Playing Games when States have Rich Structure

Łukasz Kaiser

CNRS & LIAFA
Paris

GT Jeux Meeting

Paris, 2010

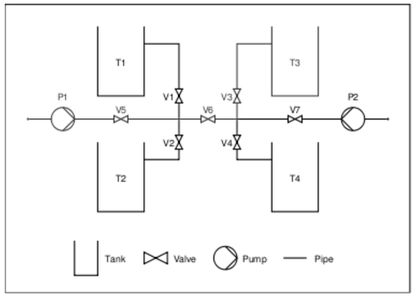
Motivation

AlgoSyn: Algorithmic Synthesis of Reactive and Discrete-Continuous Systems

Motivation

AlgoSyn: Algorithmic Synthesis of Reactive and Discrete-Continuous Systems

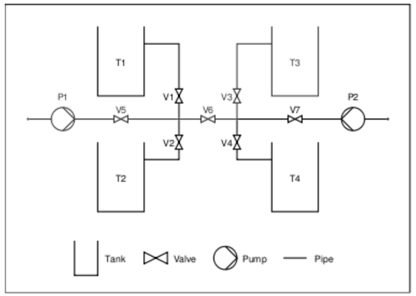
Part of a pumping station



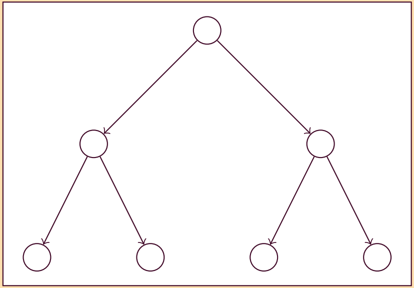
Motivation

AlgoSyn: Algorithmic Synthesis of Reactive and Discrete-Continuous Systems

Part of a pumping station



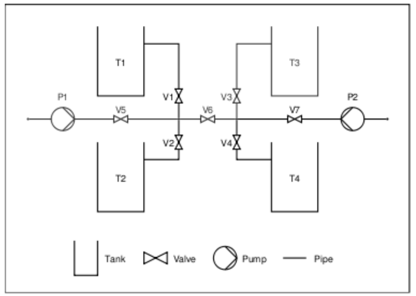
Structures we usually consider



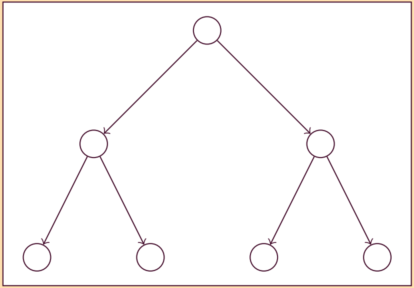
Motivation

AlgoSyn: Algorithmic Synthesis of Reactive and Discrete-Continuous Systems

Part of a pumping station



Structures we usually consider



Can we model states by arbitrary relational structures?

(1) change described using appropriate rewriting rules

(2) properties given in MSO on structures + temporal logic for change

Overview

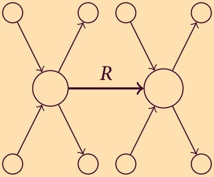
Structure Rewriting

Separated Games

Outlook

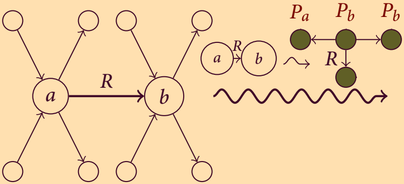
Structure Rewriting Rules

Rewriting Example



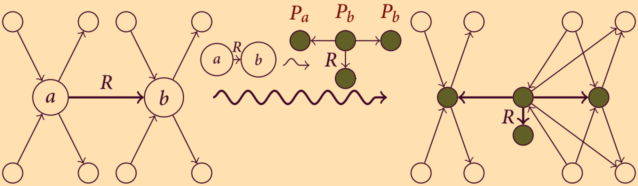
Structure Rewriting Rules

Rewriting Example



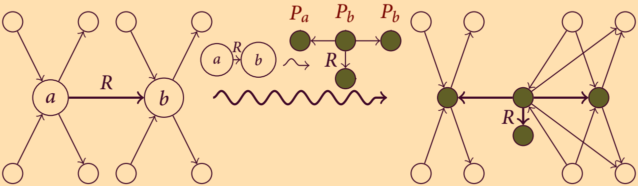
Structure Rewriting Rules

Rewriting Example



Structure Rewriting Rules

Rewriting Example



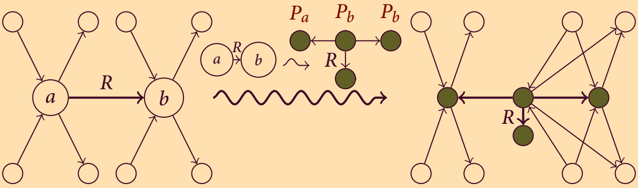
Relational Structures and Embeddings

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

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

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 $l \in L\} \cup \{(a, a) \mid a \in A\}$,

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

(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**: $\mathfrak{A}_{\text{next}} = \mathfrak{A}[\mathfrak{L} \to \mathfrak{R}/\sigma]$, 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 $\mathfrak{R}$

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:

- L_μ (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 $\mathfrak{L} \to \mathfrak{R}$:

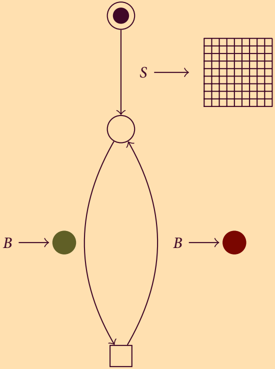
- **Existential**: $\mathfrak{A}_{\text{next}} = \mathfrak{A}[\mathfrak{L} \to \mathfrak{R}/\sigma]$, the player chooses the embedding σ
- **Universal**: $\mathfrak{A}_{\text{next}} = \mathfrak{A}[\mathfrak{L} \to \mathfrak{R}]$, **all** occurrences of $\mathfrak{L}$ are rewritten to $\mathfrak{R}$

Winning conditions:

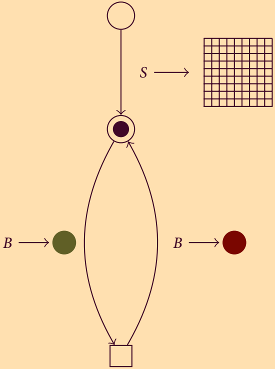
- L_μ (or temporal) formula ψ with **MSO sentences** for predicates, or
- MSO formula φ to be evaluated on the **limit** of the play
  Limit of $\mathfrak{A}_0\mathfrak{A}_1\mathfrak{A}_2 \ldots = \left(\bigcup_{n\in\mathbb{N}} \bigcap_{i\geq n} A_i, \bigcup_{n\in\mathbb{N}} \bigcap_{i\geq n} R^{\mathfrak{A}_i}\right)$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

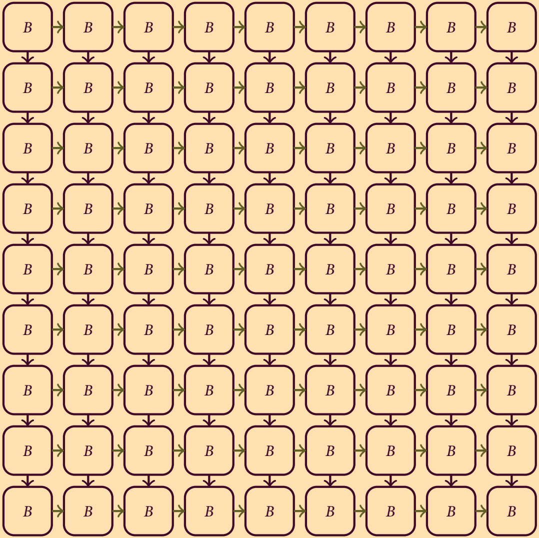
Motivation: many questions are **naturally defined as such games**:
constraint satisfaction, model checking, graph measures, games people play

Example Game: Gomoku (Connect–5)

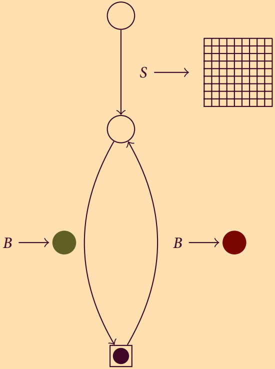


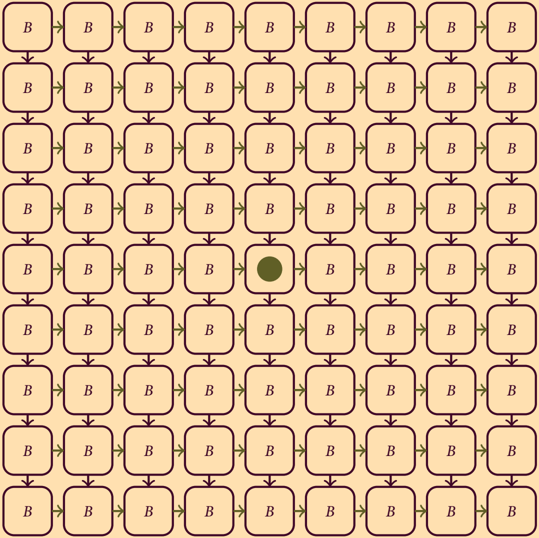
Example Game: Gomoku (Connect–5)



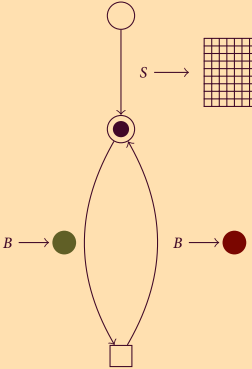


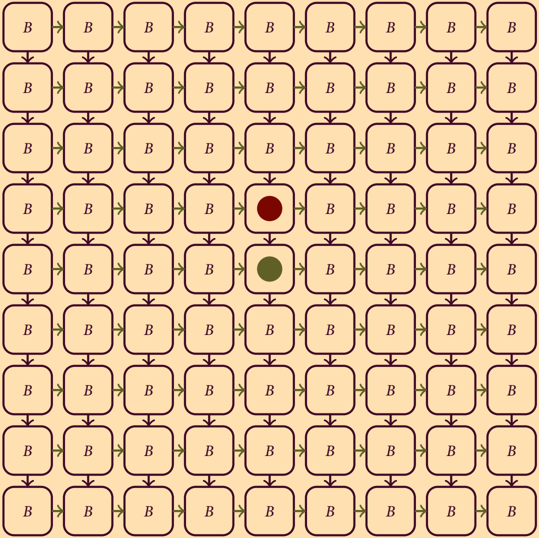
Example Game: Gomoku (Connect–5)



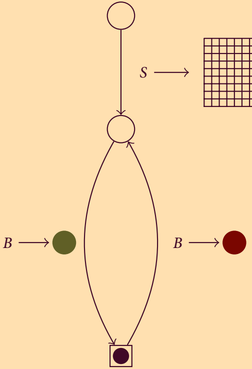


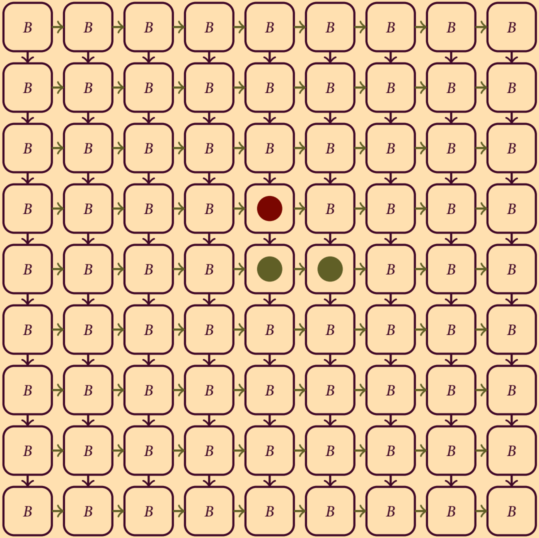
Example Game: Gomoku (Connect–5)



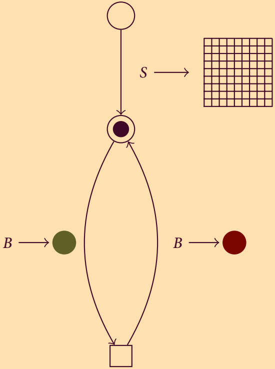


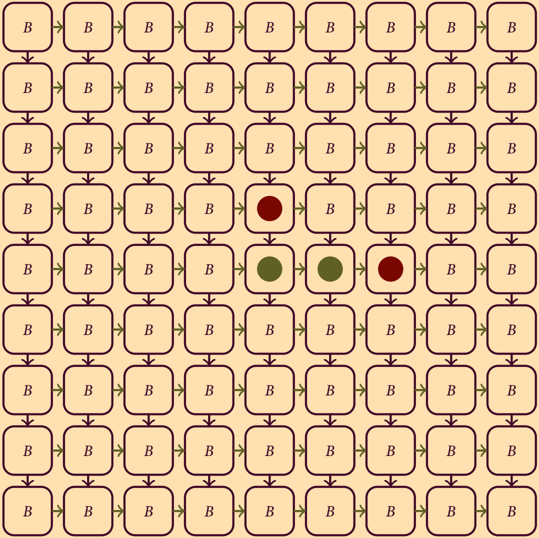
Example Game: Gomoku (Connect–5)



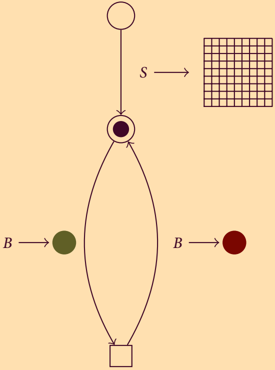


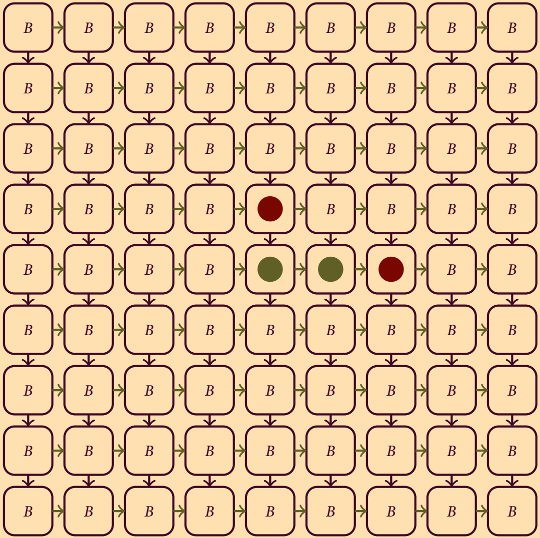
Example Game: Gomoku (Connect–5)





Example Game: Gomoku (Connect–5)





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

Overview

Structure Rewriting

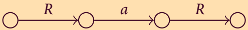
Separated Games

Outlook

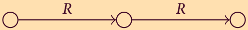
Simple Structure Rewriting

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

Separated:



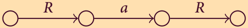
Not Separated:



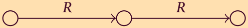
Simple Structure Rewriting

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

Separated:



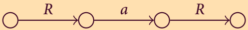
Not Separated:

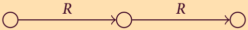


Simple Rule $\mathfrak{L} \to \mathfrak{R}$: $\mathfrak{R}$ is **separated** and $\mathfrak{L}$ is a **single tuple in relation**

Simple Structure Rewriting

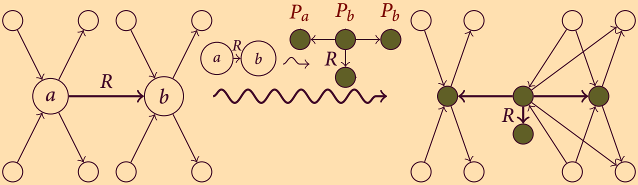
Separated Structures: no element is in two **non-terminal** relations
(Courcelle, Engelfriet, Rozenberg, 1991)

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

- $L_\mu[\mathrm{MSO}]$**:** Temporal properties expressed in L_μ (subsumes LTL) with properties of structures (states) expressed in MSO
- $\lim \mathrm{MSO}$**:** 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 $L_\mu[\mathrm{MSO}]$ or $\lim \mathrm{MSO}$ 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

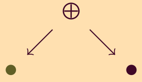
Description of how to build $\mathfrak{A}$ is a tree $\mathcal{T}(\mathfrak{A})$ with:

- **Leafs** of **different colours** $1 \ldots k$
- $\oplus$ 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:

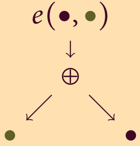
- **Leafs** of **different colours** $1 \ldots k$
- $\oplus$ representing **disjoint sum**
- $i \leftarrow j$ to **change colour** of **all** nodes from i to j
- $e(i, j)$ to **add all pairs** of (i, j)-coloured nodes to e



Proof: Interpreting a Structure in a Tree

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

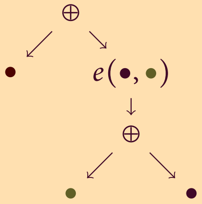
- **Leafs** of **different colours** $1 \ldots k$
- $\oplus$ representing **disjoint sum**
- $i \leftarrow j$ to **change colour** of **all** nodes from i to j
- $e(i, j)$ to **add all pairs** of (i, j)-coloured nodes to e



Proof: Interpreting a Structure in a Tree

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

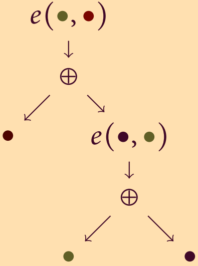
- **Leafs** of **different colours** $1 \dots k$
- $\oplus$ representing **disjoint sum**
- $i \leftarrow j$ to **change colour** of **all** nodes from i to j
- $e(i, j)$ to **add all pairs** of (i, j)-coloured nodes to e



Proof: Interpreting a Structure in a Tree

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

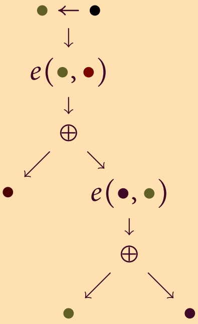
- **Leafs** of **different colours** $1 \ldots k$
- $\oplus$ representing **disjoint sum**
- $i \leftarrow j$ to **change colour** of **all** nodes from i to j
- $e(i, j)$ to **add all pairs** of (i, j)-coloured nodes to e



Proof: Interpreting a Structure in a Tree

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

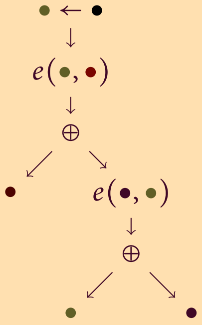
- **Leafs** of **different colours** $1 \dots k$
- $\oplus$ representing **disjoint sum**
- $i \leftarrow j$ to **change colour** of **all** nodes from i to j
- $e(i, j)$ to **add all pairs** of (i, j)-coloured nodes to e



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



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 $\mathcal{I}(\mathcal{T}(\mathfrak{A})) \cong \mathfrak{A}$.

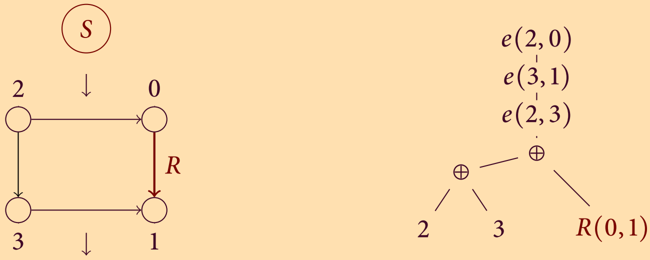
Proof: Simple Rewriting in the Tree

S

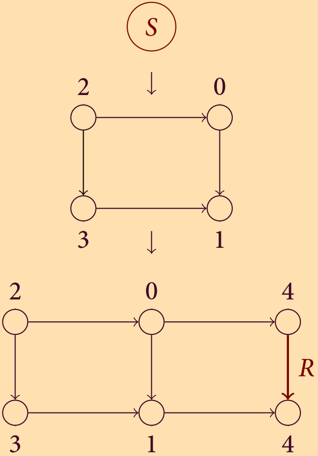
$\downarrow$

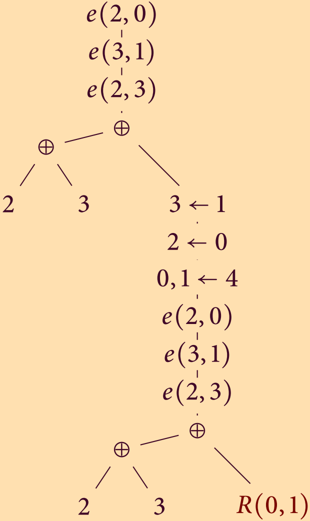
S

Proof: Simple Rewriting in the Tree

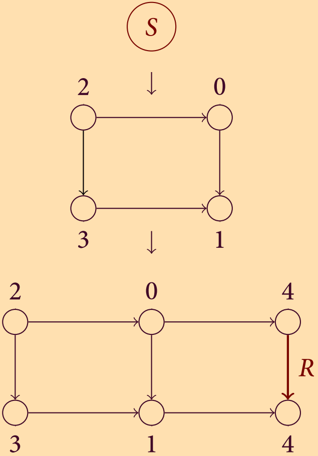


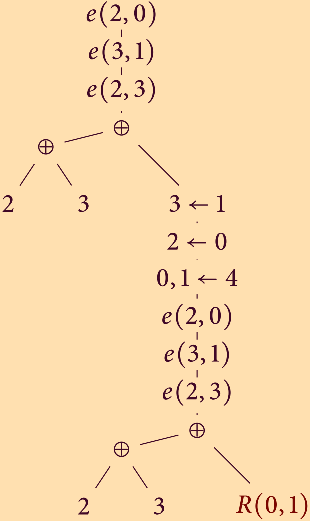
Proof: Simple Rewriting in the Tree





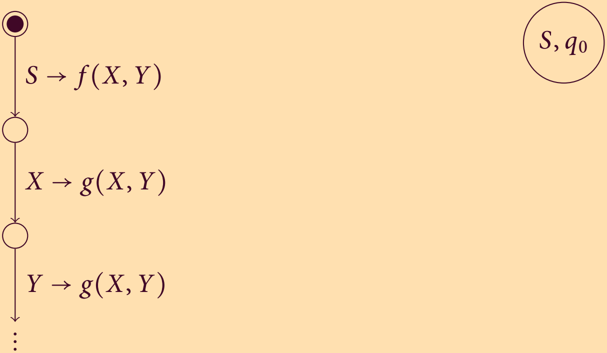
Proof: Simple Rewriting in the Tree



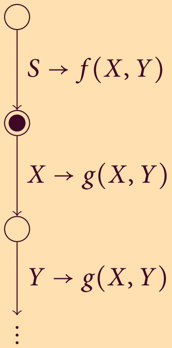


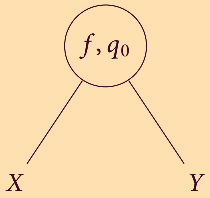
MSO-to-MSO interpretation: $\varphi \to \psi$

Proof: From Tree to Alternating Word Automata

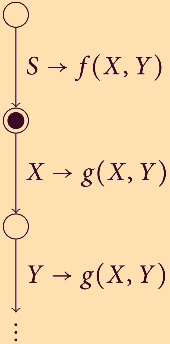


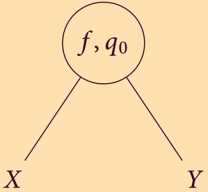
Proof: From Tree to Alternating Word Automata





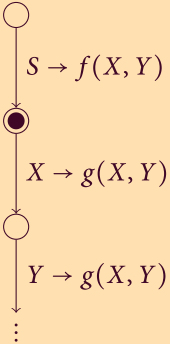
Proof: From Tree to Alternating Word Automata

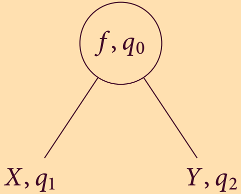




existential: pick transition

Proof: From Tree to Alternating Word Automata

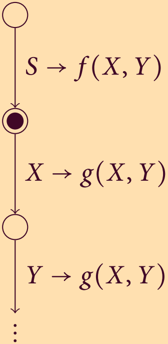


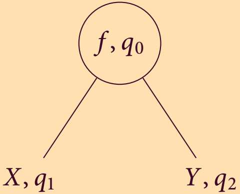


existential: pick transition

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

Proof: From Tree to Alternating Word Automata



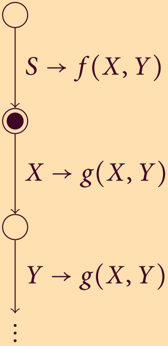


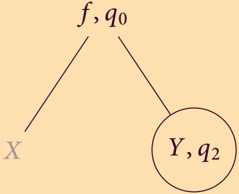
existential: pick transition

universal: left or right

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

Proof: From Tree to Alternating Word Automata



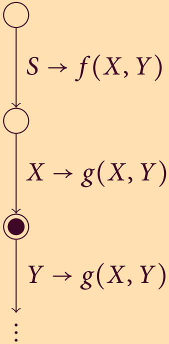


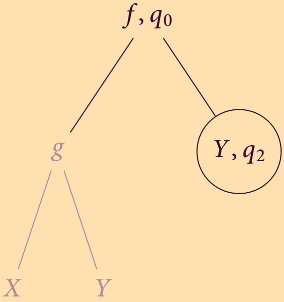
existential: pick transition

universal: left or right

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

Proof: From Tree to Alternating Word Automata





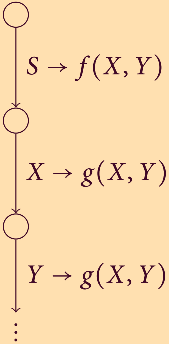
existential: pick transition

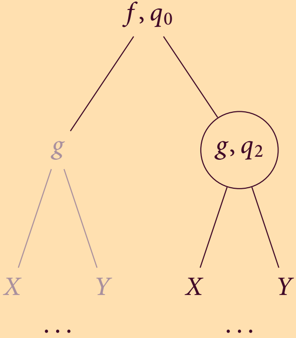
universal: left or right

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

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

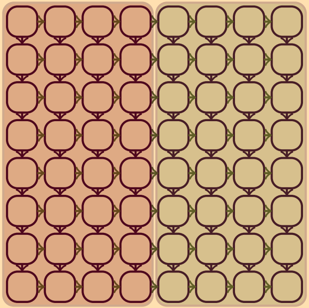
Structure Rewriting

Separated Games

Outlook

Imperfect Information

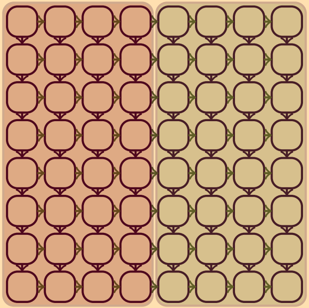
How to Represent Imperfect Information?



- simply allow **three-valued relations**?
- using **observable elements**?
- with **formulas** known to hold?
- methods from **databases with imperfect information**?

Imperfect Information

How to Represent Imperfect Information?



- simply allow **three-valued relations**?
- using **observable elements**?
- with **formulas** known to hold?
- methods from **databases with imperfect information**?

Application: **abstraction** and abstraction refinement for **complex games**.

Strategies and Higher-Order Games

Strategies in Separated Games

- **Winning positions** can be defined in **MSO**
- What is a **simple form of strategy**?

  $\mathfrak{A}_1 \to \mathfrak{A}_2 \to \ldots \to \mathfrak{A}_n \to ?$
- Certain forms can be **derived from the presented proof**

Strategies and Higher-Order Games

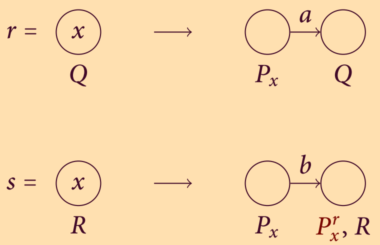
Strategies in Separated Games

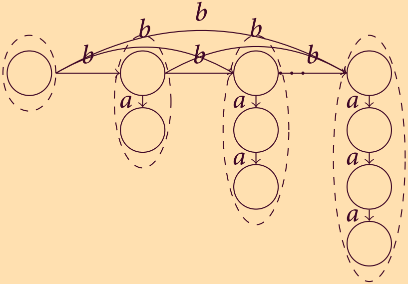
- **Winning positions** can be defined in **MSO**
- What is a **simple form of strategy**?

  $\mathfrak{A}_1 \to \mathfrak{A}_2 \to \ldots \to \mathfrak{A}_n \to ?$

- Certain forms can be **derived from the presented proof**

Higher-Order Structures and Rewriting





Does this correspond to **higher-order pushdown systems** and **strategies**?

Conclusions

Structure Rewriting Games

- **General** model of games with **structured states**
- Establishing the winner is **decidable** for **certain subclasses**

Conclusions

Structure Rewriting Games

- **General** model of games with **structured states**
- Establishing the winner is **decidable** for **certain subclasses**

Extensions

- **Preconditions and postconditions** in rewriting rules
- More complex **kinds of connections** in rules
- **Continuous dynamics** can be added
  - defined e.g. using $\mathbb{R}$-**structures and differential equations**
  - simple **quantitative logics** can be used

Questions

- When do **strategies** of a simple form exist?
- What about games with **imperfect information**?
- How can we define **higher-order** games?

Conclusions

Structure Rewriting Games

- **General** model of games with **structured states**
- Establishing the winner is **decidable** for **certain subclasses**

Extensions

- **Preconditions and postconditions** in rewriting rules
- More complex **kinds of connections** in rules
- **Continuous dynamics** can be added
  - defined e.g. using **$\mathbb{R}$-structures and differential equations**
  - simple **quantitative logics** can be used

Questions

- When do **strategies** of a simple form exist?
- What about games with **imperfect information**?
- How can we define **higher-order** games?

Thank You