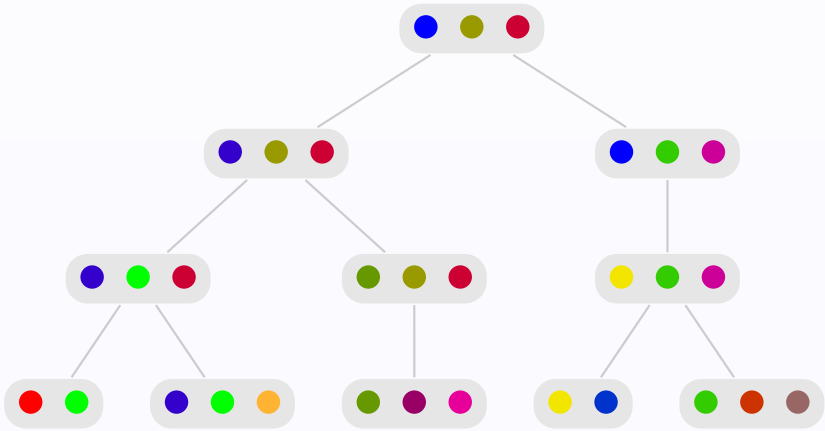
Starting point: Courcelle's theorem

Theorem [Courcelle, 1990]

For every MSO-sentence ψ and every $k \in \mathbb{N}$, one can decide in linear time whether a given structure $\mathcal{A}$ of tree-width $\leq k$ satisfies ψ.



graph G



tree decomposition of width 2 for G

- Every vertex and every edge is contained in some bag
- For all vertices v: bags containing v form a subtree

Below polynomial time

Theorem [Elberfeld et al., 2010]

On structures of bounded tree-width, MSO-properties can be decided in logarithmic space.

Below polynomial time

Theorem [Elberfeld et al., 2010]

On structures of bounded tree-width, MSO-properties can be decided in logarithmic space.

$\longrightarrow$ Determine the exact (parallel/space) complexity of problems on bounded tree-width structures.

$$\mathbf{AC}^0 \subseteq \mathbf{TC}^0 \subseteq \mathbf{NC}^1 \subseteq \mathbf{L} \subseteq \mathbf{NC} \subseteq \mathbf{P} \subseteq \mathbf{NP}$$

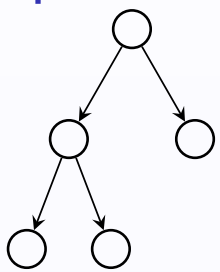
parallel alg. | sequential alg. | infeasible

Below logarithmic space

Theorem [Elberfeld et al., 2012]

On structures of bounded tree-width, MSO-properties can be decided in $\mathbf{NC}^1$, if tree decomposition is given as a term.

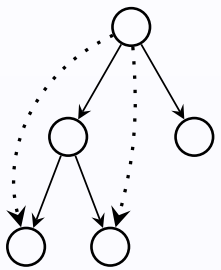
Tree representations



Pointer representation

(((())())())

Term representation $\equiv_{\mathbf{TC}^0}$



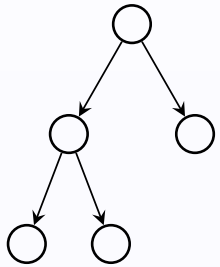
Ancestor representation

Below logarithmic space

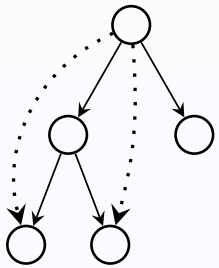
Theorem [Elberfeld et al., 2012]

On structures of bounded tree-width, MSO-properties can be decided in $\mathbf{NC}^1$, if tree decomposition is given as a term.

Tree representations



((( ) ( )) ( ))



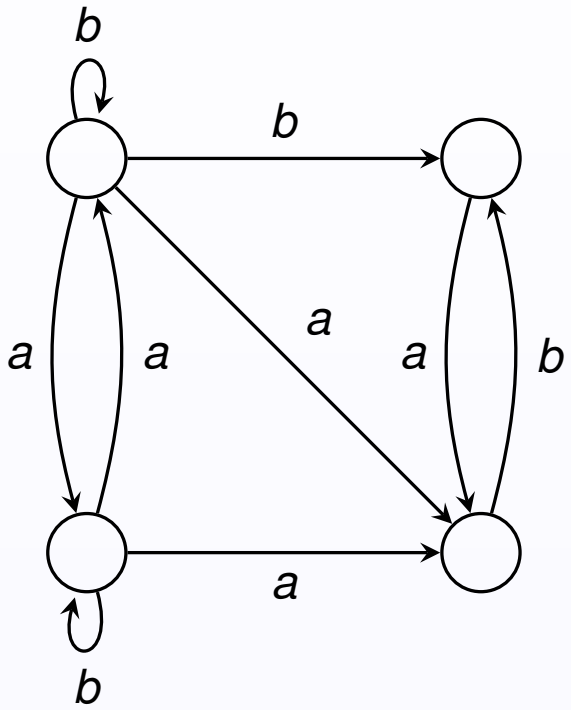
Pointer representation Term representation $\equiv_{\mathbf{TC}^0}$ Ancestor representation

Tree isomorphism problem

- In pointer representation: **L**-complete [Lindell, 1992]
- In term representation: $\mathbf{NC}^1$-complete [Buss, 1997]

Bisimulation game

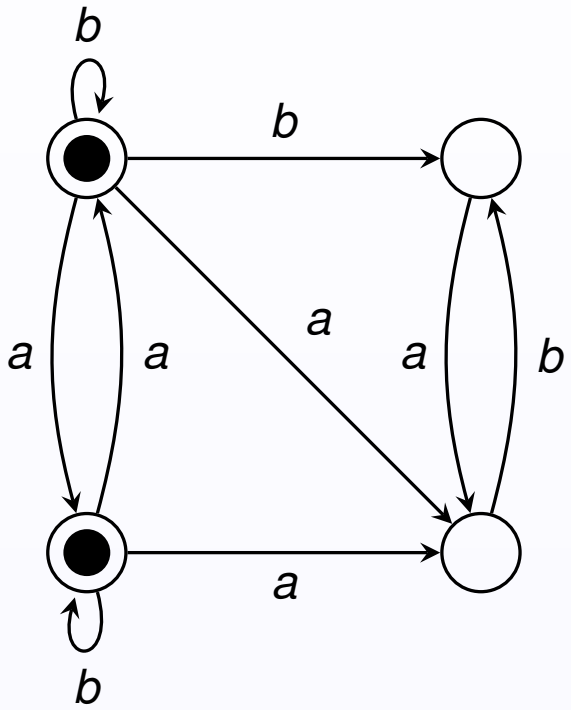
Attacker and Defender alternatingly move two pebbles along transitions.



- If a player cannot move, the other player wins.
- Infinite plays are won by Defender.
- u, v are bisimilar if Defender has a winning strategy from (u, v).

Bisimulation game

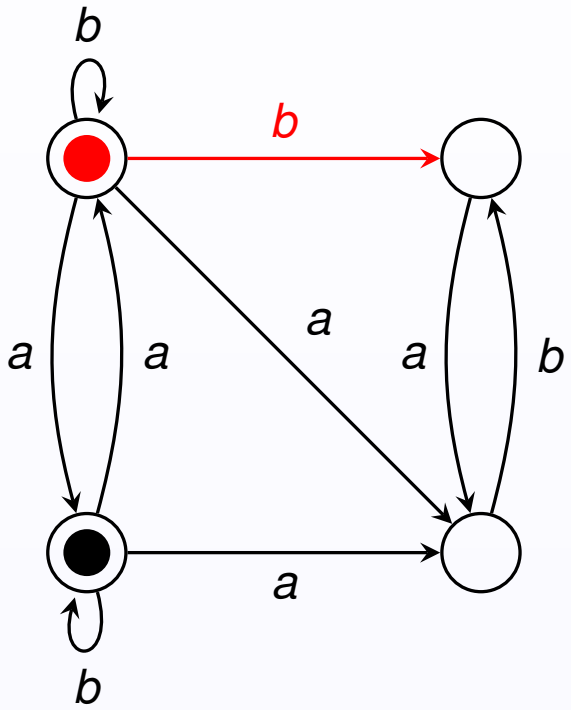
Attacker and Defender alternatingly move two pebbles along transitions.



- If a player cannot move, the other player wins.
- Infinite plays are won by Defender.
- u, v are bisimilar if Defender has a winning strategy from (u, v).

Bisimulation game

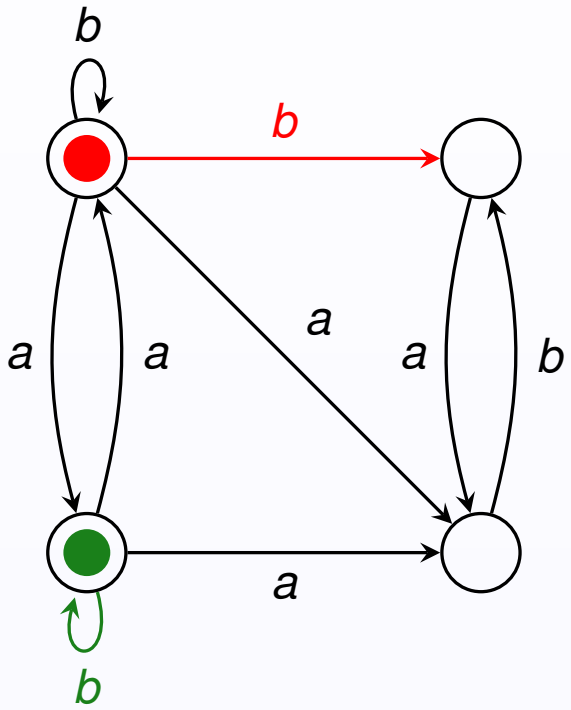
Attacker and Defender alternatingly move two pebbles along transitions.



- If a player cannot move, the other player wins.
- Infinite plays are won by Defender.
- u, v are bisimilar if Defender has a winning strategy from (u, v).

Bisimulation game

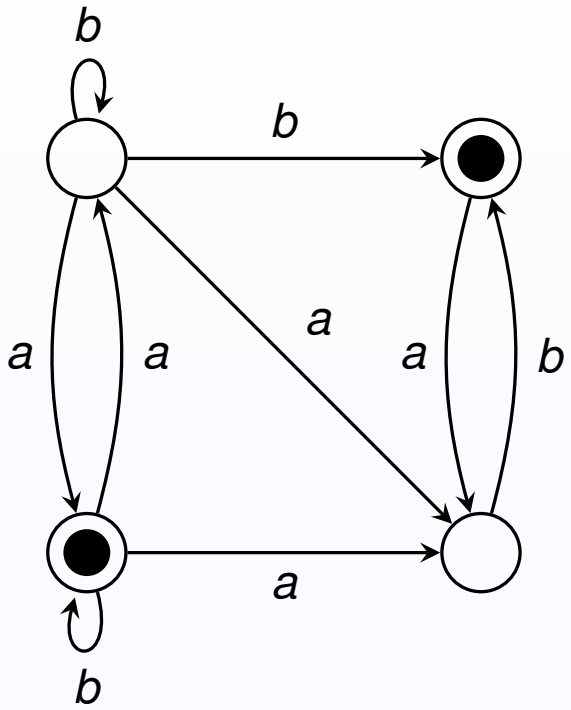
Attacker and Defender alternatingly move two pebbles along transitions.



- If a player cannot move, the other player wins.
- Infinite plays are won by Defender.
- u, v are bisimilar if Defender has a winning strategy from (u, v).

Bisimulation game

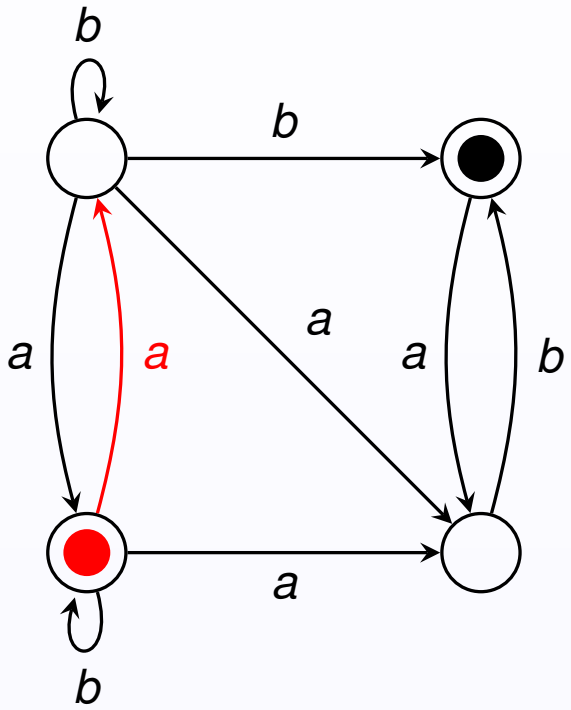
Attacker and Defender alternatingly move two pebbles along transitions.



- If a player cannot move, the other player wins.
- Infinite plays are won by Defender.
- u, v are bisimilar if Defender has a winning strategy from (u, v).

Bisimulation game

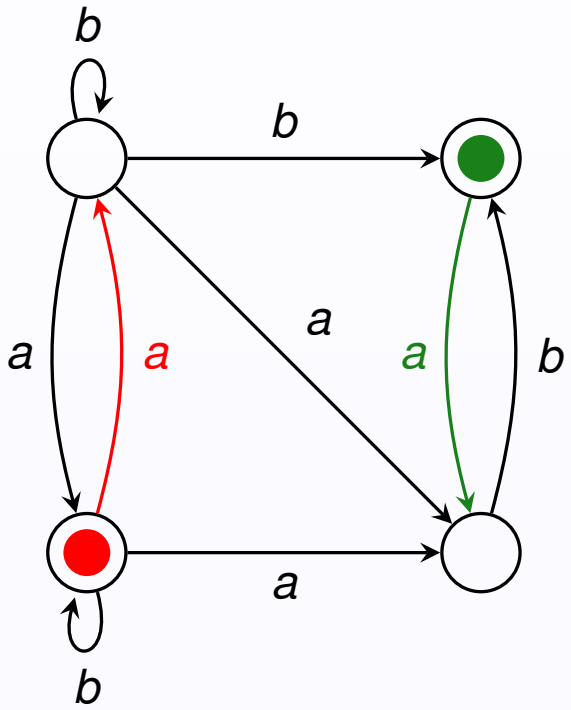
Attacker and Defender alternatingly move two pebbles along transitions.



- If a player cannot move, the other player wins.
- Infinite plays are won by Defender.
- u, v are bisimilar if Defender has a winning strategy from (u, v).

Bisimulation game

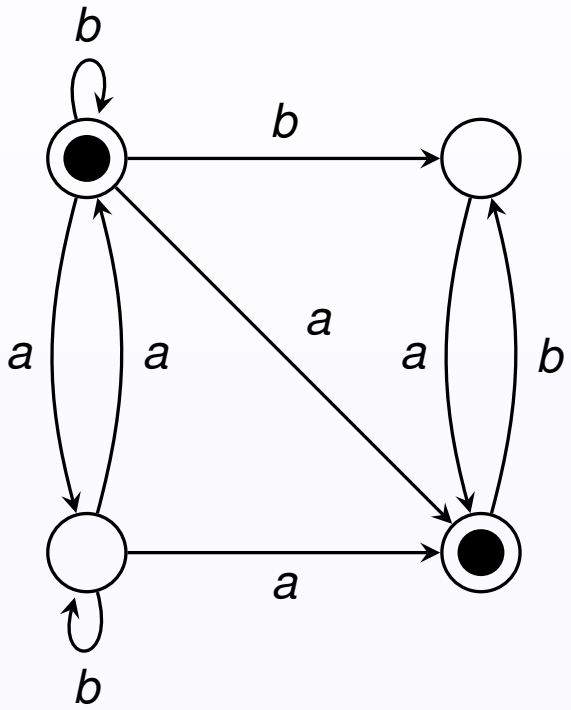
Attacker and Defender alternatingly move two pebbles along transitions.



- If a player cannot move, the other player wins.
- Infinite plays are won by Defender.
- u, v are bisimilar if Defender has a winning strategy from (u, v).

Bisimulation game

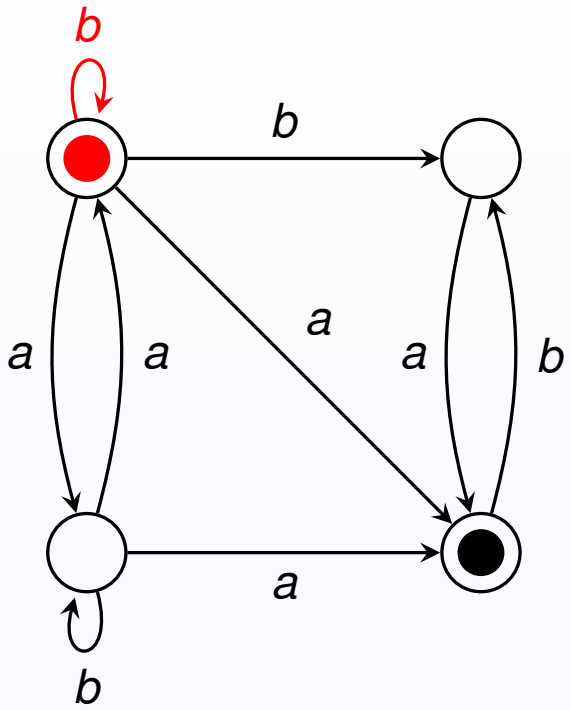
Attacker and Defender alternatingly move two pebbles along transitions.



- If a player cannot move, the other player wins.
- Infinite plays are won by Defender.
- u, v are bisimilar if Defender has a winning strategy from (u, v).

Bisimulation game

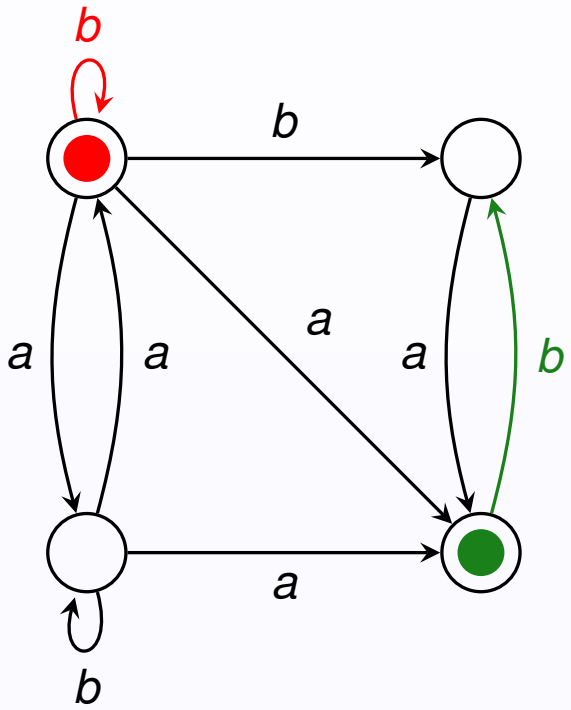
Attacker and Defender alternatingly move two pebbles along transitions.



- If a player cannot move, the other player wins.
- Infinite plays are won by Defender.
- u, v are bisimilar if Defender has a winning strategy from (u, v).

Bisimulation game

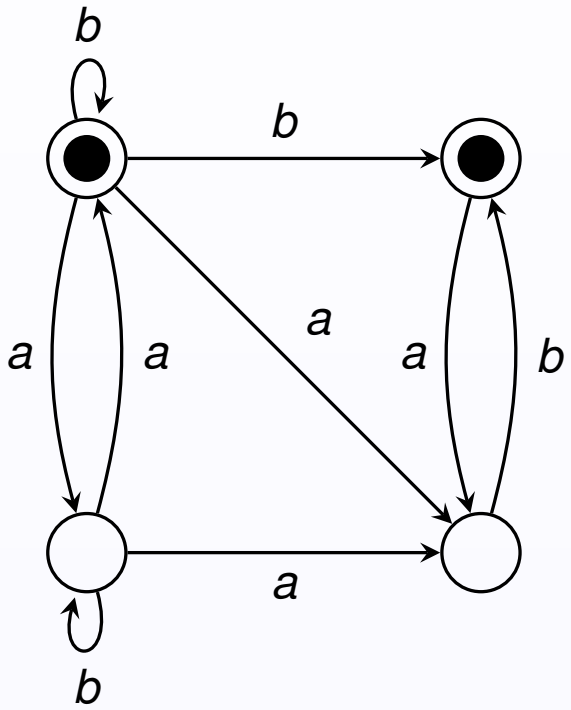
Attacker and Defender alternatingly move two pebbles along transitions.



- If a player cannot move, the other player wins.
- Infinite plays are won by Defender.
- u, v are bisimilar if Defender has a winning strategy from (u, v).

Bisimulation game

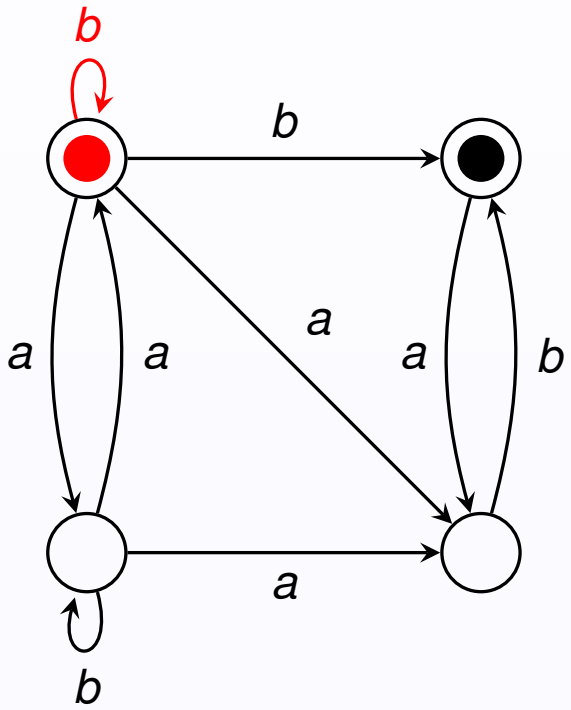
Attacker and Defender alternatingly move two pebbles along transitions.



- If a player cannot move, the other player wins.
- Infinite plays are won by Defender.
- u, v are bisimilar if Defender has a winning strategy from (u, v).

Bisimulation game

Attacker and Defender alternatingly move two pebbles along transitions.



- If a player cannot move, the other player wins.
- Infinite plays are won by Defender.
- u, v are bisimilar if Defender has a winning strategy from (u, v).

Parallel complexity of bisimulation

- over finite systems: deciding bisimilarity is in **P** (partition refinement algorithm)
- infinite state systems: many (un)decidability results

Theorem [Balcazar et al., 1992]

The bisimulation problem over finite systems is **P**-complete.

Parallel complexity of bisimulation

- over finite systems: deciding bisimilarity is in **P** (partition refinement algorithm)
- infinite state systems: many (un)decidability results

Theorem [Balcazar et al., 1992]

The bisimulation problem over finite systems is **P**-complete.

Are there efficient parallel (**NC**-)algorithms on restricted graph classes?

Parallel complexity of bisimulation

- over finite systems: deciding bisimilarity is in **P** (partition refinement algorithm)
- infinite state systems: many (un)decidability results

Theorem [Balcazar et al., 1992]

The bisimulation problem over finite systems is **P**-complete.

Are there efficient parallel (**NC**-)algorithms on restricted graph classes?

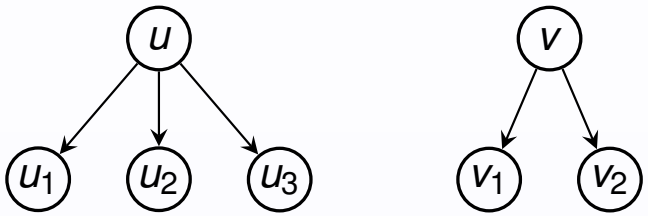
Theorem

The bisimulation problem for trees is

- in pointer representation: **L**-complete.
- in term representation: $\mathbf{NC}^1$-complete.

Bisimulation over unlabelled trees

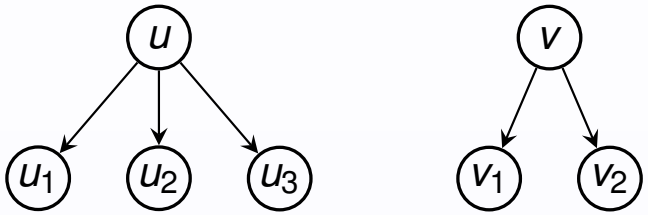
Given two trees T_1, T_2, construct the bisimulation circuit $C(T_1, T_2)$:
gate $x_{u,v} = 1 \iff u$ and v are bisimilar



$$x_{u,v} = \bigwedge_i \bigvee_j x_{u_i,v_j} \wedge \bigwedge_j \bigvee_i x_{u_i,v_j}$$

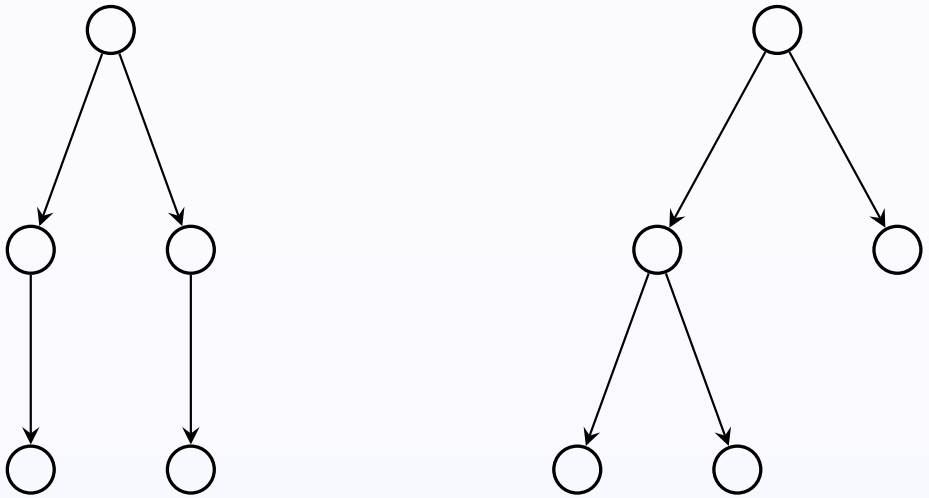
Bisimulation over unlabelled trees

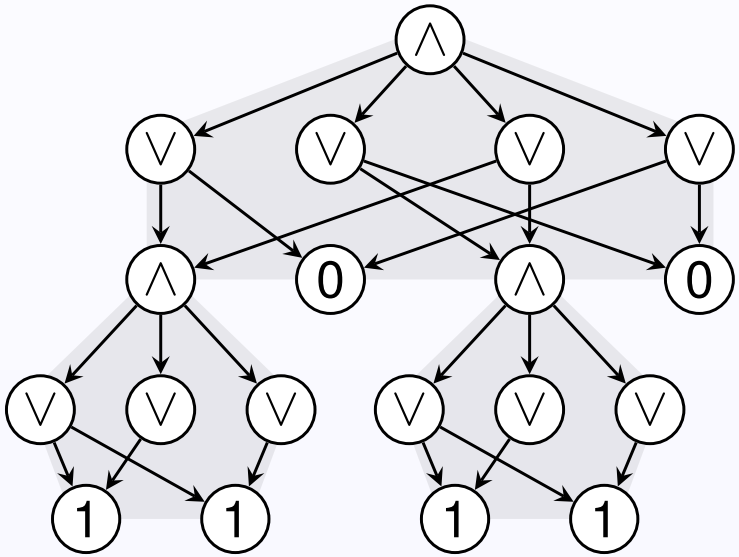
Given two trees T_1, T_2, construct the bisimulation circuit $C(T_1, T_2)$:
gate $x_{u,v} = 1 \iff u$ and v are bisimilar



$$x_{u,v} = \bigwedge_i \bigvee_j x_{u_i,v_j} \wedge \bigwedge_j \bigvee_i x_{u_i,v_j}$$

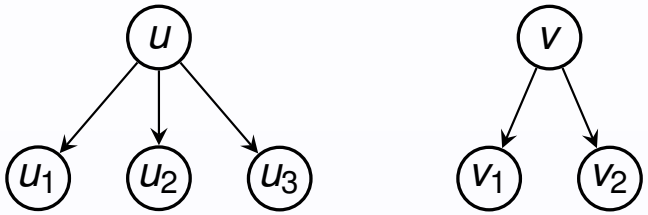
Example





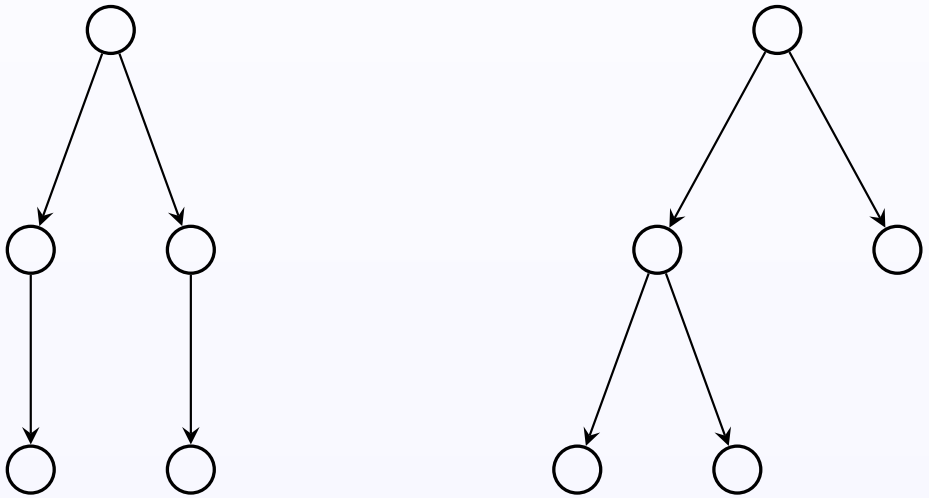
Bisimulation over unlabelled trees

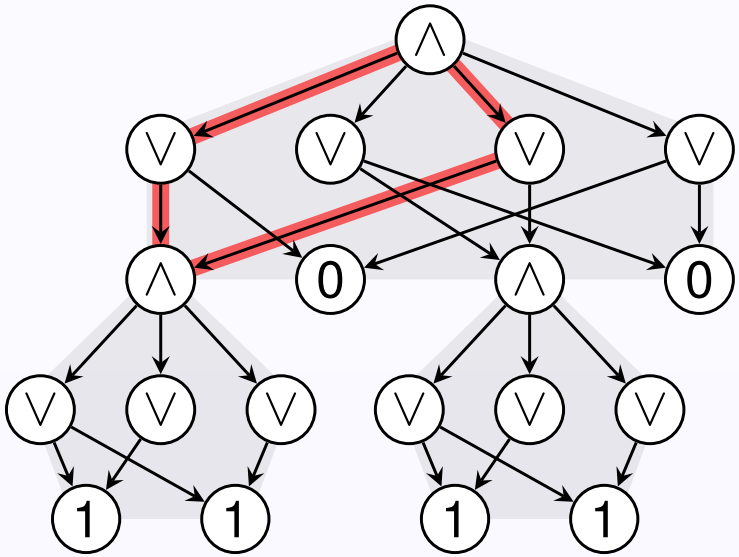
Given two trees T_1, T_2, construct the bisimulation circuit $C(T_1, T_2)$:
gate $x_{u,v} = 1 \iff u$ and v are bisimilar



$$x_{u,v} = \bigwedge_i \bigvee_j x_{u_i,v_j} \wedge \bigwedge_j \bigvee_i x_{u_i,v_j}$$

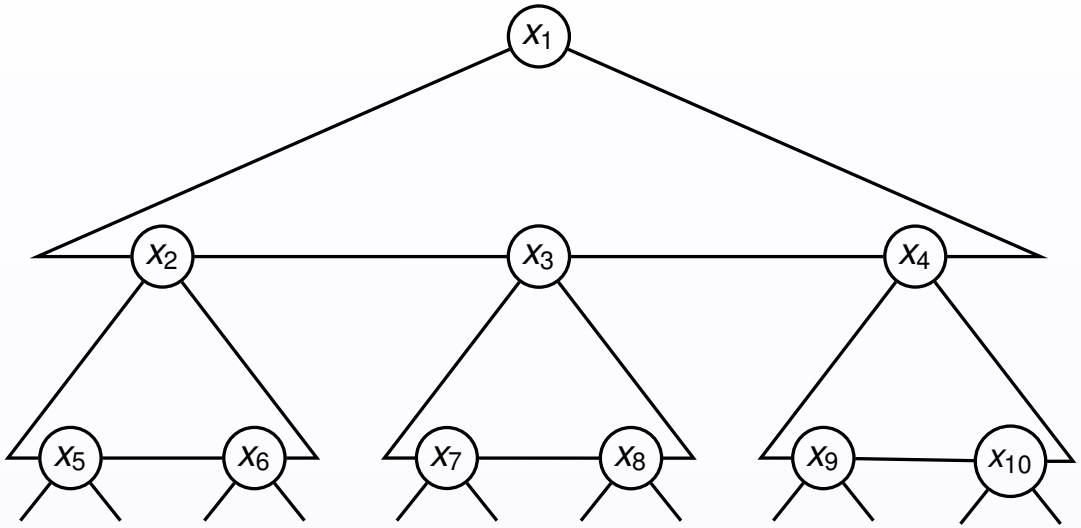
Example





bounded number of paths

Tree-shaped circuits

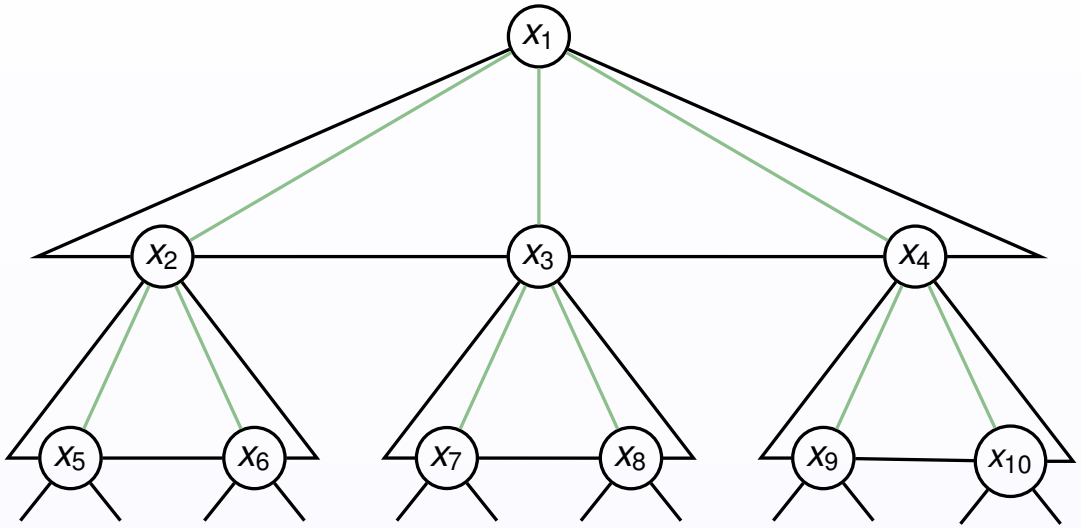


Syntactic definition

$\mathcal{S} = (x_i := \varphi_i)_{1 \leq i \leq n}$

- Boolean formulas $\varphi_1, \ldots, \varphi_n$ over $x_1, \ldots, x_n$

Tree-shaped circuits

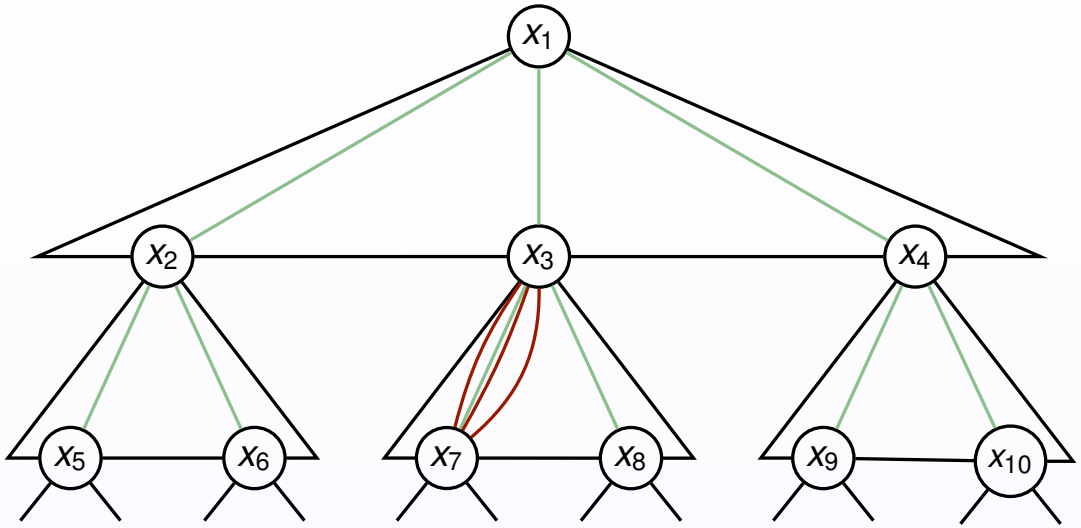


Syntactic definition

$\mathcal{S} = (x_i := \varphi_i)_{1 \leq i \leq n}$

- Boolean formulas $\varphi_1, \ldots, \varphi_n$ over $x_1, \ldots, x_n$
- $T_{\mathcal{S}} = (\{x_1, \ldots, x_n\}, \{(x_i, x_j) : x_j \text{ occurs in } \varphi_i\})$ is a tree

Tree-shaped circuits



Syntactic definition

$\mathcal{S} = (x_i := \varphi_i)_{1 \leq i \leq n}$

- Boolean formulas $\varphi_1, \ldots, \varphi_n$ over $x_1, \ldots, x_n$
- $T_{\mathcal{S}} = (\{x_1, \ldots, x_n\}, \{(x_i, x_j) : x_j \text{ occurs in } \varphi_i\})$ is a tree
- width of $\mathcal{S}$: max. number of occurrences of a variable x_j in some φ_i

Tree-shaped circuits

Theorem

For every $m \in \mathbb{N}$, evaluating tree-shaped circuits $\mathcal{S}$ of width $\leq m$ is in

- logarithmic space.
- $\mathbf{NC}^1$ if $T_{\mathcal{S}}$ is given as a term.

Tree-shaped circuits

Theorem

For every $m \in \mathbb{N}$, evaluating tree-shaped circuits $\mathcal{S}$ of width $\leq m$ is in

- logarithmic space.
- $\mathbf{NC}^1$ if $T_{\mathcal{S}}$ is given as a term.

Two proofs

1. Recursive logspace algorithm
2. Courcelle's theorem (for the $\mathbf{NC}^1$-result)

Tree-shaped circuits

Theorem

For every $m \in \mathbb{N}$, evaluating tree-shaped circuits $\mathcal{S}$ of width $\leq m$ is in

- logarithmic space.
- $\mathbf{NC}^1$ if $T_{\mathcal{S}}$ is given as a term.

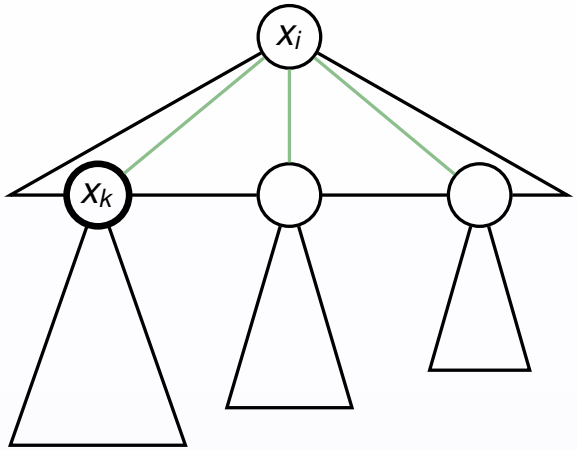
Two proofs

1. Recursive logspace algorithm
2. Courcelle's theorem (for the $\mathbf{NC}^1$-result)

- Size of a tree node u: size of the subtree rooted in u
- u is heavy if $|u| \geq |v|$ for all siblings v of u, otherwise u is light.
- Each path in a tree T has $O(\log |T|)$ light nodes.

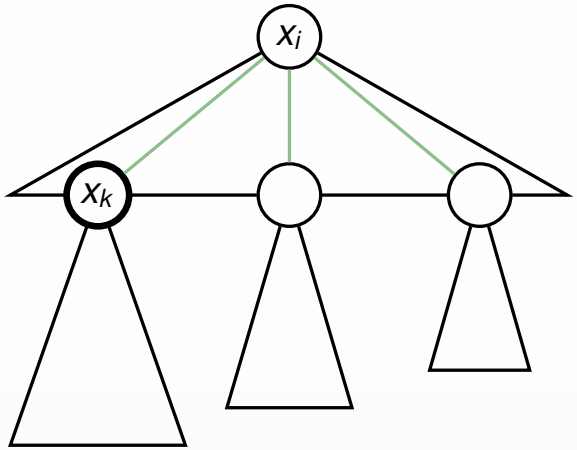
Preparation

Compute in $T_{\mathcal{S}}$ for each variable x_i its size and a heavy child x_k



Preparation

Compute in $T_{\mathcal{S}}$ for each variable x_i its size and a heavy child x_k

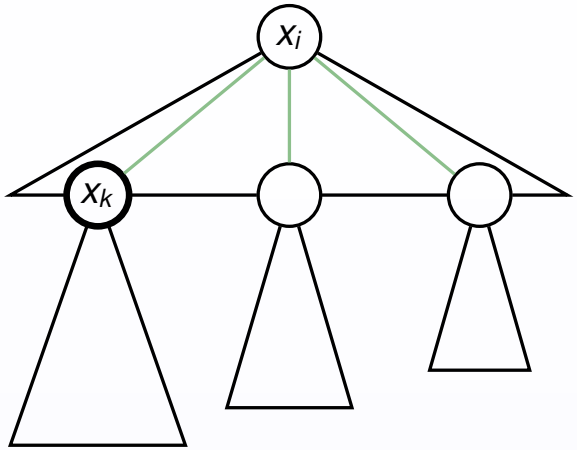


How to calculate x_i

1. Evaluate heavy child x_k recursively and push its value on a stack

Preparation

Compute in $T_{\mathcal{S}}$ for each variable x_i its size and a heavy child x_k

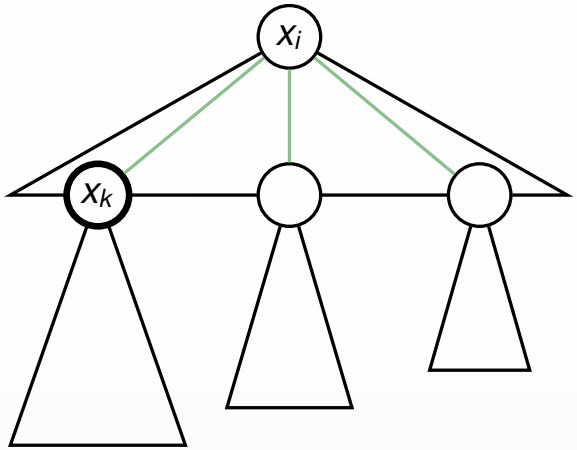


How to calculate x_i

1. Evaluate heavy child x_k recursively and push its value on a stack
2. Evaluate φ_i by a depth-first traversal

Preparation

Compute in $T_{\mathcal{S}}$ for each variable x_i its size and a heavy child x_k

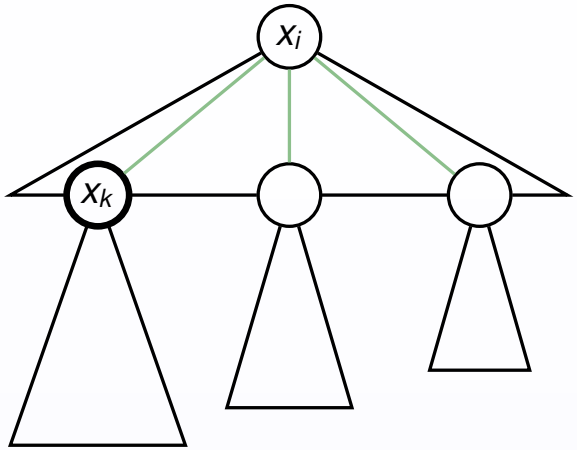


How to calculate x_i

1. Evaluate heavy child x_k recursively and push its value on a stack
2. Evaluate φ_i by a depth-first traversal
   - value of x_k is known

Preparation

Compute in T_S for each variable x_i its size and a heavy child x_k



How to calculate x_i

1. Evaluate heavy child x_k recursively and push its value on a stack
2. Evaluate φ_i by a depth-first traversal
   - value of x_k is known
   - each occurrence of a light variable $x_j \neq x_k$ is evaluated recursively (store position $d \in \{1, \ldots, m\}$ on the stack)

Alternative proof

Theorem

Circuits of bounded tree-width can be evaluated in logspace.
(in $\mathbf{NC}^1$ if tree decomposition is given as a term)

Alternative proof

Theorem

Circuits of bounded tree-width can be evaluated in logspace.
(in $\mathbf{NC}^1$ if tree decomposition is given as a term)

But: The bisimulation circuits have unbounded tree- and clique-width.

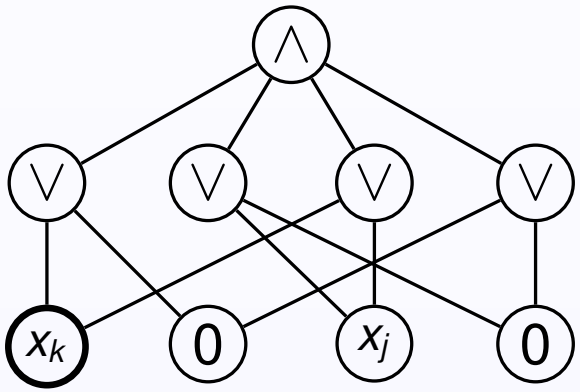
Alternative proof

Theorem

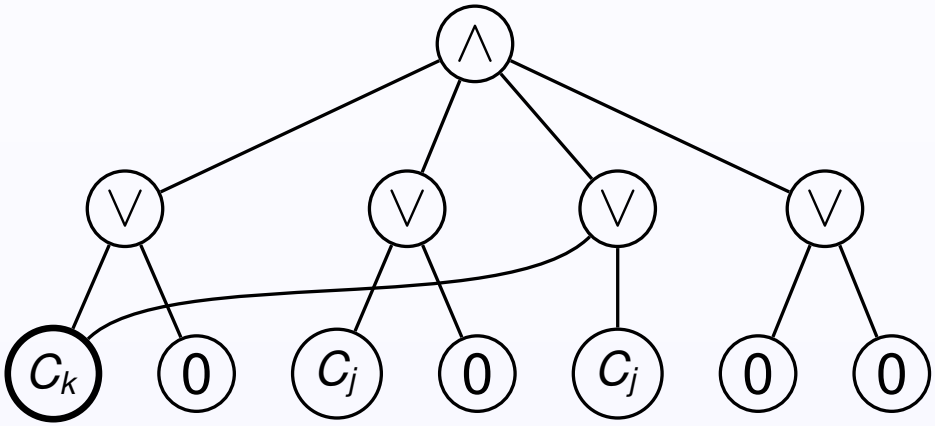
Circuits of bounded tree-width can be evaluated in logspace.
(in $\mathbf{NC}^1$ if tree decomposition is given as a term)

But: The bisimulation circuits have unbounded tree- and clique-width.

Partial unfolding



Subcircuit corresponding to φ_i



Partial unfolding C_i

Size of the partial unfolding

- $m = \text{width}(\mathcal{S})$.
- $\ell_i =$ number of light nodes from x_1 to x_i in $T_{\mathcal{S}}$.
- A gate x_i is copied at most m^{ℓ_i} times.

Size of the partial unfolding

- $m = \text{width}(\mathcal{S})$.
- $\ell_i =$ number of light nodes from x_1 to x_i in $T_{\mathcal{S}}$.
- A gate x_i is copied at most m^{ℓ_i} times.

Furthermore, it has tree-width ≤ 2 and is computable in

- logspace,
- $\mathbf{TC}^0$ if $T_{\mathcal{S}}$ given as term.

Size of the partial unfolding

- $m = \text{width}(\mathcal{S})$.
- $\ell_i =$ number of light nodes from x_1 to x_i in $T_{\mathcal{S}}$.
- A gate x_i is copied at most m^{ℓ_i} times.

Furthermore, it has tree-width ≤ 2 and is computable in

- logspace,
- $\mathbf{TC}^0$ if $T_{\mathcal{S}}$ given as term.

Theorem

The set equality problem is $\mathbf{NC}^1$-complete.

From Trees to Bounded Tree-Width

Challenges in the Bisimulation Problem

- Synchronized product creates grid-like structures
- Symmetry makes it hard to prove lower bounds
- “Defender’s forcing” does not work

From Trees to Bounded Tree-Width

Challenges in the Bisimulation Problem

- Synchronized product creates grid-like structures
- Symmetry makes it hard to prove lower bounds
- “Defender’s forcing” does not work

Simulation problem

- Attacker (Defender) always moves the “left” (“right”) pebble
- u is simulated by v if Defender has a winning strategy

Simulation Problem

Theorem

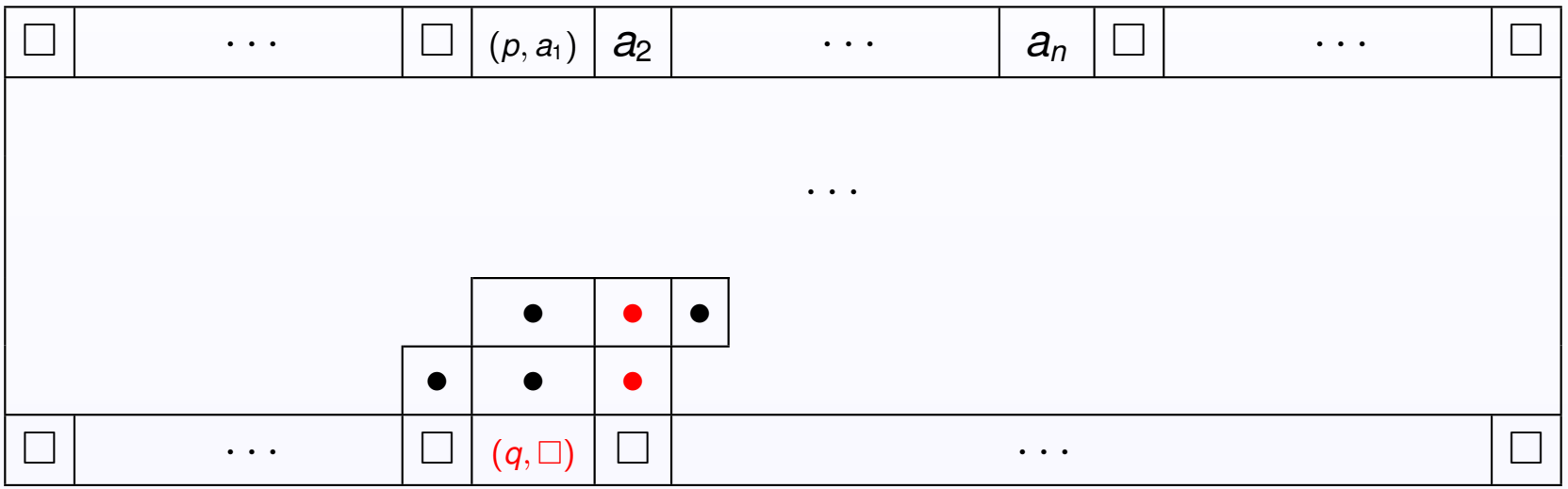
The simulation problem on graphs of bounded path-width is **P**-hard.

Simulation Problem

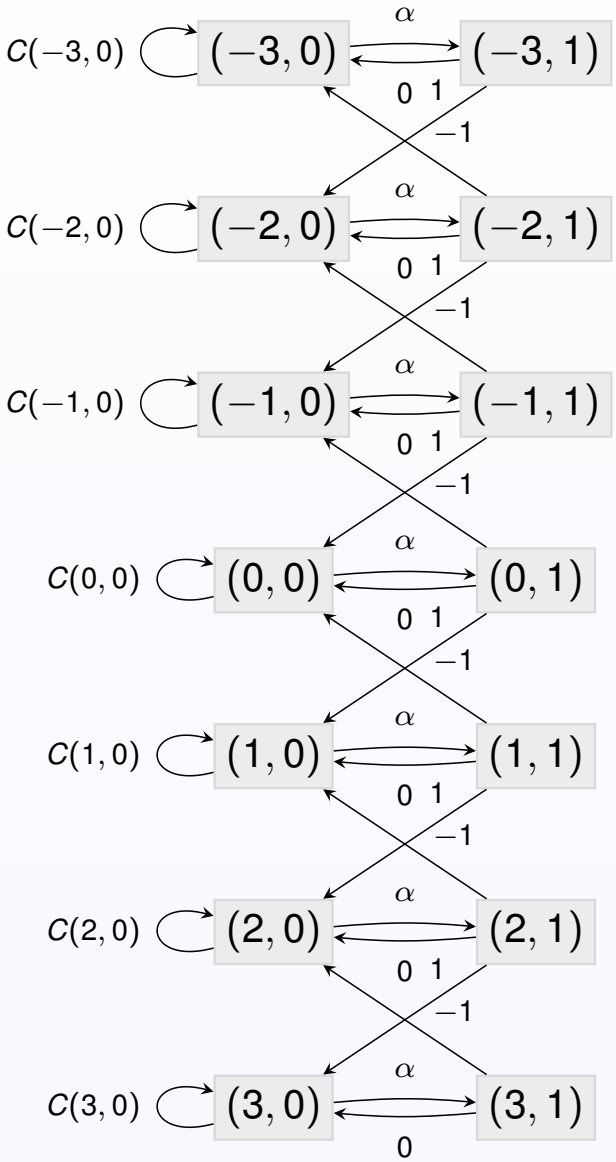
Theorem

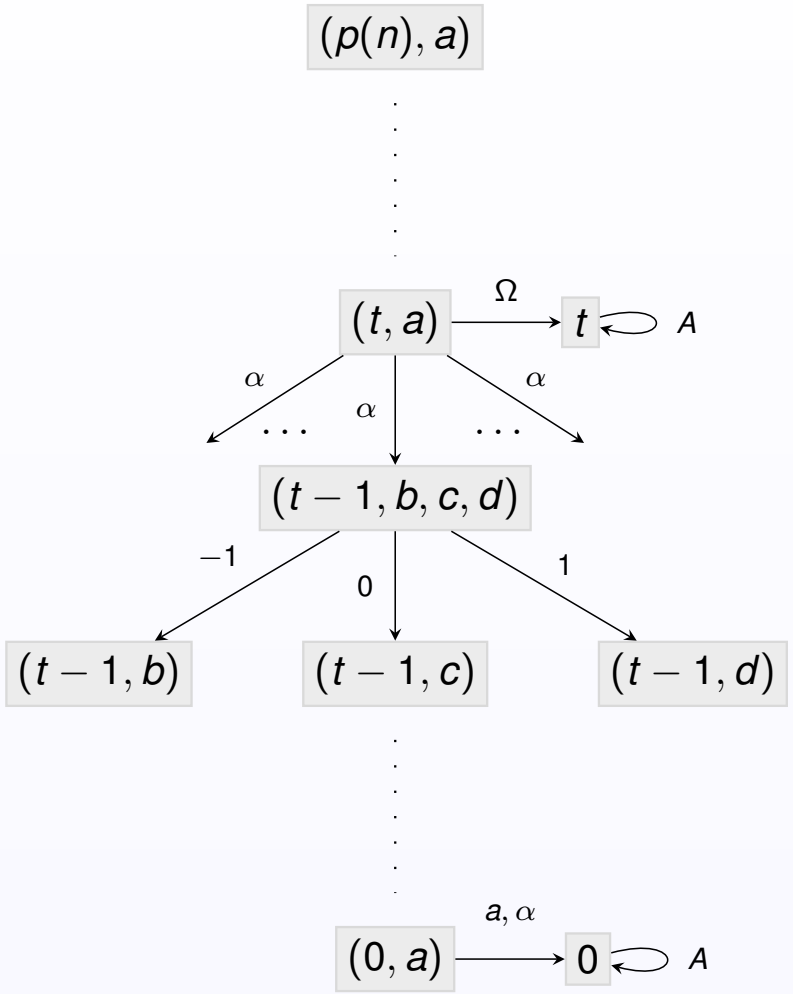
The simulation problem on graphs of bounded path-width is **P**-hard.

Fix a polynomial-time TM with a **P**-complete membership problem.



Proof sketch





Summary

The tree bisimulation problem is

- **L**-complete (for pointer structures)
- **NC**^{1}-complete (for term expressions)

Is there an **NC**-algorithm for bisimulation on graphs of bounded tree-width?