Quantitative Logics on Structure Rewriting Systems

Diana Fischer, Łukasz Kaiser, Simon Leßenich, Łukasz Stafiniak

Mathematische Grundlagen der Informatik, RWTH Aachen
Instytut Informatyki, Uniwersytet Wrocławski
LIAFA, CNRS & Université Paris Diderot – Paris 7

EPI Contraintes Seminar, INRIA Paris-Rocquencourt, 2012

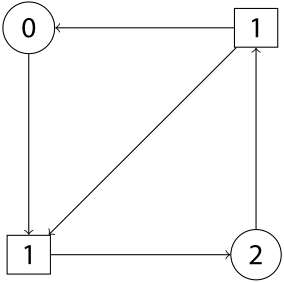
Overview

Overview

Motivation

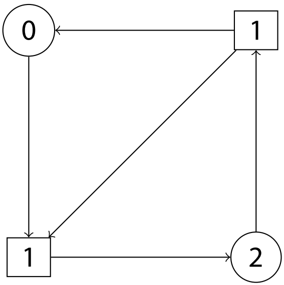
Motivation

Look, I can do synthesis for that:

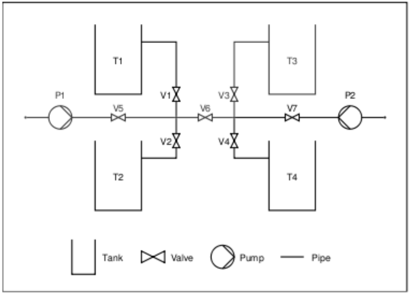


Motivation

Look, I can do synthesis for that:

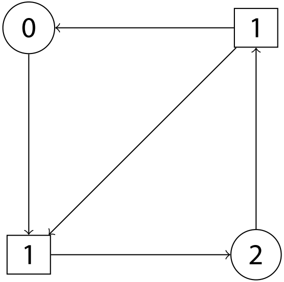


Great, I have something very similar:

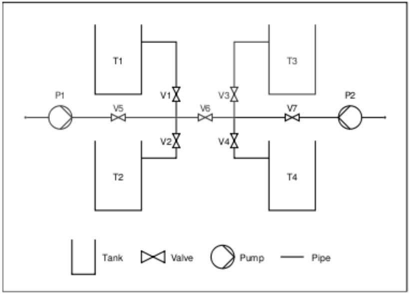


Motivation

Look, I can do synthesis for that:



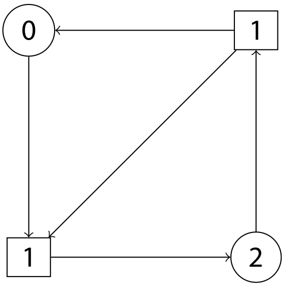
Great, I have something very similar:



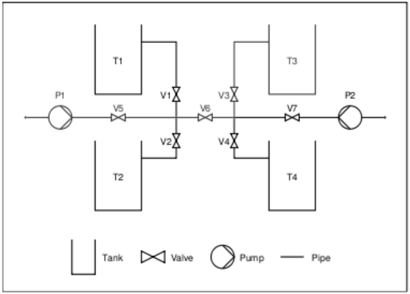
But the whole right side is just **one state** on the left!

Motivation

Look, I can do synthesis for that:



Great, I have something very similar:



But the whole right side is just **one state** on the left!

Can we model states by arbitrary relational structures?

Structure Transition Systems

Kripke Structures

$$\mathcal{K} = (V, E_a, E_b, \dots)$$

Finite Relational Structures

$$\mathfrak{A} = (A, R_1, R_2, \dots)$$

Metafinite Structures

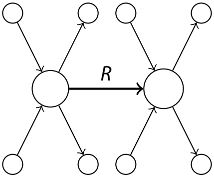
$$\mathfrak{A} \textbf{ and } f_1, f_2, \dots : A \to \mathbb{R}$$

Structure Transition Systems

$$\mathcal{K} \textbf{ and } s : V \to (\mathfrak{A}, \bar{f})$$

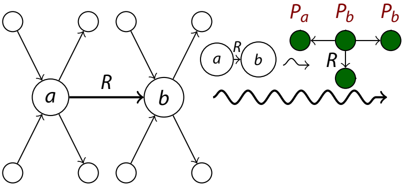
Discrete Structure Rewriting Rules

Rewriting Example



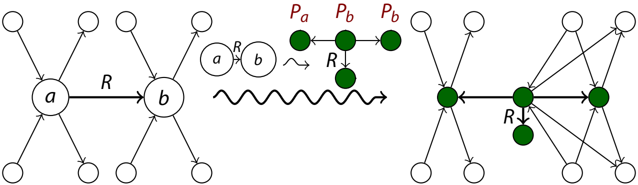
Discrete Structure Rewriting Rules

Rewriting Example



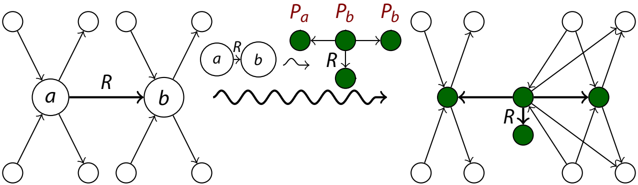
Discrete Structure Rewriting Rules

Rewriting Example



Discrete Structure Rewriting Rules

Rewriting Example



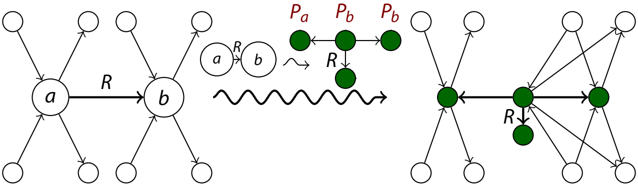
Relational Structures and Embeddings

$$\sigma \;:\quad \mathfrak{A} = (A, R_1^{\mathfrak{A}}, R_2^{\mathfrak{A}}, \dots, R_k^{\mathfrak{A}}) \quad\to\quad (B, R_1^{\mathfrak{B}}, R_2^{\mathfrak{B}}, \dots, R_k^{\mathfrak{B}}) = \mathfrak{B}$$

Embedding: σ is **injective** and $R_i^{\mathfrak{A}}(a_1, \dots, a_{r_i}) \Leftrightarrow R_i^{\mathfrak{B}}(\sigma(a_1), \dots, \sigma(a_{r_i}))$

Discrete Structure Rewriting Rules

Rewriting Example



Relational Structures and Embeddings

$$\sigma \;:\quad \mathfrak{A} = (A, R_1^{\mathfrak{A}}, R_2^{\mathfrak{A}}, \ldots, R_k^{\mathfrak{A}}) \quad\to\quad (B, R_1^{\mathfrak{B}}, R_2^{\mathfrak{B}}, \ldots, R_k^{\mathfrak{B}}) = \mathfrak{B}$$

Embedding: σ is **injective** and $R_i^{\mathfrak{A}}(a_1, \ldots, a_{r_i}) \Leftrightarrow R_i^{\mathfrak{B}}(\sigma(a_1), \ldots, \sigma(a_{r_i}))$

Rewriting Definition

$\mathfrak{B} = \mathfrak{A}[\mathfrak{L} \to \mathfrak{R}/\sigma]$ iff $B = (A \setminus \sigma(L)) \dot\cup R$ and,

for $M = \{(r, a) \mid a = \sigma(l), r \in P_l^{\mathfrak{R}}$ for some $l \in L\} \cup \{(a, a) \mid a \in A\}$,

$$(b_1, \ldots, b_{r_i}) \in R_i^{\mathfrak{B}} \;\Leftrightarrow\; (b_1, \ldots, b_{r_i}) \in R_i^{\mathfrak{R}} \text{ or } (b_1 M \times \ldots \times b_{r_i} M) \cap R_i^{\mathfrak{A}} \neq \emptyset.$$

(in the second case at least one $b_j \notin \mathfrak{A}$)

Structure Rewriting Games

Game arena **(of a two-player zero-sum game)** is a **directed graph** with:

- vertices partitioned into positions of **Player 0** and **Player 1**
- edges **labelled by rewriting rules**

Structure Rewriting Games

Game arena **(of a two-player zero-sum game)** is a **directed graph** with:

- vertices partitioned into positions of **Player 0** and **Player 1**
- edges **labelled by rewriting rules**

Two interpretations of $\mathfrak{L} \to \mathfrak{R}$:

- **Existential**: $\mathfrak{A}_{\mathrm{next}} = \mathfrak{A}[\mathfrak{L} \to \mathfrak{R}/\sigma]$, the player chooses the embedding σ
- **Universal**: $\mathfrak{A}_{\mathrm{next}} = \mathfrak{A}[\mathfrak{L} \to \mathfrak{R}]$, **all** occurrences of $\mathfrak{L}$ are rewritten to $\mathfrak{R}$

Structure Rewriting Games

Game arena (of a two-player zero-sum game) is a **directed graph** with:

- vertices partitioned into positions of **Player 0** and **Player 1**
- edges **labelled by rewriting rules**

Two interpretations of $\mathfrak{L} \to \mathfrak{R}$:

- **Existential**: $\mathfrak{A}_{\text{next}} = \mathfrak{A}[\mathfrak{L} \to \mathfrak{R}/\sigma]$, the player chooses the embedding σ
- **Universal**: $\mathfrak{A}_{\text{next}} = \mathfrak{A}[\mathfrak{L} \to \mathfrak{R}]$, **all** occurrences of $\mathfrak{L}$ are rewritten to $\mathfrak{R}$

Winning conditions:

- Lμ (or temporal) formula ψ with **MSO sentences** for predicates, or
- MSO formula φ to be evaluated on the **limit** of the play
  Limit of $\mathfrak{A}_0\mathfrak{A}_1\mathfrak{A}_2 \ldots \; = \; \left(\bigcup_{n\in\mathbb{N}} \bigcap_{i\geq n} A_i, \; \bigcup_{n\in\mathbb{N}} \bigcap_{i\geq n} R^{\mathfrak{A}_i}\right)$
- **Reach** φ: Player 0 **wins** if the play reaches $\mathfrak{A}$ s.t. $\mathfrak{A} \models \varphi$

Structure Rewriting Games

Game arena **(of a two-player zero-sum game)** is a **directed graph** with:

- vertices partitioned into positions of **Player 0** and **Player 1**
- edges **labelled by rewriting rules**

Two interpretations of $\mathfrak{L} \to \mathfrak{R}$:

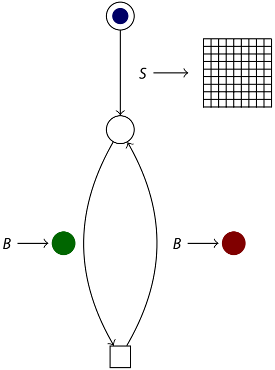
- **Existential**: $\mathfrak{A}_{\text{next}} = \mathfrak{A}[\mathfrak{L} \to \mathfrak{R}/\sigma]$, the player chooses the embedding σ
- **Universal**: $\mathfrak{A}_{\text{next}} = \mathfrak{A}[\mathfrak{L} \to \mathfrak{R}]$, **all** occurrences of $\mathfrak{L}$ are rewritten to $\mathfrak{R}$

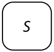
Winning conditions:

- Lμ (or temporal) formula ψ with **MSO sentences** for predicates, or
- MSO formula φ to be evaluated on the **limit** of the play
  Limit of $\mathfrak{A}_0\mathfrak{A}_1\mathfrak{A}_2 \ldots = \left(\bigcup_{n\in\mathbb{N}} \bigcap_{i\geq n} A_i, \bigcup_{n\in\mathbb{N}} \bigcap_{i\geq n} R^{\mathfrak{A}_i}\right)$
- **Reach** φ: Player 0 **wins** if the play reaches $\mathfrak{A}$ s.t. $\mathfrak{A} \models \varphi$

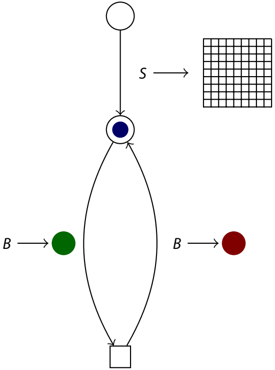
Motivation: many questions are **naturally defined as such games**:
constraint satisfaction, model checking, graph measures, fun games

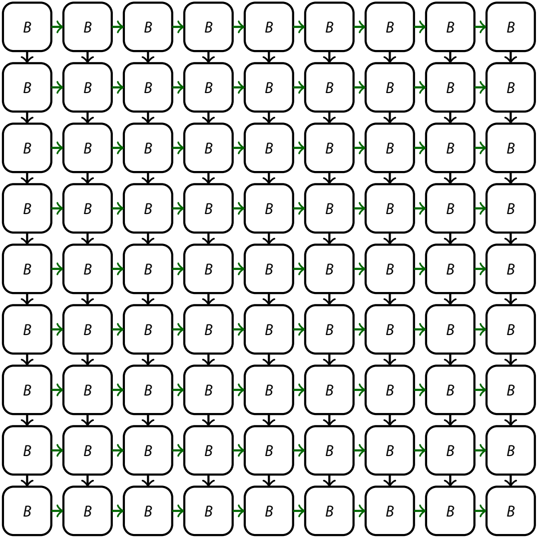
Example Game: Gomoku (Connect–5)



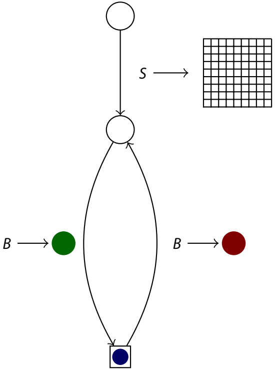


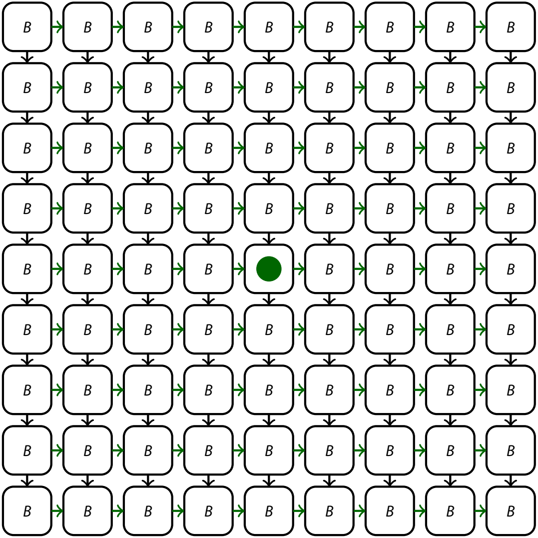
Example Game: Gomoku (Connect–5)



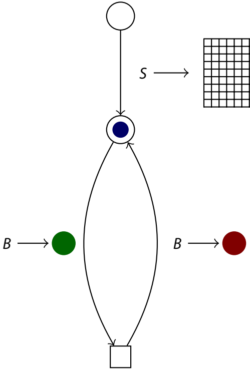


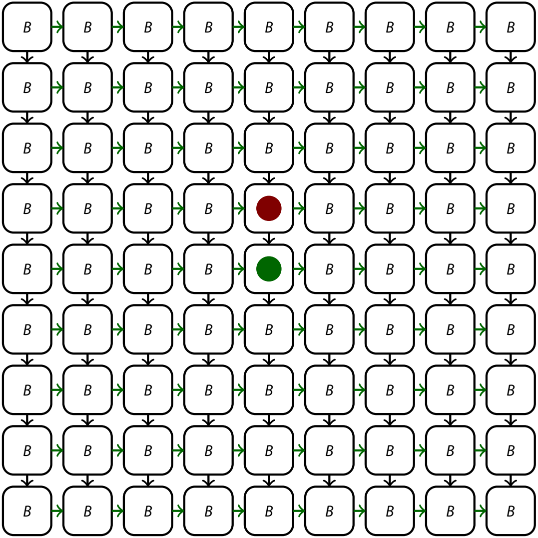
Example Game: Gomoku (Connect–5)



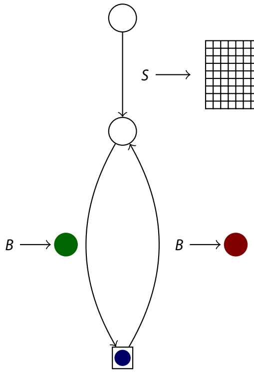


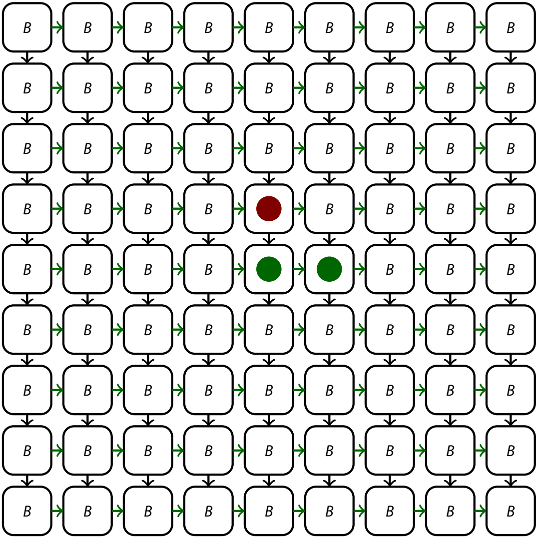
Example Game: Gomoku (Connect–5)



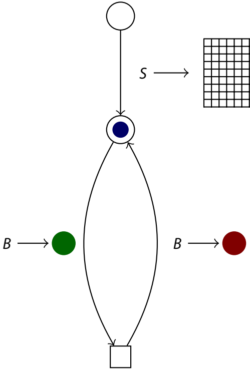


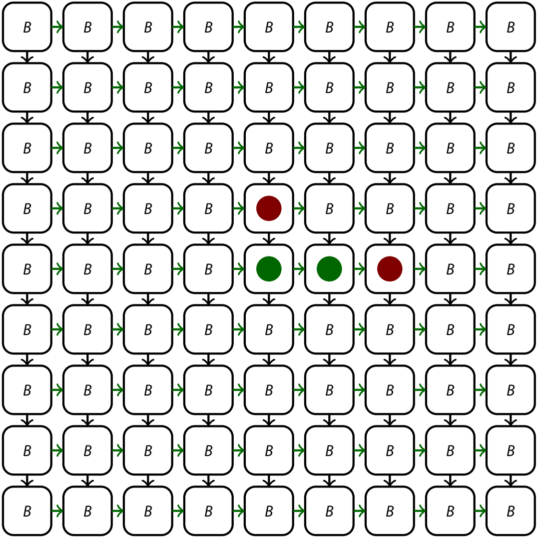
Example Game: Gomoku (Connect–5)



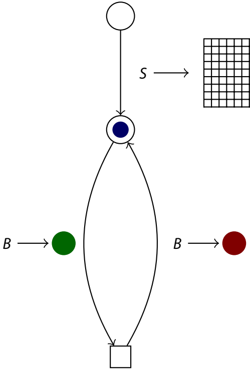


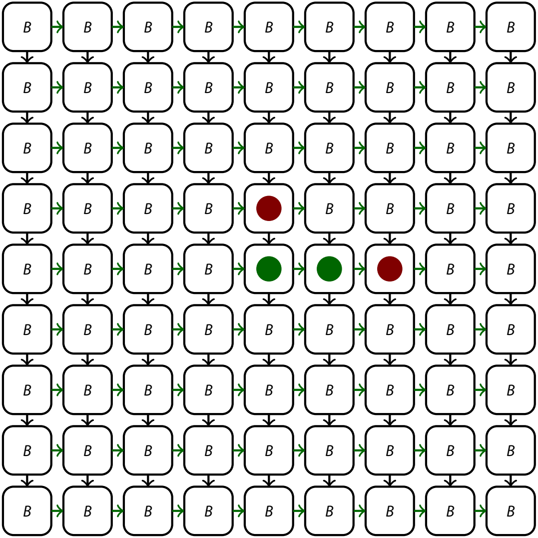
Example Game: Gomoku (Connect–5)





Example Game: Gomoku (Connect–5)





$$\exists x_1 \ldots x_5 \Big(\bigwedge_{1 \le i \le 5} G(x_i) \wedge \Big(\bigwedge_{1 \le i \le 5} R(x_i, x_{i+1}) \vee \bigwedge_{1 \le i \le 5} C(x_i, x_{i+1}) \vee \bigwedge_{1 \le i \le 5} \exists y (R(x_i, y) \wedge C(y, x_{i+1})) \vee \bigwedge_{1 \le i \le 5} \exists y (R(x_i, y) \wedge C(x_{i+1}, y)) \Big) \Big)$$

Overview

Structure Rewriting Systems

- Structure Transition Systems and Games
- Separated Rules and Decidability
- Simulation-Based Playing
- Continuous Dynamics

Quantitative Logics

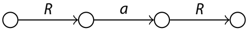
- Standard μ-Calculus
- Quantitative μ-Calculus
- Model Checking on Linear Hybrid Systems
- Model Checking on Increasing Tree Rewriting Systems

Summary

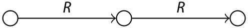
Simple Structure Rewriting

Separated Structures: no element is in two **non-terminal** relations **(Courcelle, Engelfriet, Rozenberg, 1991)**

Separated:

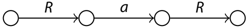


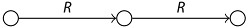
Not Separated:



Simple Structure Rewriting

Separated Structures: no element is in two **non-terminal** relations
(Courcelle, Engelfriet, Rozenberg, 1991)

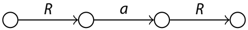
Separated: 

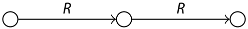
Not Separated: 

Simple Rule $\mathfrak{L} \to \mathfrak{R}$: $\mathfrak{R}$ is **separated** and $\mathfrak{L}$ is a **single tuple in relation**

Simple Structure Rewriting

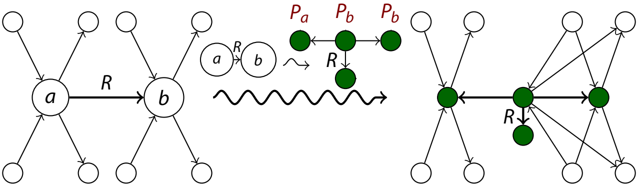
Separated Structures: no element is in two **non-terminal** relations
(Courcelle, Engelfriet, Rozenberg, 1991)

Separated: 

Not Separated: 

Simple Rule $\mathfrak{L} \to \mathfrak{R}$: $\mathfrak{R}$ is **separated** and $\mathfrak{L}$ is a **single tuple in relation**

Example



Decidability of Simple Rewriting Games

Logics

- $L_\mu[\text{MSO}]$**:** Temporal properties expressed in $L\mu$ (subsumes LTL) with properties of structures (states) expressed in MSO
- lim MSO**:** Property of the limit structure expressed in MSO

Theorem

- *Let R be a* ***finite*** *set of* ***(universal) simple structure rewriting rules,***
- *and φ be an $L_\mu[\text{MSO}]$ or lim MSO formula.*

Then the set $\{\pi \in R^\omega \ : \ (\lim)S(\pi) \models \varphi\}$ is ***ω-regular****.*

Corollary

Establishing the winner of (universal) finite simple rewriting games is decidable. The winner has a winning strategy of a simple form.

Proof: Interpreting a Structure in a Tree

Description of how to build $\mathfrak{A}$ is a tree $\mathcal{T}(\mathfrak{A})$ with:

- **Leafs** of **different colours** $1 \ldots k$
- $\oplus$ representing **disjoint sum**
- $i \leftarrow j$ to **change colour** of **all** nodes from i to j
- $e(i, j)$ to **add all pairs** of (i, j)-coloured nodes to e

Proof: Interpreting a Structure in a Tree

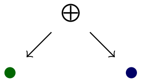
Description of how to build $\mathfrak{A}$ is a tree $\mathcal{T}(\mathfrak{A})$ with:

- **Leafs** of **different colours** $1 \dots k$
- $\oplus$ representing **disjoint sum**
- $i \leftarrow j$ to **change colour** of **all** nodes from i to j
- $e(i, j)$ to **add all pairs** of (i, j)-coloured nodes to e

Proof: Interpreting a Structure in a Tree

Description of how to build $\mathfrak{A}$ is a tree $\mathcal{T}(\mathfrak{A})$ with:

- **Leafs** of **different colours** $1 \ldots k$
- $\oplus$ representing **disjoint sum**
- $i \leftarrow j$ to **change colour** of **all** nodes from i to j
- $e(i, j)$ to **add all pairs** of (i, j)-coloured nodes to e

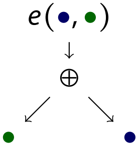


Proof: Interpreting a Structure in a Tree

Description of how to build $\mathfrak{A}$ is a tree $\mathcal{T}(\mathfrak{A})$ with:

- **Leafs** of **different colours** $1 \ldots k$
- $\oplus$ representing **disjoint sum**
- $i \leftarrow j$ to **change colour** of **all** nodes from i to j
- $e(i, j)$ to **add all pairs** of (i, j)-coloured nodes to e

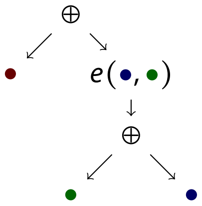
$\bullet \longrightarrow \bullet$



Proof: Interpreting a Structure in a Tree

Description of how to build $\mathfrak{A}$ is a tree $\mathcal{T}(\mathfrak{A})$ with:

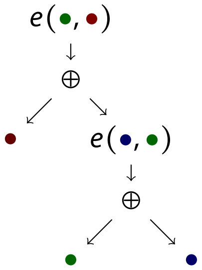
- **Leafs** of **different colours** $1 \ldots k$
- $\oplus$ representing **disjoint sum**
- $i \leftarrow j$ to **change colour** of **all** nodes from i to j
- $e(i, j)$ to **add all pairs** of (i, j)-coloured nodes to e



Proof: Interpreting a Structure in a Tree

Description of how to build $\mathfrak{A}$ is a tree $\mathcal{T}(\mathfrak{A})$ with:

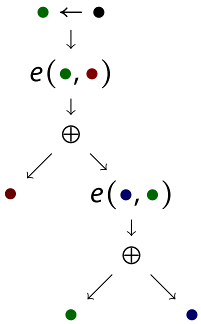
- **Leafs** of **different colours** $1 \ldots k$
- $\oplus$ representing **disjoint sum**
- $i \leftarrow j$ to **change colour** of **all** nodes from i to j
- $e(i, j)$ to **add all pairs** of (i, j)-coloured nodes to e



Proof: Interpreting a Structure in a Tree

Description of how to build $\mathfrak{A}$ is a tree $\mathcal{T}(\mathfrak{A})$ with:

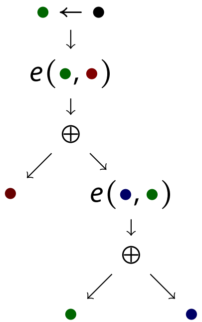
- **Leafs** of **different colours** $1 \dots k$
- $\oplus$ representing **disjoint sum**
- $i \leftarrow j$ to **change colour** of **all** nodes from i to j
- $e(i, j)$ to **add all pairs** of (i, j)-coloured nodes to e



Proof: Interpreting a Structure in a Tree

Description of how to build $\mathfrak{A}$ is a tree $\mathcal{T}(\mathfrak{A})$ with:

- **Leafs** of **different colours** $1 \dots k$
- $\oplus$ representing **disjoint sum**
- $i \leftarrow j$ to **change colour** of **all** nodes from i to j
- $e(i, j)$ to **add all pairs** of (i, j)-coloured nodes to e



Theorem:
For every k there is an MSO-**to**-MSO **interpretation** $\mathcal{I}$ such that for all structures $\mathfrak{A}$ of **clique-width** $\leq k$ holds $\mathcal{I}(\mathcal{T}(\mathfrak{A})) \cong \mathfrak{A}$.

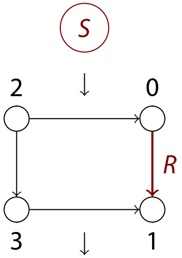
Proof: Simple Rewriting in the Tree

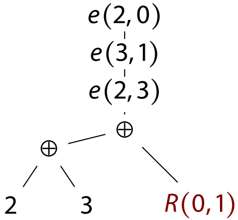
S

$\downarrow$

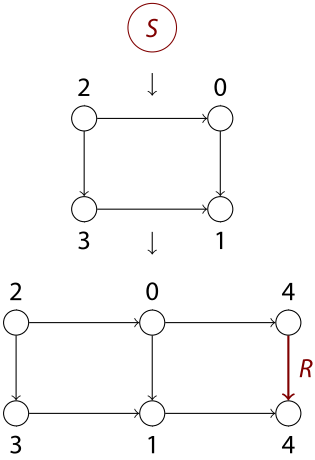
S

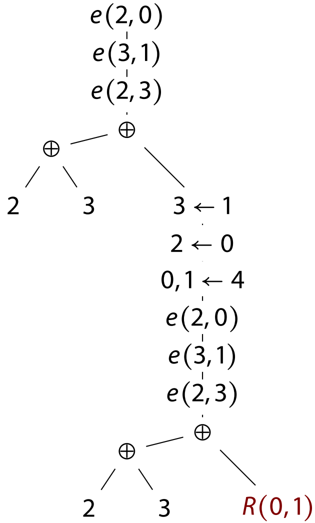
Proof: Simple Rewriting in the Tree



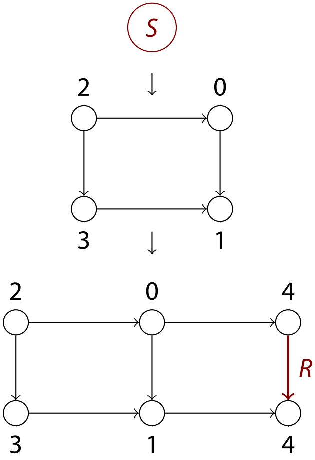


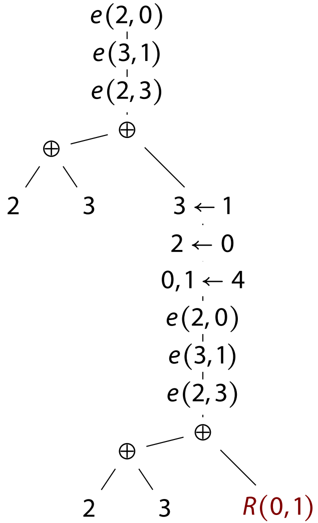
Proof: Simple Rewriting in the Tree





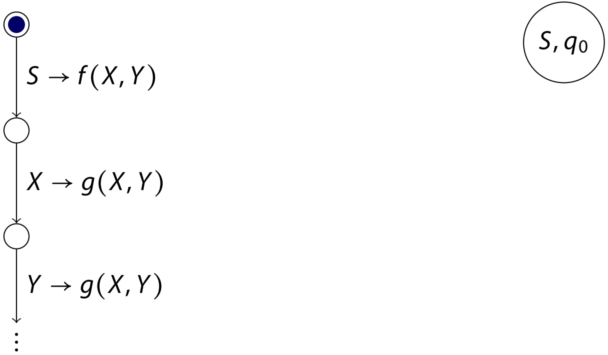
Proof: Simple Rewriting in the Tree



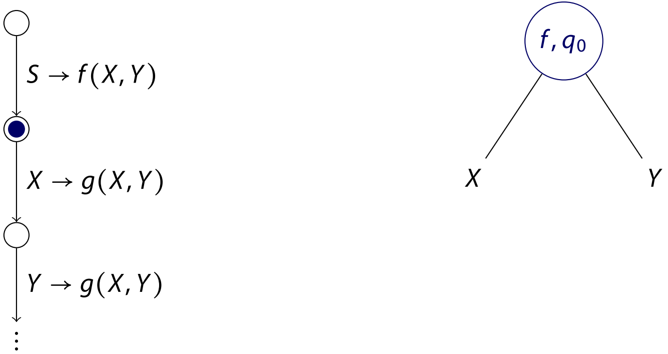


MSO-to-MSO interpretation: $\varphi \to \psi$

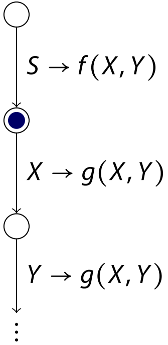
Proof: From Tree to Alternating Word Automata

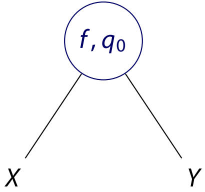


Proof: From Tree to Alternating Word Automata



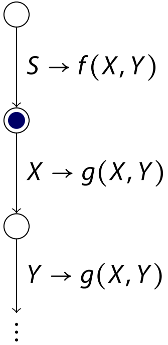
Proof: From Tree to Alternating Word Automata

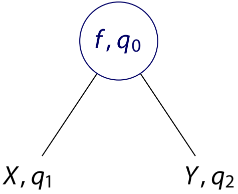




existential: pick transition

Proof: From Tree to Alternating Word Automata

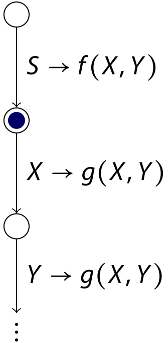


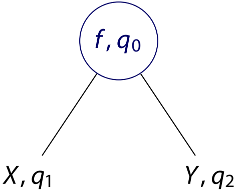


existential: pick transition

$f, q_0 \to (q_1, q_2)$

Proof: From Tree to Alternating Word Automata



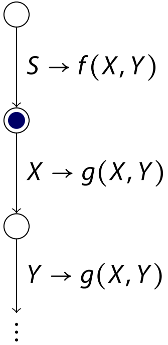


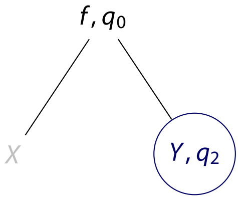
existential: pick transition

$f, q_0 \to (q_1, q_2)$

universal: left or right

Proof: From Tree to Alternating Word Automata



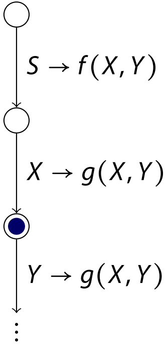


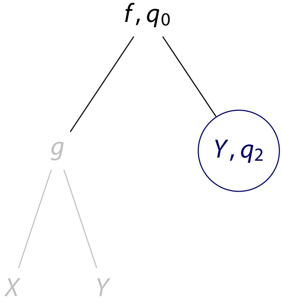
existential: pick transition

$f, q_0 \to (q_1, q_2)$

universal: left or right

Proof: From Tree to Alternating Word Automata





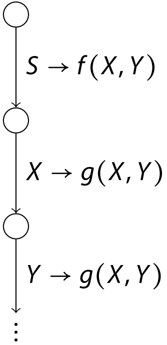
existential: pick transition

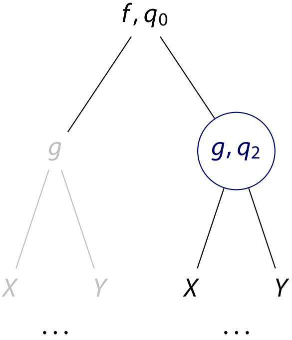
universal: left or right

$f, q_0 \to (q_1, q_2)$

ignore

Proof: From Tree to Alternating Word Automata





existential: pick transition

universal: left or right

$f, q_0 \to (q_1, q_2)$

ignore

Overview

Motivation

Implementing the Decidability Result

- **Tool**: **MONA**
  - Developed at **BRICS** since **1996** by Nils Klarlund and Anders Møller
  - Symbolic representation with **BDDs**
  - **Minimisation** at each step
- **Example:** a simple **tic-tac-toe** game $\rightsquigarrow$ **memory overflow**
- **Problems** due to **inefficient coding**
  - **Bounded clique-width graphs** not good for MONA
  - Only **universal interpretation** decidable, must encode games

Motivation

Implementing the Decidability Result

- **Tool**: **MONA**
  - Developed at **BRICS** since **1996** by Nils Klarlund and Anders Møller
  - Symbolic representation with **BDDs**
  - **Minimisation** at each step
- **Example:** a simple **tic-tac-toe** game $\rightsquigarrow$ **memory overflow**
- **Problems** due to **inefficient coding**
  - **Bounded clique-width graphs** not good for MONA
  - Only **universal interpretation** decidable, must encode games

Simulation-Based Game Playing (Joint work with Ł. Stafiniak)

Motivation

Implementing the Decidability Result

- **Tool**: **MONA**
  - Developed at **BRICS** since **1996** by Nils Klarlund and Anders Møller
  - Symbolic representation with **BDDs**
  - **Minimisation** at each step
- **Example:** a simple **tic-tac-toe** game $\rightsquigarrow$ **memory overflow**
- **Problems** due to **inefficient coding**
  - **Bounded clique-width graphs** not good for MONA
  - Only **universal interpretation** decidable, must encode games

Simulation-Based Game Playing (Joint work with Ł. Stafiniak)

How to determine the **value of a position** v in a **general** game?

Two Approaches

- **UCT** based algorithms
- **Generate** heuristic evaluation functions

UCT: Upper Confidence Bounds for Trees

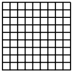
Building a Tree during Random Plays

- **Idea:** memorise first random moves, play **minimax** there
- **History:** encouraged by the success of **MoGo**

UCT: Upper Confidence Bounds for Trees

Building a Tree during Random Plays

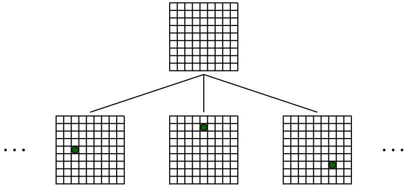
- **Idea:** memorise first random moves, play **minimax** there
- **History:** encouraged by the success of **MoGo**



UCT: Upper Confidence Bounds for Trees

Building a Tree during Random Plays

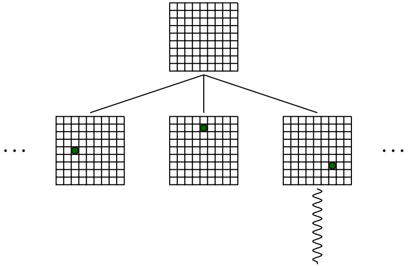
- **Idea:** memorise first random moves, play **minimax** there
- **History:** encouraged by the success of **MoGo**



UCT: Upper Confidence Bounds for Trees

Building a Tree during Random Plays

- **Idea:** memorise first random moves, play **minimax** there
- **History:** encouraged by the success of **MoGo**



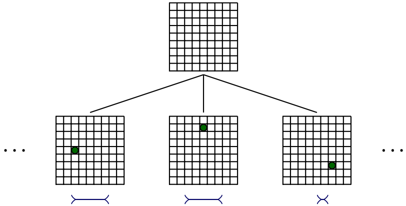
UCT: Upper Confidence Bounds for Trees

Building a Tree during Random Plays

- **Idea:** memorise first random moves, play **minimax** there
- **History:** encouraged by the success of **MoGo**

Pick Max Upper Confidence

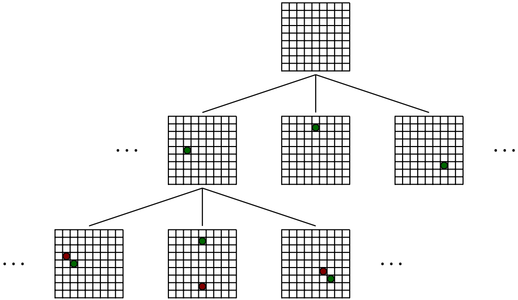
$$C \cdot \sqrt{\frac{\ln(n(v)+1)}{n(w)+1}}$$



UCT: Upper Confidence Bounds for Trees

Building a Tree during Random Plays

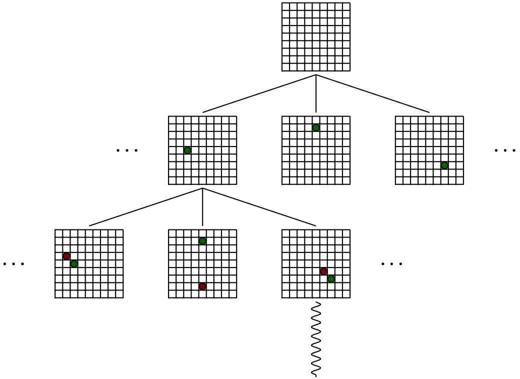
- **Idea:** memorise first random moves, play **minimax** there
- **History:** encouraged by the success of **MoGo**



UCT: Upper Confidence Bounds for Trees

Building a Tree during Random Plays

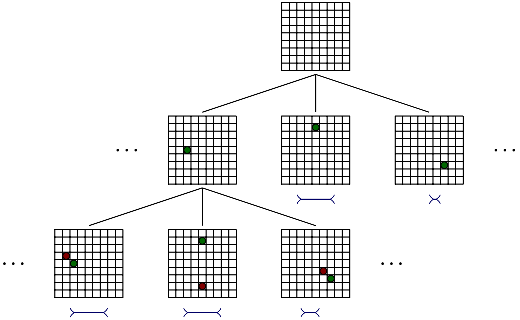
- **Idea:** memorise first random moves, play **minimax** there
- **History:** encouraged by the success of **MoGo**



UCT: Upper Confidence Bounds for Trees

Building a Tree during Random Plays

- **Idea:** memorise first random moves, play **minimax** there
- **History:** encouraged by the success of **MoGo**



Deriving Heuristic Evaluation Functions

Methods

- Goal expansion and Type Normal Form
- Summing over conjuncts in existentially quantified conjunctions
- Retain stable guards and include rule preconditions

Example: Tic-Tac-Toe without Diagonals

$$\big(P(x)\wedge P(y)\wedge P(z)\wedge R(x,y)\wedge R(y,z)\big)\vee\big(P(x)\wedge P(y)\wedge P(z)\wedge C(x,y)\wedge C(y,z)\big)$$

$$\rightsquigarrow$$

$$\sum_{x|P(x)}\Big(\tfrac{1}{8}+\sum_{y|P(y)\wedge R(x,y)}\big(\tfrac{1}{4}+\sum_{z|P(z)\wedge R(y,z)}1\big)\Big)+\sum_{x|P(x)}\Big(\tfrac{1}{8}+\sum_{y|P(y)\wedge C(x,y)}\big(\tfrac{1}{4}+\sum_{z|P(z)\wedge C(y,z)}1\big)\Big)$$

Deriving Heuristic Evaluation Functions

Methods

- Goal expansion and Type Normal Form
- Summing over conjuncts in existentially quantified conjunctions
- Retain stable guards and include rule preconditions

Example: Tic-Tac-Toe without Diagonals

$$\big(P(x) \wedge P(y) \wedge P(z) \wedge R(x,y) \wedge R(y,z)\big) \vee \big(P(x) \wedge P(y) \wedge P(z) \wedge C(x,y) \wedge C(y,z)\big)$$

$$\rightsquigarrow$$

$$\textstyle\sum_{x|P(x)} \Big(\frac{1}{8} + \sum_{y|P(y) \wedge R(x,y)} \big(\frac{1}{4} + \sum_{z|P(z) \wedge R(y,z)} 1\big)\Big) + \sum_{x|P(x)} \Big(\frac{1}{8} + \sum_{y|P(y) \wedge C(x,y)} \big(\frac{1}{4} + \sum_{z|P(z) \wedge C(y,z)} 1\big)\Big)$$

Results vs. Fluxplayer

	Toss Wins	Fluxplayer Wins	Tie
Breakthrough	95%	5%	0%
Connect4	20%	75%	5%
Connect5	0%	0%	100%
Pawn Whopping	50%	50%	0%

(fixed depth)

Deriving Heuristic Evaluation Functions

Methods

- Goal expansion and Type Normal Form
- Summing over conjuncts in existentially quantified conjunctions
- Retain stable guards and include rule preconditions

Example: Tic-Tac-Toe without Diagonals

$$\big(P(x)\wedge P(y)\wedge P(z)\wedge R(x,y)\wedge R(y,z)\big)\vee\big(P(x)\wedge P(y)\wedge P(z)\wedge C(x,y)\wedge C(y,z)\big)$$

$$\leadsto$$

$$\textstyle\sum_{x|P(x)}\Big(\frac{1}{8}+\sum_{y|P(y)\wedge R(x,y)}\big(\frac{1}{4}+\sum_{z|P(z)\wedge R(y,z)}1\big)\Big)+\sum_{x|P(x)}\Big(\frac{1}{8}+\sum_{y|P(y)\wedge C(x,y)}\big(\frac{1}{4}+\sum_{z|P(z)\wedge C(y,z)}1\big)\Big)$$

Results vs. Fluxplayer

(variable depth)

	Toss Wins	Fluxplayer Wins	Tie
Breakthrough	95%	5%	0%
Connect4	45%	20%	35%
Connect5	0%	0%	100%
Pawn Whopping	60%	40%	0%

Overview

Continuous Rewriting

Metafinite structures: $\mathfrak{A} = (A, R_1, \dots, R_k, f_1, \dots, f_l)$ with $f_i : A \to \mathbb{R}$

Additional Parameters to a Rule:

- **dynamics**: system of **ordinary differential equations**
- **updates**: equations assigning **values on the right-hand side**
- **constraints**: precondition, invariant, postcondition

Continuous Rewriting

Metafinite structures: $\mathfrak{A} = (A, R_1, \ldots, R_k, f_1, \ldots, f_l)$ with $f_i : A \to \mathbb{R}$

Additional Parameters to a Rule:

- **dynamics**: system of **ordinary differential equations**
- **updates**: equations assigning **values on the right-hand side**
- **constraints**: precondition, invariant, postcondition

Logic

- **Monadic Second-Order Logic** (MSO):
$$\forall X(x \in X \wedge (\forall z, v(z \in X \wedge R(z,v) \to v \in X)) \to y \in X)$$
- **Real-valued terms with counting**: $2 \cdot \chi\big(\exists y(P(y) \wedge R(x,y))\big) + f(x)$
- **Real quantification**: $\exists a \in \mathbb{R}(a^2 \cdot f(x) + a - 1 = 0) \wedge (f(x) > 2)$

Continuous Rewriting

Metafinite structures: $\mathfrak{A} = (A, R_1, \ldots, R_k, f_1, \ldots, f_l)$ with $f_i : A \to \mathbb{R}$

Additional Parameters to a Rule:

- **dynamics**: system of **ordinary differential equations**
- **updates**: equations assigning **values on the right-hand side**
- **constraints**: precondition, invariant, postcondition

Logic

- **Monadic Second-Order Logic** (MSO):
$$\forall X(x \in X \land (\forall z, v(z \in X \land R(z,v) \to v \in X)) \to y \in X)$$
- **Real-valued terms with counting**: $2 \cdot \chi\big(\exists y(P(y) \land R(x,y))\big) + f(x)$
- **Real quantification**: $\exists a \in \mathbb{R}(a^2 \cdot f(x) + a - 1 = 0) \land (f(x) > 2)$

Semantics of Application

(1) All dynamics applies **concurrently**

(2) Many **universal** rules with **trivial** discrete part

(3) Single non-trivial discrete rewriting **after** continuous evolution

Games with Continuous Dynamics

Moves: the player chooses

- the **rule** and a **match**
- the **time** from an interval
- **parameters** for all rules from allowed intervals

Games with Continuous Dynamics

Moves: the player chooses

- the **rule** and a **match**
- the **time** from an interval
- **parameters** for all rules from allowed intervals

Payoffs

- **logics** as described before
- evaluated on the **final state**
- **Example:** value of f on the matched element
  $\leadsto$ **parameter optimisation**

Overview

Structure Rewriting Systems

- Structure Transition Systems and Games
- Separated Rules and Decidability
- Simulation-Based Playing
- Continuous Dynamics

Quantitative Logics

- Standard μ-Calculus
- Quantitative μ-Calculus
- Model Checking on Linear Hybrid Systems
- Model Checking on Increasing Tree Rewriting Systems

Summary

Overview

The μ-Calculus

Syntax:

$$\varphi ::= P_i \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box\,\varphi \mid \Diamond\,\varphi \mid \mu X.\varphi \mid \nu X.\varphi$$

Evaluation of $\mu X.\varphi(X)$:

set $X_0 = \emptyset$ and compute $X_{i+1} := \varphi(X_i)$ until $X_{i+1} = X_i$ **(Knaster-Tarski)**

Example:

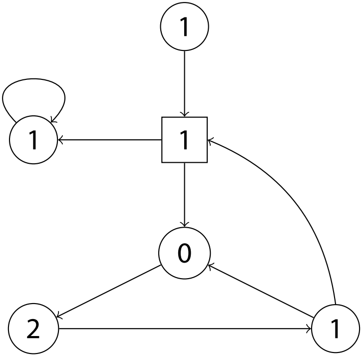
$$P \text{ until } Q := \mu X.(Q \vee (P \wedge \Diamond X))$$

Motivation for μ-calculus:

- The μ-calculus is more expressive than LTL and CTL
- Can express all **bisimulation invariant MSO** properties
- The connection between μ-calculus and **parity games**

Parity Games

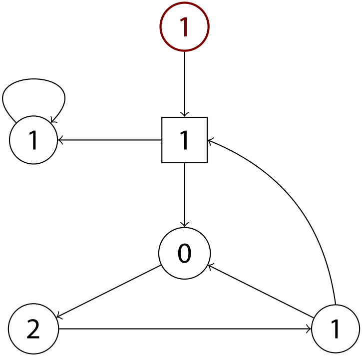
$\mathcal{G} = (V, V_0, V_1, E, \Omega)$ and $vE \neq \varnothing$ for all $v \in V$



Player 0 wins $\mathcal{G}$ from v_0 when she has a strategy σ so that for all strategies ρ of Pl. 1 the minimal colour appearing infinitely often on $\pi_{v_0}(\sigma, \rho)$ is **even**.

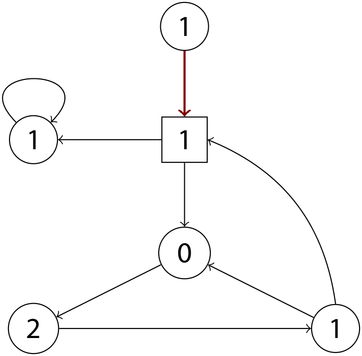
Parity Games

$\mathcal{G} = (V, V_0, V_1, E, \Omega)$ and $vE \neq \emptyset$ for all $v \in V$



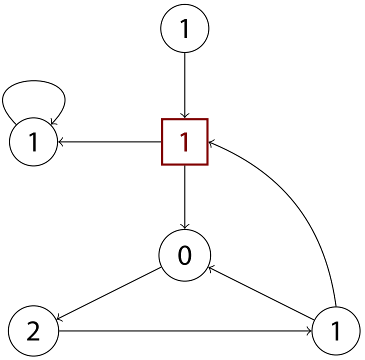
Parity Games

$\mathcal{G} = (V, V_0, V_1, E, \Omega)$ and $vE \neq \varnothing$ for all $v \in V$



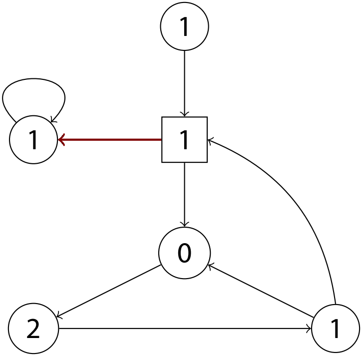
Parity Games

$\mathcal{G} = (V, V_0, V_1, E, \Omega)$ and $vE \neq \varnothing$ for all $v \in V$



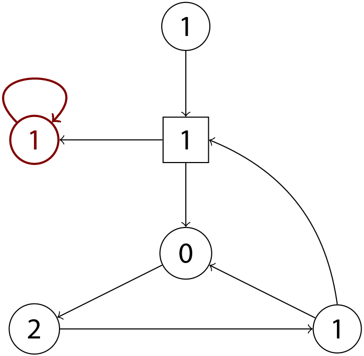
Parity Games

$\mathcal{G} = (V, V_0, V_1, E, \Omega)$ and $vE \neq \emptyset$ for all $v \in V$



Parity Games

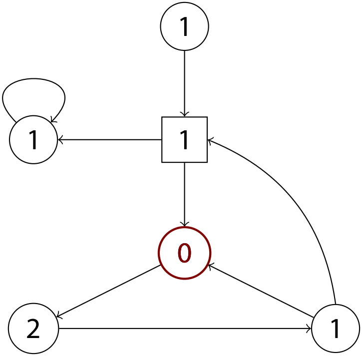
$\mathcal{G} = (V, V_0, V_1, E, \Omega)$ and $vE \neq \emptyset$ for all $v \in V$



Outcome: π won by Abélard since the lowest colour on the cycle is odd.

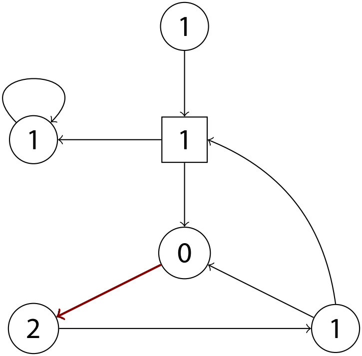
Parity Games

$\mathcal{G} = (V, V_0, V_1, E, \Omega)$ and $vE \neq \varnothing$ for all $v \in V$



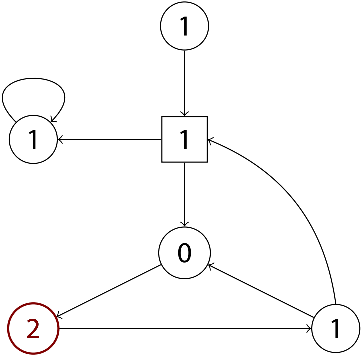
Parity Games

$\mathcal{G} = (V, V_0, V_1, E, \Omega)$ and $vE \neq \emptyset$ for all $v \in V$



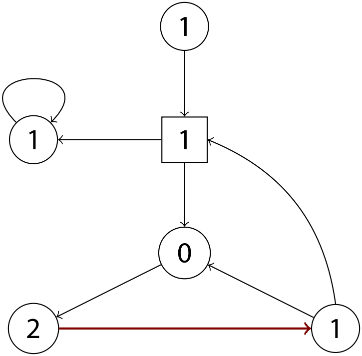
Parity Games

$\mathcal{G} = (V, V_0, V_1, E, \Omega)$ and $vE \neq \emptyset$ for all $v \in V$



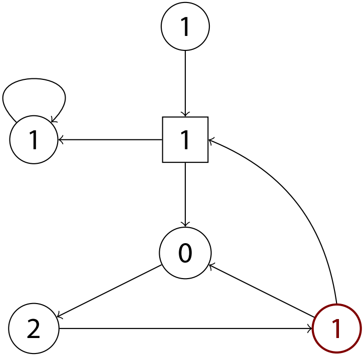
Parity Games

$\mathcal{G} = (V, V_0, V_1, E, \Omega)$ and $vE \neq \varnothing$ for all $v \in V$



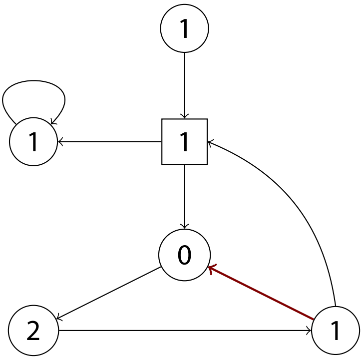
Parity Games

$\mathcal{G} = (V, V_0, V_1, E, \Omega)$ and $vE \neq \emptyset$ for all $v \in V$



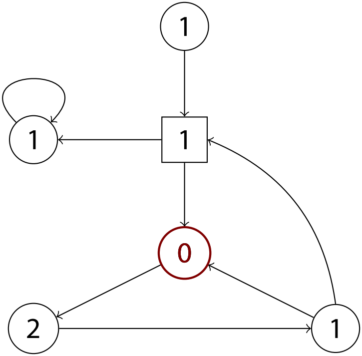
Parity Games

$\mathcal{G} = (V, V_0, V_1, E, \Omega)$ and $vE \neq \emptyset$ for all $v \in V$



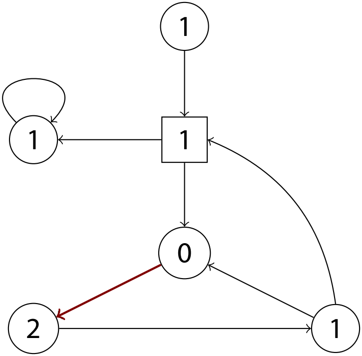
Parity Games

$\mathcal{G} = (V, V_0, V_1, E, \Omega)$ and $vE \neq \emptyset$ for all $v \in V$



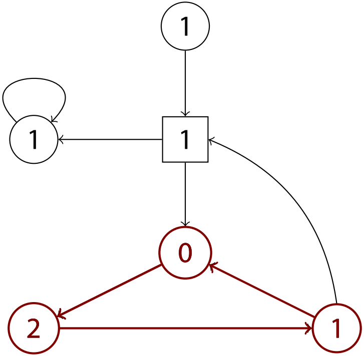
Parity Games

$\mathcal{G} = (V, V_0, V_1, E, \Omega)$ and $vE \neq \emptyset$ for all $v \in V$



Parity Games

$\mathcal{G} = (V, V_0, V_1, E, \Omega)$ and $vE \neq \emptyset$ for all $v \in V$

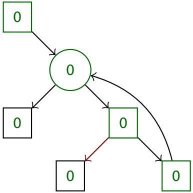


Outcome: π won by Eloïse since the lowest colour on the cycle is even.

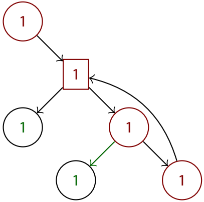
One or Two Colours

One Colour

Safety Games

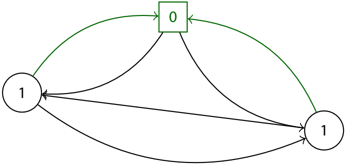


Reachability Games

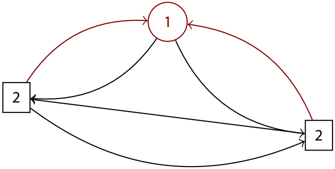


Two Colours

Büchi Games

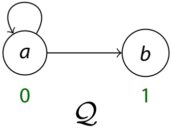


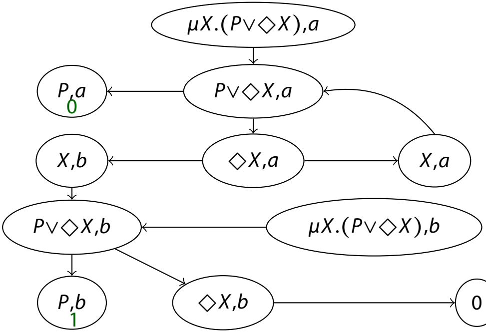
Co-Büchi Games



Parity Games and the μ-Calculus

Model Checking Games e.g. $\mathrm{MC}[\mathcal{Q}, \varphi]$ for $\mathcal{Q}$ and $\varphi = \mu X.(P \vee \Diamond X)$





Parity games are model checking games for $L\mu$

Overview

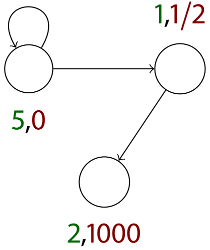
Quantitative Transition Systems

Quantitative Transition System (QTS):

$$\mathcal{Q} = (V, E, P_1, P_2, \ldots, P_n)$$

$$P_i : V \to \mathbb{R}_\infty, \quad \text{not } V \to \{\top, \bot\}$$

Example: QTS with quantitative predicates P and Q



Quantitative μ-Calculus

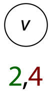
Syntax:

$$\varphi ::= P_i \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box\varphi \mid \Diamond\varphi \mid \mu X.\varphi \mid \nu X.\varphi$$

Semantics:

- evaluation on QTS
- $[\![\varphi]\!]^{\mathcal{K}} : V \to \mathbb{R}_\infty$
- $\wedge \rightsquigarrow$ min
- $\vee \rightsquigarrow$ max

Example:



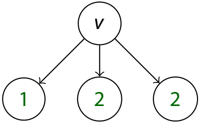
$$[\![P \wedge Q]\!](v) = 2$$

Evaluation of $\Box$ and $\Diamond$

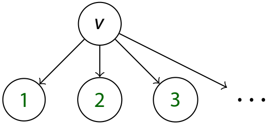
Intuition:

- $\Box \rightsquigarrow$ inf
- $\Diamond \rightsquigarrow$ sup
- min and max for finitely branching systems

Example:



$[\![\Box P]\!](v) = 1$



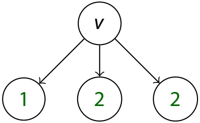
$[\![\Diamond P]\!](v) = \infty$

Evaluation of $\Box$ and $\Diamond$

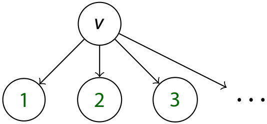
Intuition:

- $\Box \rightsquigarrow \inf$
- $\Diamond \rightsquigarrow \sup$
- min and max for finitely branching systems

Example:



$[\![\Box P]\!](v) = 1$



$[\![\Diamond P]\!](v) = \infty$

Formally:

- $[\![\Diamond\varphi]\!]^{\mathcal{K}}(v) = \sup_{v' \in vE} [\![\varphi]\!]^{\mathcal{K}}(v')$
- $[\![\Box\varphi]\!]^{\mathcal{K}}(v) = \inf_{v' \in vE} [\![\varphi]\!]^{\mathcal{K}}(v')$

Evaluation of Fixed Points

Intuition:

- Lattice $(\mathcal{F}, \leq)$
- $\mathcal{F} := \{f : f \text{ is a function from } V \text{ to } \mathbb{R}_\infty\}$
- top element ∞, bottom element $-\infty$
- **Theorem of Knaster and Tarski** applies

Inductive evaluation of μ:

$$g_0 = 0$$

$$g_\alpha = \begin{cases} [\![\varphi]\!]_{\varepsilon[X \leftarrow g_{\alpha-1}]} & \text{for } \alpha \text{ successor ordinal,} \\ \lim_{\beta<\alpha} [\![\varphi]\!]_{\varepsilon[X \leftarrow g_\beta]} & \text{for } \alpha \text{ limit ordinal,} \end{cases}$$

$$[\![\mu X.\varphi]\!]^{\mathcal{K}}_\varepsilon = g_\gamma \text{ where } g_\gamma = g_{\gamma+1}$$

Evaluation of Fixed Points

Intuition:

- Lattice $(\mathcal{F}, \leq)$
- $\mathcal{F} := \{f : f \text{ is a function from } V \text{ to } \mathbb{R}_\infty\}$
- top element ∞, bottom element $-\infty$
- **Theorem of Knaster and Tarski** applies

Inductive evaluation of μ:

$$g_0 = 0$$

$$g_\alpha = \begin{cases} [\![\varphi]\!]_{\varepsilon[X \leftarrow g_{\alpha-1}]} & \text{for } \alpha \text{ successor ordinal,} \\ \lim_{\beta<\alpha} [\![\varphi]\!]_{\varepsilon[X \leftarrow g_\beta]} & \text{for } \alpha \text{ limit ordinal,} \end{cases}$$

$$[\![\mu X.\varphi]\!]_\varepsilon^{\mathcal{K}} = g_\gamma \text{ where } g_\gamma = g_{\gamma+1}$$

Formally:

- $[\![\mu X.\varphi]\!]_\varepsilon^{\mathcal{K}} = \inf\{f \in \mathcal{F} : f = [\![\varphi]\!]_{\varepsilon[X \leftarrow f]}^{\mathcal{K}}\}$
- $[\![\nu X.\varphi]\!]_\varepsilon^{\mathcal{K}} = \sup\{f \in \mathcal{F} : f = [\![\varphi]\!]_{\varepsilon[X \leftarrow f]}^{\mathcal{K}}\}$

Quantitative μ-Calculus

Syntax: $\varphi ::= P_i \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box\, \varphi \mid \Diamond\, \varphi \mid \mu X.\varphi \mid \nu X.\varphi$

Semantics:
Evaluation on quantitative transitions system, $[\![\varphi]\!]^{\mathcal{K}} : V \to \mathbb{R}_\infty$

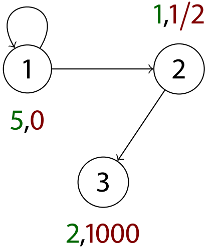
Quantitative μ-Calculus

Syntax: $\varphi ::= P_i \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box\,\varphi \mid \Diamond\,\varphi \mid \mu X.\varphi \mid \nu X.\varphi$

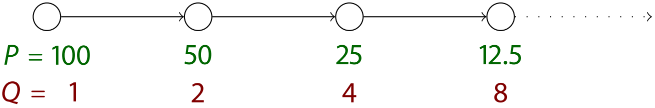
Semantics:
Evaluation on quantitative transitions system, $[\![\varphi]\!]^{\mathcal{K}} : V \to \mathbb{R}_\infty$

- $[\![\varphi \wedge \psi]\!] = \min\{[\![\varphi]\!], [\![\psi]\!]\}$
- $[\![\varphi \vee \psi]\!] = \max\{[\![\varphi]\!], [\![\psi]\!]\}$
- $[\![\Diamond\varphi]\!] = \sup[\![\varphi]\!](succ)$
- $[\![\Box\varphi]\!] = \inf[\![\varphi]\!](succ)$
- inductive evaluation of fixed points over lattice $(\mathcal{F}, \leq)$

QTS $\mathcal{Q}_1 = (V, E, P, Q)$



Basic Example in Quantitative μ-Calculus

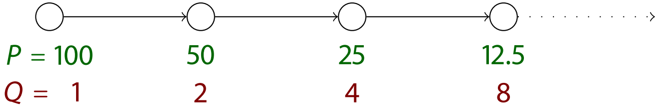


$$P \text{ until } Q := \mu X.(Q \vee (P \wedge \Diamond X))$$

Inductive Evaluation:

- $[\![P \text{ until } Q]\!]_0(v) = -\infty$,
- $[\![P \text{ until } Q]\!]_1(v) = Q(v)$,
- $[\![P \text{ until } Q]\!]_2(v) = \max\{Q(v), \min\{P(v), \max_{w \in vE}\{Q(w)\}\}$
- $\ldots$

Basic Example in Quantitative μ-Calculus



$$P \text{ until } Q := \mu X.(Q \vee (P \wedge \Diamond X))$$

Inductive Evaluation:

- $[\![P \text{ until } Q]\!]_0(v) = -\infty$,
- $[\![P \text{ until } Q]\!]_1(v) = Q(v)$,
- $[\![P \text{ until } Q]\!]_2(v) = \max\{Q(v), \min\{P(v), \max_{w \in vE}\{Q(w)\}\}$
- $\ldots$

Intuition: Value of P at the last time $P > Q$

Overview

Linear Hybrid Systems

Finite representation of an infinite QTS

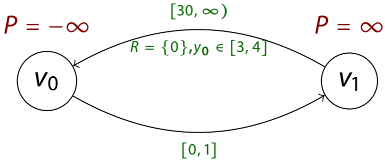
- Variables $y_1, \ldots, y_k$ evolve in time, in v by $\frac{dy_i}{dt} = \delta_i(v)$
- Transitions are labelled by triples $(I, \overline{C}, R)$
  - Interval I: possible **period of time** to stay in v
  - Vector $\overline{C}$: interval **constraints** on the variables $\overline{y}$
  - Set R: variables to **reset** after the transition

Quantitative μ-Calculus on LHS

$$\varphi ::= y_j \mid P_i \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box\, \varphi \mid \Diamond\, \varphi \mid \mu X.\varphi \mid \nu X.\varphi$$

Semantics defined on the **represented QTS** (where y_j are predicates).

Example



Main Result

Definition

A LHS is **initialised** if for each transition (v, l, w) and variable y_i

$$\delta_i(v) \neq \delta_i(w) \implies i \in R_l$$

Intuition: $\mathrm{Q}\mu$ can be **approximated** on **initialised** LHS

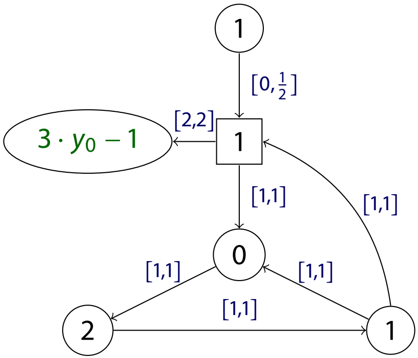
Theorem

- *Let $\mathcal{K}$ be an initialised LHS over $\mathbb{Q}$,*
- *and φ a quantitative μ-calculus formula,*
- *and $n > 0$ an integer (approximation quality).*

It is decidable whether $[\![\varphi]\!]^{\mathcal{K}} = \infty$, $[\![\varphi]\!]^{\mathcal{K}} = -\infty$, and else a number $r \in \mathbb{Q}$ can be computed such that $|[\![\varphi]\!]^{\mathcal{K}} - r| < \frac{1}{n}$.

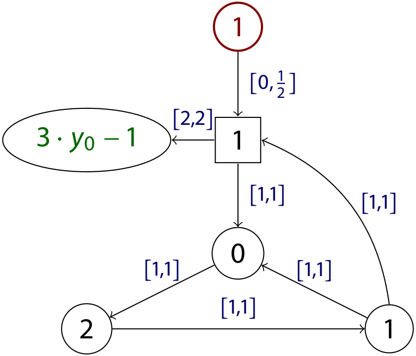
Interval Parity Games

An **interval parity game** $\mathcal{G}$ is played on an **LHS** with priorities in locations



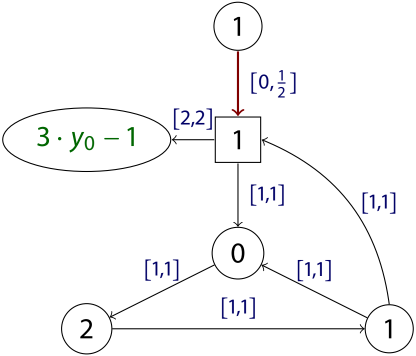
Interval Parity Games

An **interval parity game** $\mathcal{G}$ is played on an **LHS** with priorities in locations



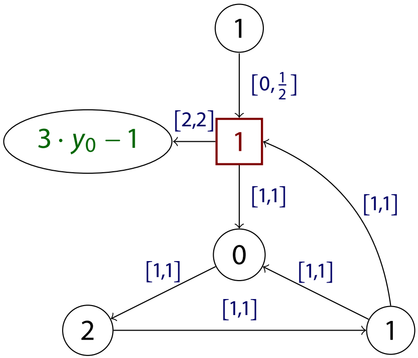
Interval Parity Games

An **interval parity game** $\mathcal{G}$ is played on an **LHS** with priorities in locations



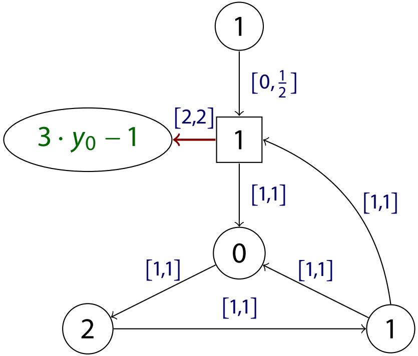
Interval Parity Games

An **interval parity game** $\mathcal{G}$ is played on an **LHS** with priorities in locations



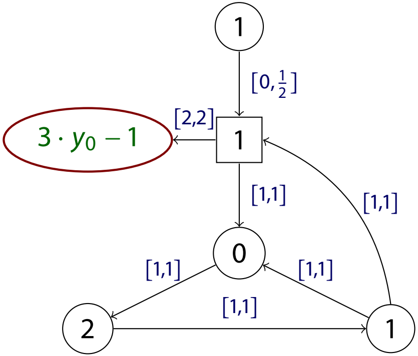
Interval Parity Games

An **interval parity game** $\mathcal{G}$ is played on an **LHS** with priorities in locations



Interval Parity Games

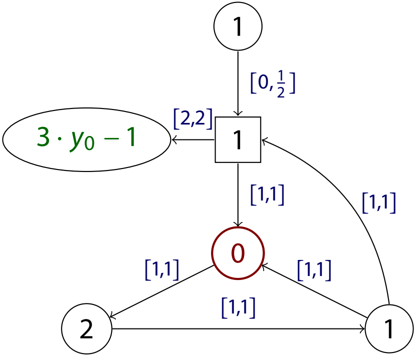
An **interval parity game** $\mathcal{G}$ is played on an **LHS** with priorities in locations



Outcome: $p(\pi) = 3 \cdot (\frac{1}{3} + 2) - 1$

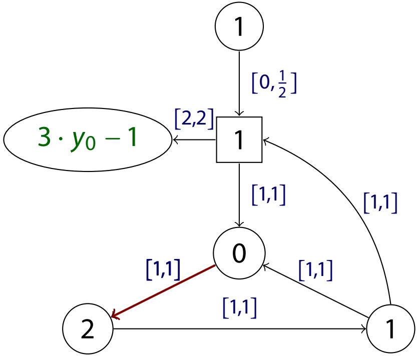
Interval Parity Games

An **interval parity game** $\mathcal{G}$ is played on an **LHS** with priorities in locations



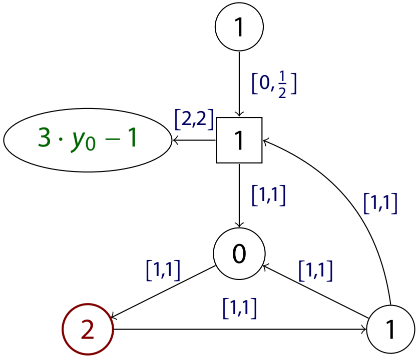
Interval Parity Games

An **interval parity game** $\mathcal{G}$ is played on an **LHS** with priorities in locations



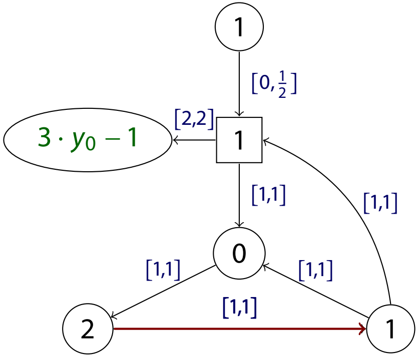
Interval Parity Games

An **interval parity game** $\mathcal{G}$ is played on an **LHS** with priorities in locations



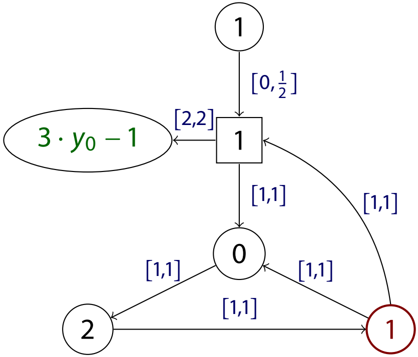
Interval Parity Games

An **interval parity game** $\mathcal{G}$ is played on an **LHS** with priorities in locations



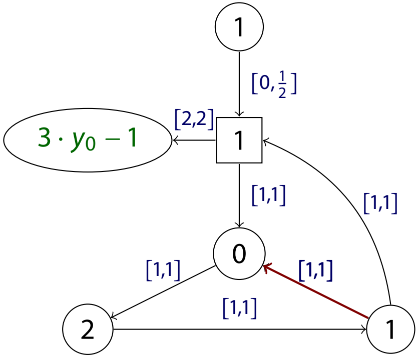
Interval Parity Games

An **interval parity game** $\mathcal{G}$ is played on an **LHS** with priorities in locations



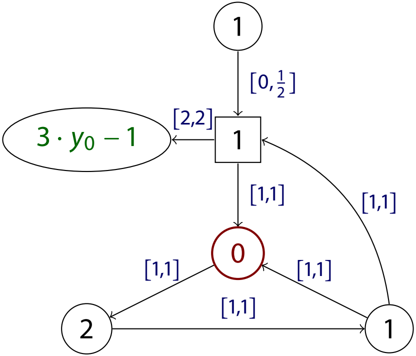
Interval Parity Games

An **interval parity game** $\mathcal{G}$ is played on an **LHS** with priorities in locations



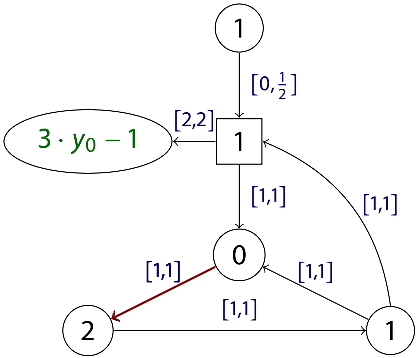
Interval Parity Games

An **interval parity game** $\mathcal{G}$ is played on an **LHS** with priorities in locations



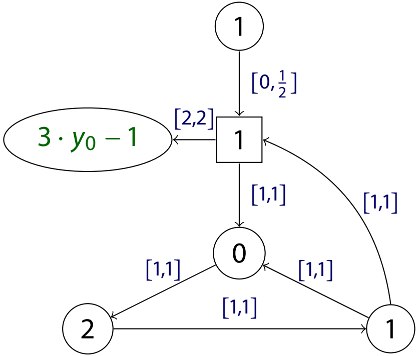
Interval Parity Games

An **interval parity game** $\mathcal{G}$ is played on an **LHS** with priorities in locations



Interval Parity Games

An **interval parity game** $\mathcal{G}$ is played on an **LHS** with priorities in locations



Outcome: $p(\pi) = \infty$ since lowest priority on the cycle is 0

Model Checking Games for $Q\mu$

Model Checking Game $MC[\mathcal{K}, \varphi]$

Positions: (ψ, s), ψ is a subformula of φ, and s location of $\mathcal{K}$, or $(-\infty)$,(∞)

Eloïse moves:

$$(\psi \vee \varphi, s) \begin{array}{l} \nearrow (\psi, s) \\ \searrow (\varphi, s) \end{array}$$

$$(\Diamond\varphi, s) \longrightarrow \begin{array}{c} (\varphi, t), t \in sE \\ (-\infty), sE = \emptyset \end{array}$$

$$(\mu X.\varphi, s) \longrightarrow (\varphi, s)$$

$$(X, s) \longrightarrow (\varphi, s)$$

Model Checking Games for Qμ

Abélard moves:

$$(\psi \wedge \varphi, s) \longrightarrow (\psi, s) \quad\text{or}\quad (\psi \wedge \varphi, s) \longrightarrow (\varphi, s)$$

$$(\Box\varphi, s) \longrightarrow \begin{cases} (\varphi, t), t \in sE \\ (\infty), sE = \emptyset \end{cases}$$

$$(\nu X.\varphi, s) \longrightarrow (\varphi, s)$$

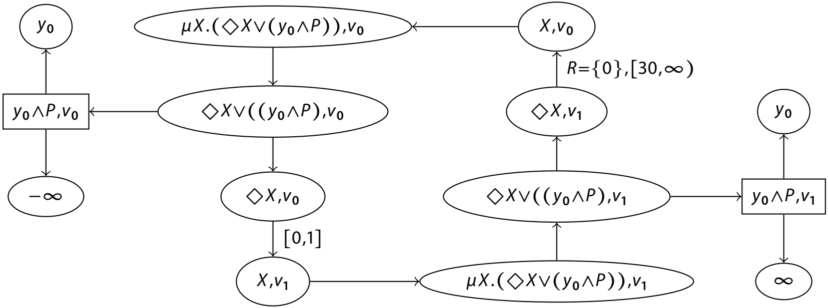
$$(X, s) \longrightarrow (\varphi, s)$$

Terminal Positions: $(P_i, s), (y_j, s), (-\infty)$, and (∞) with payoff accordingly

Priorities: $\Omega(X, s)$ is even if X is a ν-variable, $\Omega(X, s)$ is odd otherwise; priority chosen according to alternation level of X, all other positions get alternation depth of φ.

Example Model Checking Game

$\mathrm{MC}[\mathcal{Q},\varphi]$ for example $\mathcal{Q}$ and $\varphi = \mu X.(\Diamond X \vee (y_0 \wedge P))$

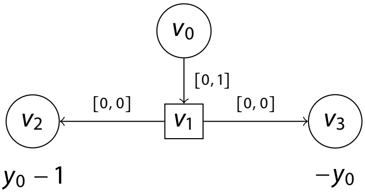


Using Interval Parity Games

Previous Result

For every formula φ in Qμ, LHS $\mathcal{K}$, and $v \in \mathcal{K}$,
the value of MC$[\mathcal{Q}, \varphi]$ from (φ, v) equals $[\![\varphi]\!]^{\mathcal{K}}(v)$.

Problem with Discretisation (we approximate)



After approximation and discretisation: Counter Parity Games
(only increments and resets), which we solve later.

Overview

Structure Rewriting Systems

- Structure Transition Systems and Games
- Separated Rules and Decidability
- Simulation-Based Playing
- Continuous Dynamics

Quantitative Logics

- Standard μ-Calculus
- Quantitative μ-Calculus
- Model Checking on Linear Hybrid Systems
- Model Checking on Increasing Tree Rewriting Systems

Summary

Increasing Tree-Rewriting Rules

Increasing tree-rewriting rules have of the form

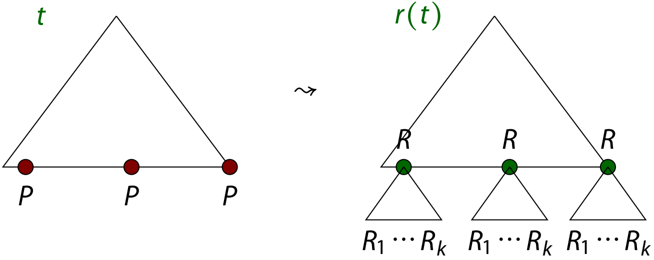
$$r = \quad P \quad \leftarrow \begin{array}{ccc} & R & \\ \swarrow & & \searrow \\ R_1 & \cdots & R_k \end{array} \qquad \text{or} \qquad r = \quad P \leftarrow R,$$

Increasing Tree-Rewriting Rules

Increasing tree-rewriting rules have of the form

$$r = \quad P \quad \leftarrow \quad \begin{array}{ccc} & R & \\ \swarrow & & \searrow \\ R_1 & \cdots & R_k \end{array} \qquad \text{or} \qquad r = \quad P \ \leftarrow \ R,$$

Applying the rule r to a tree t replaces **every** P-leaf.



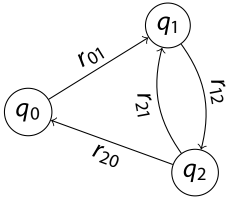
Increasing Tree-Rewriting Systems

$\mathcal{R}$ – a set of increasing tree-rewriting rules

Definition

An **increasing tree-rewriting system (ITRS)** consists of

- a finite set Q of states
- a labelled edge relation $E \subseteq Q \times \mathcal{R} \times Q$



Increasing Tree-Rewriting Systems

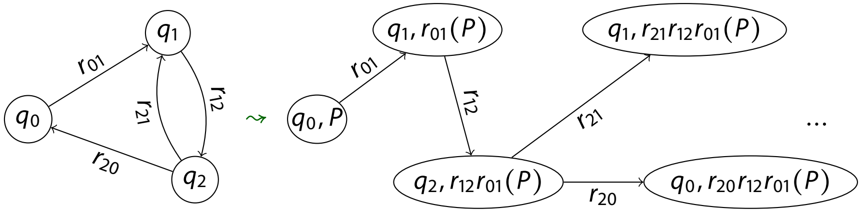
$\mathcal{R}$ – a set of increasing tree-rewriting rules

Definition

An **increasing tree-rewriting system (ITRS)** consists of

- a finite set Q of states
- a labelled edge relation $E \subseteq Q \times \mathcal{R} \times Q$

An ITRS induces a **tree-labelled transition system**



The quantitative μ-calculus for ITRS

Definition (MSO counting term)

An MSO **counting term** has the form $\#_{\overline{x}}\varphi(\overline{x})$, where $\overline{x} = \mathrm{free}(\varphi)$.

$$[\![\#_{\overline{x}}\varphi(\overline{x})]\!]^{\mathfrak{A}} = |\{\overline{a} \mid \mathfrak{A} \models \varphi(\overline{a})\}|$$

The quantitative μ-calculus for ITRS

Definition (MSO **counting term**)

An MSO **counting term** has the form $\#_{\overline{x}}\varphi(\overline{x})$, where $\overline{x} = \mathrm{free}(\varphi)$.

$$[\![\#_{\overline{x}}\varphi(\overline{x})]\!]^{\mathfrak{A}} = |\{\overline{a} \mid \mathfrak{A} \models \varphi(\overline{a})\}|$$

Definition ($Q\mu[\#\mathrm{MSO}]$)

As $Q\mu$ but with MSO **counting terms** as quantitative predicates.

$$\psi ::= \#_{\overline{x}}\varphi(\overline{x}) \mid X \mid \neg\psi \mid \psi \wedge \psi \mid \psi \vee \psi \mid \Box\,\psi \mid \Diamond\,\psi \mid \mu X.\psi \mid \nu X.\psi$$

Evaluated on the induced tree-labelled transition systems.

Examples

- $\mu X.\, \#_x P_a(x) \vee \Diamond X$

$$
\begin{tikzcd}
\mathsf{a} \arrow[r] \arrow[d] & \mathsf{ab} \arrow[r] \arrow[d] & \mathsf{abb} \arrow[r] \arrow[d] & \cdots \\
\mathsf{aa} & \mathsf{aba} \arrow[d] & \mathsf{abba} \arrow[d] & \\
 & \mathsf{abaa} & \mathsf{abbaa} \arrow[d] & \\
 & & \mathsf{abbaaa} &
\end{tikzcd}
$$

Examples

- $\mu X.\ \#_x P_a(x) \vee \Diamond X$

Step 1

$$
\begin{tikzcd}
{}^{1}\,a \arrow[r] \arrow[d, "2"'] & {}^{1}\,ab \arrow[r] \arrow[d, "2"'] & {}^{1}\,abb \arrow[r] \arrow[d, "2"'] & \cdots \\
aa & aba \arrow[d, "3"'] & abba \arrow[d, "3"'] & \\
 & abaa & abbaa \arrow[d, "4"'] & \\
 & & abbaaa &
\end{tikzcd}
$$

Examples

- $\mu X.\, \#_x P_a(x) \vee \Diamond X$

Step 2

$$
\begin{tikzcd}
\overset{2}{\text{a}} \arrow[r] \arrow[d] & \overset{2}{\text{ab}} \arrow[r] \arrow[d] & \overset{2}{\text{abb}} \arrow[r] \arrow[d] & \cdots \\
\overset{2}{\text{aa}} & \overset{3}{\text{aba}} \arrow[d] & \overset{3}{\text{abba}} \arrow[d] & \\
 & \overset{3}{\text{abaa}} & \overset{4}{\text{abbaa}} \arrow[d] & \\
 & & \overset{4}{\text{abbaaa}} &
\end{tikzcd}
$$

Examples

- $\mu X.\, \#_x P_a(x) \vee \Diamond X$

Step 3

$$
\begin{tikzcd}
{}^{2}\,\mathrm{a} \arrow[r] \arrow[d] & {}^{3}\,\mathrm{ab} \arrow[r] \arrow[d] & {}^{3}\,\mathrm{abb} \arrow[r] \arrow[d] & \cdots \\
{}^{2}\,\mathrm{aa} & {}^{3}\,\mathrm{aba} \arrow[d] & {}^{4}\,\mathrm{abba} \arrow[d] & \\
& {}^{3}\,\mathrm{abaa} & {}^{4}\,\mathrm{abbaa} \arrow[d] & \\
& & {}^{4}\,\mathrm{abbaaa} &
\end{tikzcd}
$$

Examples

- $\mu X.\, \#_x P_a(x) \vee \Diamond X$

Step 4

$$
\begin{tikzcd}
{}^{3}\,\mathrm{a} \arrow[r] \arrow[d] & {}^{3}\,\mathrm{ab} \arrow[r] \arrow[d] & {}^{4}\,\mathrm{abb} \arrow[r] \arrow[d] & \cdots \\
{}^{2}\,\mathrm{aa} & {}^{3}\,\mathrm{aba} \arrow[d] & {}^{4}\,\mathrm{abba} \arrow[d] & \\
 & {}^{3}\,\mathrm{abaa} & {}^{4}\,\mathrm{abbaa} \arrow[d] & \\
 & & {}^{4}\,\mathrm{abbaaa} &
\end{tikzcd}
$$

Examples

- $\mu X.\, \#_x P_a(x) \vee \Diamond X$

Step 5

$$
\begin{tikzcd}
\overset{3}{\mathrm{a}} \arrow[r] \arrow[d] & \overset{4}{\mathrm{ab}} \arrow[r] \arrow[d] & \overset{5}{\mathrm{abb}} \arrow[r] \arrow[d] & \cdots \\
\overset{2}{\mathrm{aa}} & \overset{3}{\mathrm{aba}} \arrow[d] & \overset{4}{\mathrm{abba}} \arrow[d] & \\
 & \overset{3}{\mathrm{abaa}} & \overset{4}{\mathrm{abbaa}} \arrow[d] & \\
 & & \overset{4}{\mathrm{abbaaa}} &
\end{tikzcd}
$$

Examples

- $\mu X.\, \#_x P_a(x) \vee \Diamond X$

Step 6

$$
\begin{tikzcd}
\overset{4}{a} \arrow[r] \arrow[d] & \overset{5}{ab} \arrow[r] \arrow[d] & \overset{6}{abb} \arrow[r] \arrow[d] & \cdots \\
\overset{2}{aa} & \overset{3}{aba} \arrow[d] & \overset{4}{abba} \arrow[d] & \\
 & \overset{3}{abaa} & \overset{4}{abbaa} \arrow[d] & \\
 & & \overset{4}{abbaaa} &
\end{tikzcd}
$$

Examples

- $\mu X.\, \#_x P_a(x) \vee \Diamond X$

Step ω

$$\begin{tikzcd}
\infty & \infty & \infty & \\
\mathrm{a} \arrow[r] \arrow[d, "2"'] & \mathrm{ab} \arrow[r] \arrow[d, "3"'] & \mathrm{abb} \arrow[r] \arrow[d, "4"'] & \cdots \\
\mathrm{aa} & \mathrm{aba} \arrow[d, "3"'] & \mathrm{abba} \arrow[d, "4"'] & \\
& \mathrm{abaa} & \mathrm{abbaa} \arrow[d, "4"'] & \\
& & \mathrm{abbaaa} &
\end{tikzcd}$$

Examples

- $\mu X.\, \#_x P_a(x) \vee \Diamond X$
  Maximal number of as seen on any path
- $\mu X.\, \#_x\, (P_a(x) \wedge \exists y\, (y < x \wedge P_b(x))) \vee \Diamond X$
  Maximal number of as after a b seen on any path
- $\nu X.\, \#_x P_a(x) \wedge \Box X$
  Minimal number of as seen on every path

Main Result

Theorem

Let $\psi \in \mathrm{Q}\mu[\#\mathrm{MSO}]$ *and* $\mathcal{T}$ *be an ITRS. Then* $[\![\psi]\!]^{\mathcal{T}}$ *can be computed.*

Main Result

Theorem

Let $\psi \in \mathrm{Q}\mu[\#\mathrm{MSO}]$ and $\mathcal{T}$ be an ITRS. Then $[\![\psi]\!]^{\mathcal{T}}$ can be computed.

Proof steps:

- Use model-checking games $\rightsquigarrow$ **quantitative parity games**
  Problem: infinite arena as trees have unbounded size
- Introduce counters to evaluate counting terms
  $\rightsquigarrow$ **counter parity games** with a finite arena
- Solve counter parity games

Techniques: MSO counting terms decomposition, counter parity games

Counters for ITRS

Idea:

- In general, the size of the increasing **trees** is unbounded
- Instead of the **trees** store the **number** of $\overline{x}$ satisfying counting terms
- Update these numbers on application of tree-rewriting rules

Counters for ITRS

Idea:

- In general, the size of the increasing **trees** is unbounded
- Instead of the **trees** store the **number** of $\overline{x}$ satisfying counting terms
- Update these numbers on application of tree-rewriting rules

Problems:

- A subset of the variables might be assigned to a future tree
- Whether an assignment is valid at present might depend on the future

Counters for ITRS

Idea:

- In general, the size of the increasing **trees** is unbounded
- Instead of the **trees** store the **number** of $\overline{x}$ satisfying counting terms
- Update these numbers on application of tree-rewriting rules

Problems:

- A subset of the variables might be assigned to a future tree
- Whether an assignment is valid at present might depend on the future

  Example

  Let $\varphi = \#_x\Big(P_a(x) \wedge \forall y\big(x < y \rightarrow \neg P_b(y)\big)\Big)$.

  The counter for ba is 1 but after adding a b it needs to be reset.

Decomposition

Theorem

For $t = $
$$\begin{tikzcd}[column sep=small] & R \arrow[dl] \arrow[dr] & \\ t_1 & \cdots & t_k \end{tikzcd}$$
, given only k and $\varphi(\overline{x}) \in \mathrm{MSO}$, one can compute

$$\Phi = \{(\varphi_0(\overline{x}_0), \cdots, \varphi_k(\overline{x}_k)) \mid (\overline{x}_0, \cdots, \overline{x}_k) \textbf{\textit{ partition of }} \overline{x}\}$$

such that $\mathrm{qr}(\varphi_i) \leq \mathrm{qr}(\varphi)$ *and*

$$t \vDash \varphi(\overline{a}) \iff \begin{tikzcd}[column sep=small] & R \vDash \varphi_0(\overline{a}^0) \arrow[dl] \arrow[dr] & \\ t_1 \vDash \varphi_1(\overline{a}^1) & \cdots & t_k \vDash \varphi_k(\overline{a}^k) \end{tikzcd} \quad \textbf{\textit{for (exactly) one }} \overline{\varphi} \in \Phi.$$

Decomposition of Counting Terms

Theorem

For $t = \begin{array}{ccc} & Q & \\ \swarrow & & \searrow \\ t_1 & \cdots & t_k \end{array}$ *, given only* k*, a counting term* $\#_{\overline{x}}\varphi(\overline{x})$*, and a partition* $[\overline{x}]_l = (\overline{x}_0, \cdots, \overline{x}_l)$ *of* $\overline{x}$*, one can compute*

$$\Psi_{[\overline{x}]_l} = \left\{\left(\tau_0(\overline{x}_0), \cdots, \tau_l(\overline{x}_l)\right) \mid (\overline{x}_0, \cdots, \overline{x}_l) = [\overline{x}]_l\right\}$$

such that $\mathrm{qr}(\tau_i) \leq \mathrm{qr}(\varphi)$ *and*

$$[\![\#_{\overline{x}}\varphi(\overline{x})]\!]^t = \sum_{[\overline{x}]_l \in \{[\overline{x}]_l\}} \sum_{\overline{\tau} \in \Psi_{[\overline{x}]_l}} [\![\#_{\overline{x}_0}\tau_0(\overline{x}_0)]\!]^{t_Q} \cdot [\![\#_{\overline{x}_1}\tau_1(\overline{x}_1)]\!]^{t_1} \cdot \ldots \cdot [\![\#_{\overline{x}_l}\tau_l(\overline{x}_l)]\!]^{t_l}.$$

Counting Terms on ITRS

A run of an ITRS has the form:

$$
\begin{tikzcd}[column sep=small]
 & & & Q \arrow[dl] \arrow[dr] & & & & & & Q \arrow[ddll] \arrow[ddrr] & & \\
Q_0 & \xrightarrow{r_0} & Q_1 & \cdots & Q_l & \cdots & \xrightarrow{r_n} & & & \vdots & & \\
 & & & & & & & Q'_1 & & \cdots & & Q'_m
\end{tikzcd}
$$

Counting Terms on ITRS

A run of an ITRS has the form:

$$Q_0 \xrightarrow{r_0} \begin{array}{c} Q \\ \swarrow \quad \searrow \\ Q_1 \cdots Q_l \end{array} \quad \cdots \quad \xrightarrow{r_n} \begin{array}{c} Q \\ \swarrow \quad \vdots \quad \searrow \\ Q'_1 \quad \cdots \quad Q'_m \end{array}$$

- After a single step decomposition of $\#_{\overline{x}}\varphi(\overline{x})$:

$$\sum (\tau_0, Q) \cdot \ \big(\tau_1, r^n \cdots r^1(Q_1)\big) \cdot \cdots \cdot \big(\tau_l, r^n \cdots r^1(Q_l)\big)$$

Counting Terms on ITRS

A run of an ITRS has the form:

$$
\begin{tikzcd}[column sep=small]
 & & & Q \arrow[dl] \arrow[dr] & & & & & & Q \arrow[dll] \arrow[drr] & & \\
Q_0 & \xrightarrow{r_0} & Q_1 & \cdots & Q_l & \cdots & \xrightarrow{r_n} & & & \vdots & & \\
 & & & & & & & Q'_1 & & \cdots & & Q'_m
\end{tikzcd}
$$

- Next step:

$$
\sum \; \big(\tau_1, r^n \cdots r^2(Q_1)\big) \cdot \cdots \cdot \big(\tau_l, r^n \cdots r^2(Q_l)\big)
$$

$$
\sum \big(\tau'_1, r^n \cdots r^2(Q^1_1)\big) \cdot \cdots \cdot \big(\tau'_k, r^n \cdots r^2(Q^1_k)\big)
$$

Counting Terms on ITRS

A run of an ITRS has the form:

$$Q_0 \xrightarrow{r_0} \begin{array}{c} Q \\ \swarrow \quad \searrow \\ Q_1 \cdots Q_l \end{array} \quad \cdots \quad \xrightarrow{r_n} \begin{array}{c} Q \\ \swarrow \quad \vdots \quad \searrow \\ Q'_1 \quad \cdots \quad Q'_m \end{array}$$

- Next step:

$$\sum \; \big(\tau_1, r^n \cdots r^2(Q_1)\big) \cdot \cdots \cdot \big(\tau_l, r^n \cdots r^2(Q_l)\big)$$

- Products of finite length $\rightsquigarrow$ finitely many

Counting Terms on ITRS

A run of an ITRS has the form:

$$
\begin{tikzcd}[column sep=small]
 & & & Q \arrow[dl] \arrow[dr] & & & & & Q \arrow[ddll] \arrow[ddrr] & & \\
Q_0 & \xrightarrow{r_0} & Q_1 & \cdots & Q_l & \cdots & \xrightarrow{r_n} & & \vdots & & \\
 & & & & & & Q'_1 & & \cdots & & Q'_m
\end{tikzcd}
$$

- Next step:

$$\sum \; \big(\tau_1, r^n \cdots r^2(Q_1)\big) \cdot \cdots \cdot \big(\tau_l, r^n \cdots r^2(Q_l)\big)$$

- Products of finite length $\rightsquigarrow$ finitely many
- It suffices to know the number of occurrences of each summand
- Such counters can be updated given a rule r_i

Counting Terms on ITRS

A run of an ITRS has the form:

$$Q_0 \xrightarrow{r_0} \begin{array}{c} Q \\ \swarrow \quad \searrow \\ Q_1 \cdots Q_l \end{array} \quad \cdots \quad \xrightarrow{r_n} \begin{array}{c} Q \\ \swarrow \quad \vdots \quad \searrow \\ Q'_1 \quad \cdots \quad Q'_m \end{array}$$

- Next step:

$$\sum \left(\tau_1, r^n \cdots r^2(Q_1)\right) \cdot \cdots \cdot \left(\tau_l, r^n \cdots r^2(Q_l)\right)$$

- Products of finite length $\rightsquigarrow$ finitely many
- It suffices to know the number of occurrences of each summand
- Such counters can be updated given a rule r_i

$\rightsquigarrow$ reduces MC games to Counter Parity Games

Counter Parity Games

Counter Parity Game (CPG) is a quantitative parity game:

- edges labelled with affine counter-update functions $f : \overline{c} \mapsto A\overline{c} + B$
- terminal vertices labelled by a payoff function $\lambda = \pm c_i$

Players: maximiser (**Maxi**) and minimiser (**Mini**)

Infinite plays payoff: ∞, if parity is satisfied, otherwise $-\infty$

Finite plays payoff: λ at terminal

Counter Parity Games

Counter Parity Game (CPG) is a quantitative parity game:

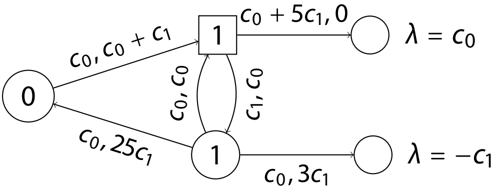
- edges labelled with affine counter-update functions $f\colon \overline{c} \mapsto A\overline{c} + B$
- terminal vertices labelled by a payoff function $\lambda = \pm c_i$

Players: maximiser (**Maxi**) and minimiser (**Mini**)

Infinite plays payoff: ∞, if parity is satisfied, otherwise $-\infty$

Finite plays payoff: λ at terminal

Example



Counter Parity Games

Counter Parity Game (CPG) is a quantitative parity game:

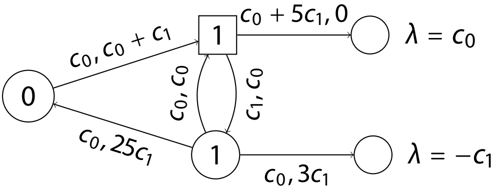
- edges labelled with affine counter-update functions $f\colon \bar{c} \mapsto A\bar{c} + B$
- terminal vertices labelled by a payoff function $\lambda = \pm c_i$

Players: maximiser (**Maxi**) and minimiser (**Mini**)

Infinite plays payoff: ∞, if parity is satisfied, otherwise $-\infty$

Finite plays payoff: λ at terminal

Example



Solution: through games with imperfect recall

Overview

What we Have

Model

- **Structure transition systems** with **rewriting** transitions
- **Logics**: first- and **monadic second-order**
  fixed-points, **counting** and some **temporal** operators
- **Continuous dynamics** by **ODE**s, **merged** for universal rules
- **Games** with payoffs defined by **counting terms**

Theorems

- Decidability of $L_\mu[\mathrm{MSO}]$ on **separated** games
- Approximability of $Q\mu$ on **finite initialised linear** hybrid systems
- Decidability of $Q\mu[\#\mathrm{MSO}]$ on **separated** games

Implementation

- **Model** is implemented with **basic** model checking
- **Games** can be played and **heuristics** are generated
- **Dynamics** is simulated but very **naïve**

What we Wish

What is not there

- More advanced **model checking**
- Continuous dynamics with **better ODE solvers**
- Game playing with good **continuous parameter search**

BIOCHAM …

- Is **discrete rewriting** of any use?
- How about **games instead of logic** sometimes?
- Using BIOCHAM (at least CMA-ES?) for **continuous model-checking**?

What we Wish

What is not there

- More advanced **model checking**
- Continuous dynamics with **better ODE solvers**
- Game playing with good **continuous parameter search**

BIOCHAM …

- Is **discrete rewriting** of any use?
- How about **games instead of logic** sometimes?
- Using BIOCHAM (at least CMA-ES?) for **continuous model-checking**?

Thank You