Analysis of Hypergraph Rewriting Systems

Łukasz Kaiser

Mathematische Grundlagen der Informatik
RWTH Aachen

Wrocław, 2008

Motivation

Intuitive Models of the World

- **Intuitive** is important, as coding is **costly** and **error-prone**
- **Hypergraphs** are a general model of **discrete** structures
  - studied in **software engineering and design** for a long time
- **Games** are a natural model of **interaction**
  - **Rewriting** can be used as **actions** of players

Motivation

Intuitive Models of the World

- **Intuitive** is important, as coding is **costly** and **error-prone**
- **Hypergraphs** are a general model of **discrete** structures
  - studied in **software engineering and design** for a long time
- **Games** are a natural model of **interaction**
  - **Rewriting** can be used as **actions** of players

Methods for Analysis of Systems

- Theorem proving, Abstraction **(very general, needs guidance)**
- **Termination analysis (guessing induction order) (general)**
- **Regularity and Automata (specific, basis for type systems)**

Overview

Graph Minors

Termination Analysis

Graphs of Bounded Clique-Width

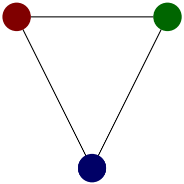
Simple Hypergraph Rewriting Games

Definition of Graph Minors

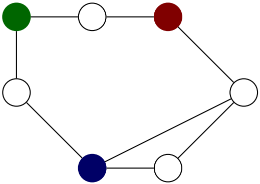
$G \preccurlyeq H$ if G can be obtained from H by

- removing edges
- contracting edges
- deleting singular vertices

Example



$\preccurlyeq$

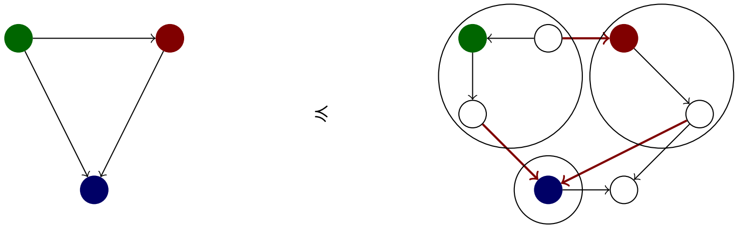


Definition of Directed Hypergraph Minors

$G \preccurlyeq H$ if there is a collapse of H to G

- map **vertices** of G to **connected graphs of vertices** of H
  - different vertices of G mapped to **disjoint** sets in H
  - any two connected vertices in the result **incident** to a hyperedge of H
- find **hyperedges** of G as **hyperedges** of H
  - connect on the i**th** position of the edge **some** vertex in the i**th set**
  - **strong:** use **other hyperedges** than the ones for incidence above

Example



Wagner Conjecture

Theorem (Seymour-Robertson; Graphs: 2004, Hypergraphs: to appear)

*Minor ordering is a **well-quasi-order:** in every infinite sequence $G_0, G_1, \ldots$ of finite graphs there exist $i < j$ such that $G_i \preccurlyeq G_j$.*

Corollary

*Every **upwards closed** set has a **finite basis.***
*Every **downwards closed** set admits a **finite obstruction set.***

Consequences

- **Kruskal's tree theorem**
- **Kuratowski's planar graphs theorem (weak form)**

Algorithmic Results

Theorem (Seymour-Robertson)

Fix G. The algorithmic problem: given H is $G \preccurlyeq H$ is in $\mathsf{TIME}\big(O(|H|^3)\big)$.

Consequences

- Every problem **downwards closed under minors** is in $O(n^3)$
- Checking **planarity** is in $O(n^3)$
- For every k, checking if **Entanglement** $G = k$ is in $O(n^3)$
  (For **undirected graphs** G)

Overview

Graph Minors

Termination Analysis

Graphs of Bounded Clique-Width

Simple Hypergraph Rewriting Games

From Term to Graph Rewriting

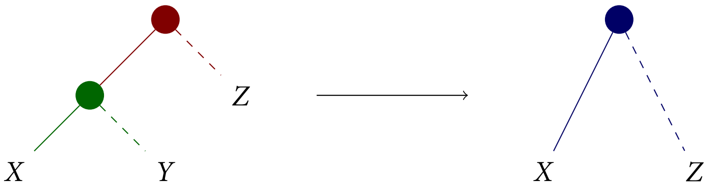
Set of rules $l \to r$

Operation

- Find $l\sigma$
- Remove it
- Insert $r\sigma$

From Term to Graph Rewriting

Set of rules $\hat{l} \rightarrow \hat{r}$



Operation

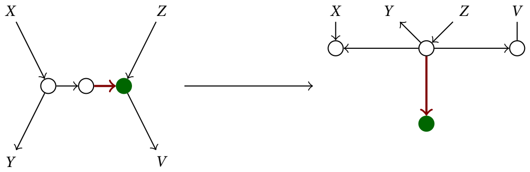
- Find a tree isomorphic to l **without variables**
- Remove it
- Insert r and **reconnect**

Notes:

- Observe **edge labels** denoting both **symbol** and **arity**
- Defined only for **left-linear (perhaps better this way)**
- Tight correspondence only for **right-linear (graphs: constant memory)**

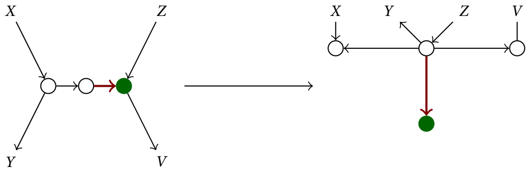
Hypergraph Rewriting

Rule

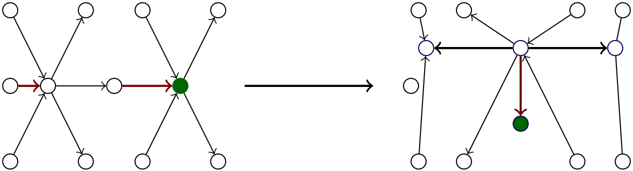


Hypergraph Rewriting

Rule



Application



Note: e.g. *Y* matched **all** black successors **(position 1 in black hyperedge)**

Term Rewriting Termination Overview

Embedding and Simplification Orders

- Let $t <_{\text{emb}} s$ if the corresponding graphs $\hat{t} \preccurlyeq \hat{s}$ **(topological embedding)**
- An ordering $<$ is a **simplification order** if it **contains** $<_{\text{emb}}$ and is **well-behaved** under contexts and substitutions
- **Kruskal's tree theorem:** simplification orders are well-quasi-orders
- $<$ is a simplification order and $r < l$ for **each rule** $\rightsquigarrow$ the system **terminates**

Term Rewriting Termination Overview

Embedding and Simplification Orders

- Let $t <_{\text{emb}} s$ if the corresponding graphs $\hat{t} \preccurlyeq \hat{s}$ **(topological embedding)**
- An ordering $<$ is a **simplification order** if it **contains** $<_{\text{emb}}$ and is **well-behaved** under contexts and substitutions
- **Kruskal's tree theorem:** simplification orders are well-quasi-orders
- $<$ is a simplification order and $r < l$ for **each rule** $\rightsquigarrow$ the system **terminates**

Classical simplification orders

- Path orderings: **LPO, RPO, RPOS (with status)**
- Knuth-Bendix ordering
- Polynomial orders

Term Rewriting Termination Overview

Embedding and Simplification Orders

- Let $t <_{\text{emb}} s$ if the corresponding graphs $\hat{t} \preccurlyeq \hat{s}$ **(topological embedding)**
- An ordering $<$ is a **simplification order** if it **contains** $<_{\text{emb}}$ and is **well-behaved** under contexts and substitutions
- **Kruskal's tree theorem:** simplification orders are well-quasi-orders
- $<$ is a simplification order and $r < l$ for **each rule** $\rightsquigarrow$ the system **terminates**

Classical simplification orders

- Path orderings: **LPO, RPO, RPOS (with status)**
- Knuth-Bendix ordering
- Polynomial orders

Problem

$$f(f(x)) \to f(g(f(x)))$$

Help: dependency pairs, abstraction

Hypergraph Rewriting Termination?

The basis given by Graph Minor Theorem!

Hypergraph Rewriting Termination?

The basis given by Graph Minor Theorem!

Problems

- **Not clear** whether $r \preccurlyeq l$ is enough
- But there are **some sufficient conditions**
  - Assume that the **collapse sets of variables are singletons**
  - More abstract conditions by **Barbara König** (2008, single pushout)
- Does **changing directions** or **labels** spoil anything?

Hypergraph Rewriting Termination?

The basis given by Graph Minor Theorem!

Problems

- **Not clear** whether $r \preccurlyeq l$ is enough
- But there are **some sufficient conditions**
  - Assume that the **collapse sets of variables are singletons**
  - More abstract conditions by **Barbara König** (2008, single pushout)
- Does **changing directions** or **labels** spoil anything?

Possible Work

- Clarify at least some of the **problems above**
- Is there something similar to **simplification orders**?
- Are there notions **analogous to LPO, RPO**?
- Can **other approaches** be used?

Overview

Graph Minors

Termination Analysis

Graphs of Bounded Clique-Width

Simple Hypergraph Rewriting Games

Let's Build a Graph

Pieces to build a graph:

- Bags of **single nodes** with **different colors** $1 \dots K$
- Paint to **change color** of **all** nodes from i to j
- Edges to **connect** **all** nodes of color i to **all** of color j

Example:

Let's Build a Graph

Pieces to build a graph:

- Bags of **single nodes** with **different colors** $1 \dots K$
- Paint to **change color** of **all** nodes from i to j
- Edges to **connect** **all** nodes of color i to **all** of color j

Example:

Let's Build a Graph

Pieces to build a graph:

- Bags of **single nodes** with **different colors** $1 \dots K$
- Paint to **change color** of **all** nodes from i to j
- Edges to **connect** **all** nodes of color i to **all** of color j

Example:

Let's Build a Graph

Pieces to build a graph:

- Bags of **single nodes** with **different colors** $1 \dots K$
- Paint to **change color** of **all** nodes from i to j
- Edges to **connect all** nodes of color i to **all** of color j

Example:

Let's Build a Graph

Pieces to build a graph:

- Bags of **single nodes** with **different colors** $1 \ldots K$
- Paint to **change color** of **all** nodes from i to j
- Edges to **connect** **all** nodes of color i to **all** of color j

Example:

Let's Build a Graph

Pieces to build a graph:

- Bags of **single nodes** with **different colors** $1 \dots K$
- Paint to **change color** of **all** nodes from i to j
- Edges to **connect** **all** nodes of color i to **all** of color j

Example:

Let's Build a Graph

Pieces to build a graph:

- Bags of **single nodes** with **different colors** $1 \dots K$
- Paint to **change color** of **all** nodes from i to j
- Edges to **connect all** nodes of color i to **all** of color j

Example:

Let's Build a Graph

Pieces to build a graph:

- Bags of **single nodes** with **different colors** $1 \dots K$
- Paint to **change color** of **all** nodes from i to j
- Edges to **connect** **all** nodes of color i to **all** of color j

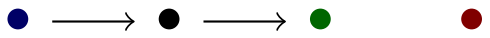
Example:

Let's Build a Graph

Pieces to build a graph:

- Bags of **single nodes** with **different colors** $1 \ldots K$
- Paint to **change color** of **all** nodes from i to j
- Edges to **connect** **all** nodes of color i to **all** of color j

Example:



Let's Build a Graph

Pieces to build a graph:

- Bags of **single nodes** with **different colors** $1 \dots K$
- Paint to **change color** of **all** nodes from i to j
- Edges to **connect** **all** nodes of color i to **all** of color j

Example:

$$\bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet$$

Let's Build a Graph

Pieces to build a graph:

- Bags of **single nodes** with **different colors** $1 \dots K$
- Paint to **change color** of **all** nodes from i to j
- Edges to **connect** **all** nodes of color i to **all** of color j

Example:

$$\bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \cdots \longrightarrow \bullet$$

Let's Build a Graph

Pieces to build a graph:

- Bags of **single nodes** with **different colors** $1 \ldots K$
- Paint to **change color** of **all** nodes from i to j
- Edges to **connect** **all** nodes of color i to **all** of color j

Example:

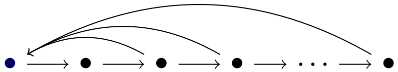
$$\bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \cdots \longrightarrow \bullet$$

Let's Build a Graph

Pieces to build a graph:

- Bags of **single nodes** with **different colors** $1 \dots K$
- Paint to **change color** of **all** nodes from i to j
- Edges to **connect all** nodes of color i to **all** of color j

Example:



Interpreting a Graph in a Tree

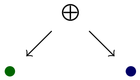
Description of how to build $\mathcal{G}$ is a tree $\mathcal{T}(\mathcal{G})$:

Interpreting a Graph in a Tree

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

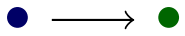
Interpreting a Graph in a Tree

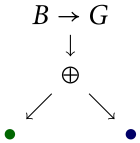
Description of how to build $\mathcal{G}$ is a tree $\mathcal{T}(\mathcal{G})$:



Interpreting a Graph in a Tree

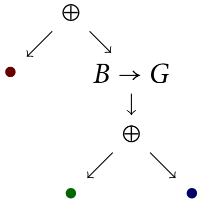
Description of how to build $\mathcal{G}$ is a tree $\mathcal{T}(\mathcal{G})$:





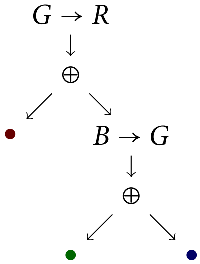
Interpreting a Graph in a Tree

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



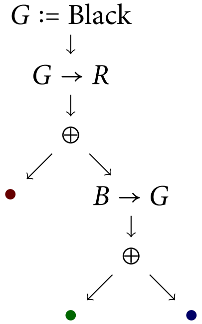
Interpreting a Graph in a Tree

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



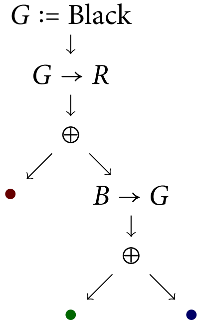
Interpreting a Graph in a Tree

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



Interpreting a Graph in a Tree

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



Theorem:
For every K there is an MSO-to-MSO **interpretation** $\mathcal{I}$ such that for all graphs $\mathcal{G}$ of **clique-width** $\leq K$ holds

$$\mathcal{I}(\mathcal{T}(\mathcal{G})) \cong \mathcal{G}$$

Bounded Clique-Width Graphs

Corollary:
For every K and $\varphi \in \mathsf{MSO}(\text{Graphs})$ there exists $\psi \in \mathsf{MSO}(\text{Tree})$ such that

$$\textbf{Clique-Width}(K) \models \varphi \iff \textbf{Binary Tree} \models \psi$$

Bounded Clique-Width Graphs

Corollary:
For every K and $\varphi \in \mathsf{MSO}(\text{Graphs})$ there exists $\psi \in \mathsf{MSO}(\text{Tree})$ such that

$$\textbf{Clique-Width}(K) \vDash \varphi \iff \textbf{Binary Tree} \vDash \psi$$

Examples:

- singly or doubly-linked lists, with back-links
- nested lists (lists of lists), trees
- all graphs of bounded tree-width
- cliques, full bipartite graphs

Bounded Clique-Width Graphs

Corollary:
For every K and $\varphi \in \mathsf{MSO}(\text{Graphs})$ there exists $\psi \in \mathsf{MSO}(\text{Tree})$ such that

$$\textbf{Clique-Width}(K) \vDash \varphi \iff \textbf{Binary Tree} \vDash \psi$$

Examples:

- singly or doubly-linked lists, with back-links
- nested lists (lists of lists), trees
- all graphs of bounded tree-width
- cliques, full bipartite graphs

Characterizations:

- **All** families of graphs uniformly MSO-interpretable in the binary tree.
- Configurations of **pushdown automata** (mod ε-transitions)
- Graphs obtained by **simple rewriting** (later)

Bounded Clique-Width Graphs

Corollary:
For every K and $\varphi \in \mathsf{MSO}(\text{Graphs})$ there exists $\psi \in \mathsf{MSO}(\text{Tree})$ such that

$$\textbf{Clique-Width}(K) \vDash \varphi \iff \textbf{Binary Tree} \vDash \psi$$

Examples:

- singly or doubly-linked lists, with back-links
- nested lists (lists of lists), trees
- all graphs of bounded tree-width
- cliques, full bipartite graphs

Characterizations:

- **All** families of graphs uniformly MSO-interpretable in the binary tree.
- Configurations of **pushdown automata** (mod ε-transitions)
- Graphs obtained by **simple rewriting** (later)

Applications: e.g. verification of **heap-manipulating programs**

Overview

Graph Minors

Termination Analysis

Graphs of Bounded Clique-Width

Simple Hypergraph Rewriting Games

Simple Hypergraph Rewriting

Simple Tree Rewriting (**ground** and left-hand side **is a constant**)

$$\mathsf{List} \to \mathsf{cons}(o, \mathsf{List})$$

Note: **simple** due to deep **connection to automata** and **decidability results**

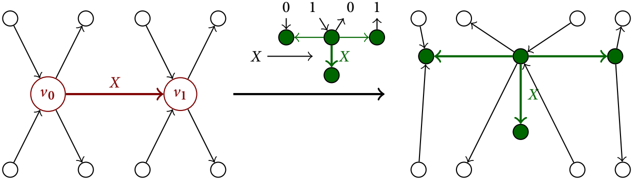
Simple Hypergraph Rewriting

Simple Tree Rewriting (**ground** and left-hand side **is a constant**)

$$\mathsf{List} \to \mathsf{cons}(o, \mathsf{List})$$

Note: **simple** due to deep **connection to automata** and **decidability results**

Simple Hypergraph Rewriting **(Courcelle, Engelfriet, Rozenberg, 1991)**



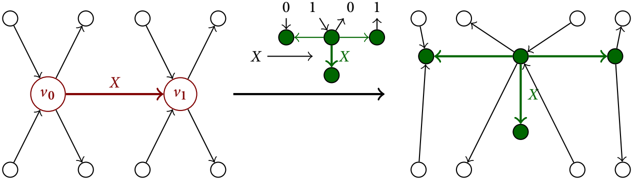
Simple Hypergraph Rewriting

Simple Tree Rewriting (**ground** and left-hand side **is a constant**)

$$\mathsf{List} \to \mathsf{cons}(o, \mathsf{List})$$

Note: **simple** due to deep **connection to automata** and **decidability results**

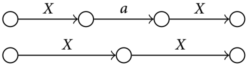
Simple Hypergraph Rewriting **(Courcelle, Engelfriet, Rozenberg, 1991)**



Separated Hypergraphs: no vertex is incident to two non-terminal edges

Separated:

Not Separated:



Games Played with Hypergraphs

Definition

- Fix a **finite set of separated handle rewriting rules** $\mathbb{S}$
- **Game:** directed graph
  - **vertices** assigned to **players**
  - **edges** labelled by **rules from** $\mathbb{S}$
- **Play:** construction of a **sequence of hypergraphs**
- **Winning condition:** defined in MSO over the **limit hypergraph**

Games Played with Hypergraphs

Definition

- Fix a **finite set of separated handle rewriting rules** $\mathbb{S}$
- **Game:** directed graph
  - **vertices** assigned to **players**
  - **edges** labelled by **rules from** $\mathbb{S}$
- **Play:** construction of a **sequence of hypergraphs**
- **Winning condition:** defined in MSO over the **limit hypergraph**

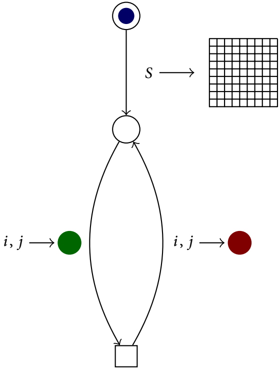
Rewriting Sequences and Limit Hypergraphs

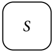
- $G[X \to H]$ is G with **all*** occurrences of X rewritten to H
- **Limit** of $G_0 \to G_1 \to G_3 \to \ldots$: $(\bigcup_{n\in\mathbb{N}} \bigcap_{i\geq n} V_i, \bigcup_{n\in\mathbb{N}} \bigcap_{i\geq n} E_i)$

***Notes:**

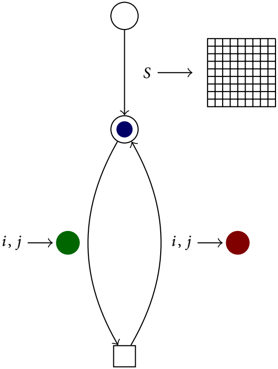
- Rewriting separated graphs is **confluent**
- If players **pick** positions: **undecidable**, see **Active context-free games**, thanks to **Anca Muscholl**

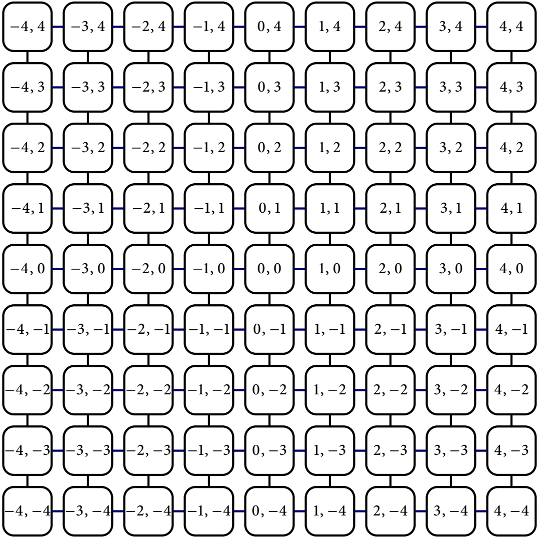
Example: Playing Gomoku



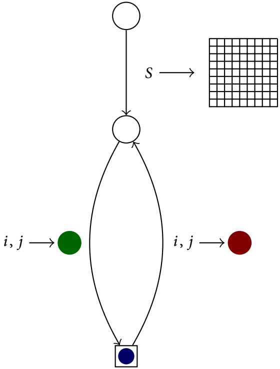


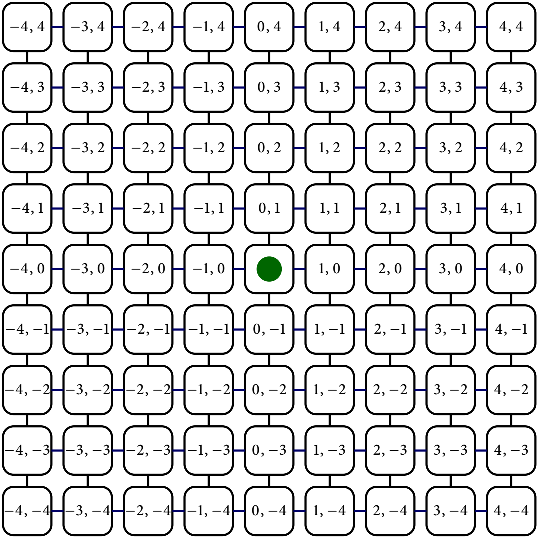
Example: Playing Gomoku



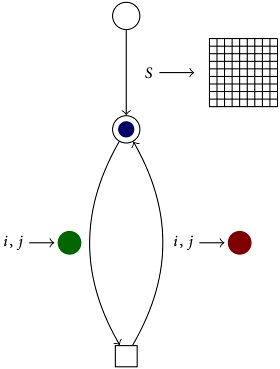


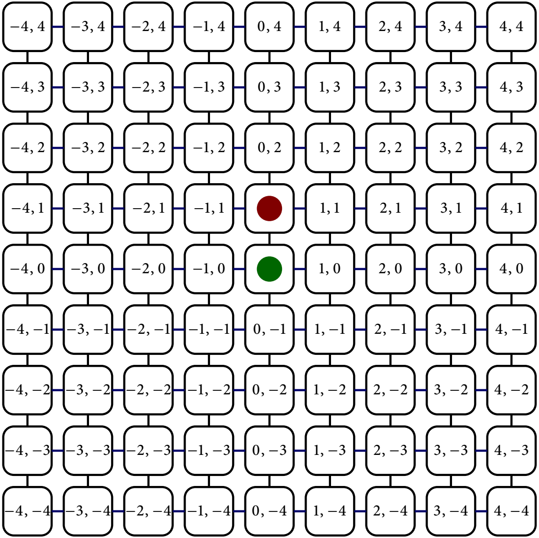
Example: Playing Gomoku



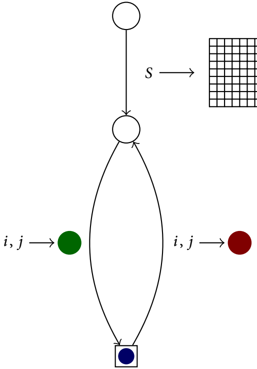


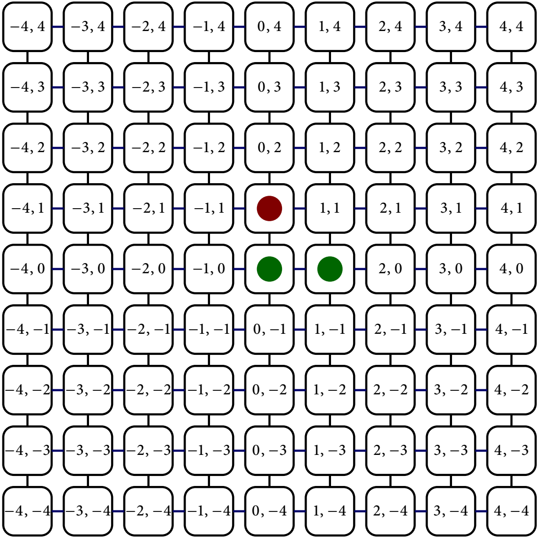
Example: Playing Gomoku



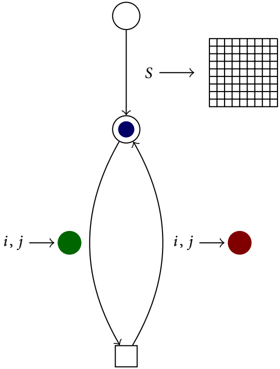


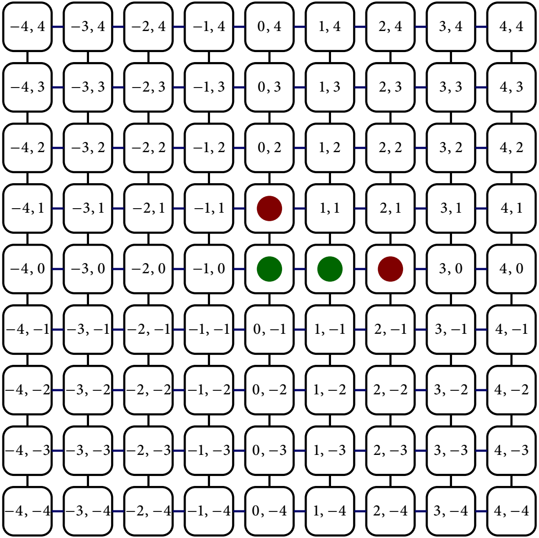
Example: Playing Gomoku



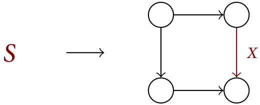


Example: Playing Gomoku

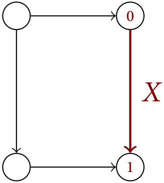


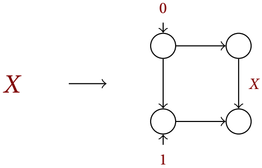


Example: Constructing a Ladder

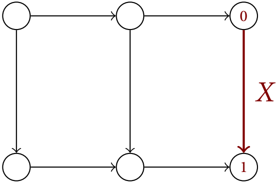


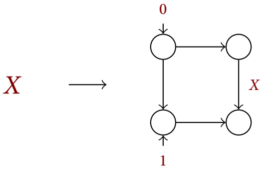
Example: Constructing a Ladder



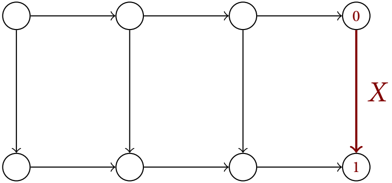


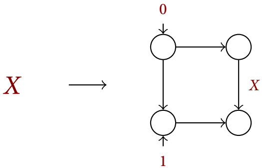
Example: Constructing a Ladder



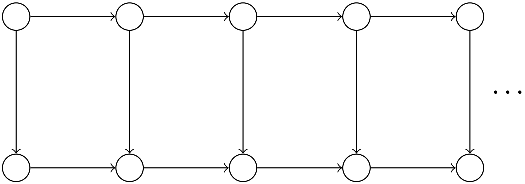


Example: Constructing a Ladder





Example: Constructing a Ladder



Result on Games

Theorem

- *Let $\mathbb{S}$ be a **finite** set of **separated handle rewriting rules***
- *and φ be an MSO formula (giving the **winning condition**)*

*Then the set $\{\pi \in \mathbb{S}^\omega \;:\; \lim G(\pi) \vDash \varphi\}$ is **ω-regular**.*

Corollary

Establishing the winner of finite separated handle rewriting games is decidable.

Result on Games

Theorem

- *Let $\mathbb{S}$ be a **finite** set of **separated handle rewriting rules***
- *and φ be an MSO formula (giving the **winning condition**)*

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

Corollary

Establishing the winner of finite separated handle rewriting games is decidable.

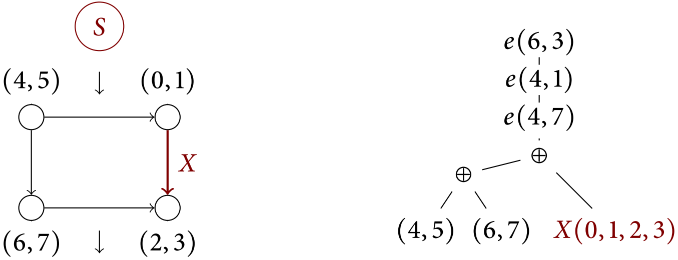
Other Consequences

- Strategies only require **finite memory**
- Decidability and determinacy for **concurrent stochastic arenas**
- In multiplayer games **rational (iteratively weakly dominant) strategies are computable**

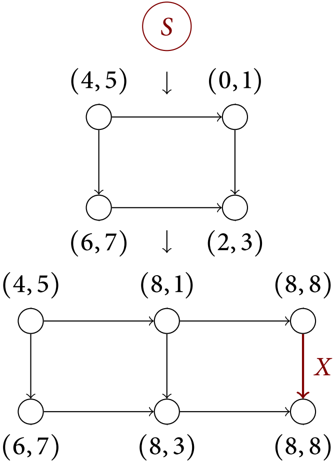
Proof: Separated Graph Rewriting as a Tree

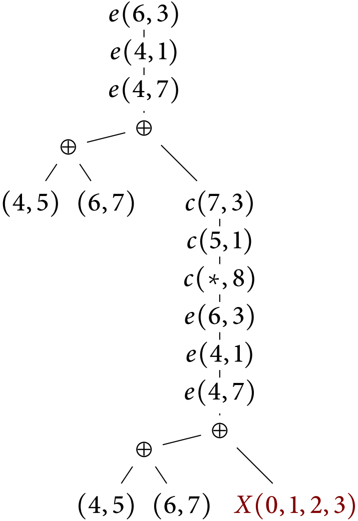
S

Proof: Separated Graph Rewriting as a Tree

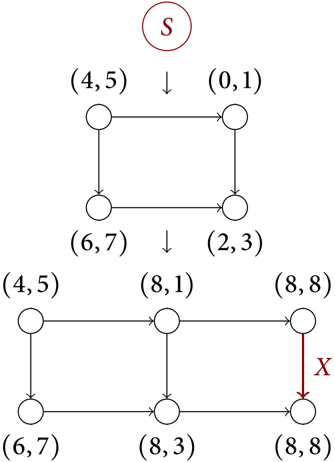


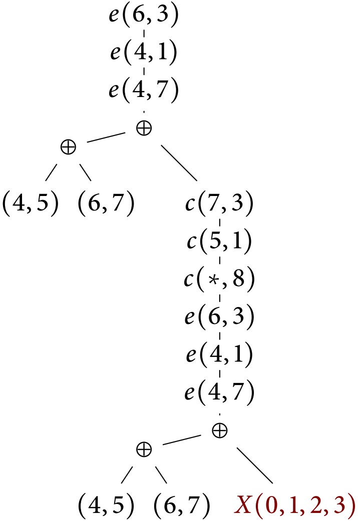
Proof: Separated Graph Rewriting as a Tree





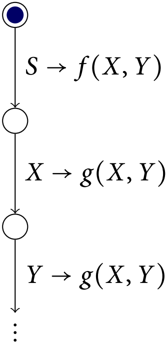
Proof: Separated Graph Rewriting as a Tree



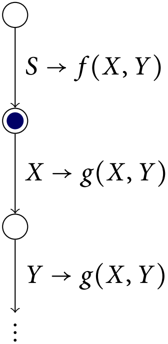


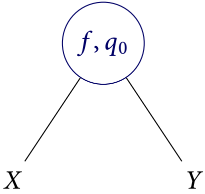
MSO-to-MSO **interpretation:** $\varphi \to \psi$

Proof: From Tree to Alternating Word Automata

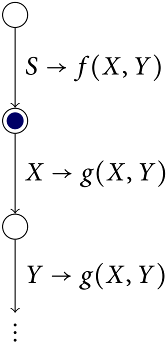


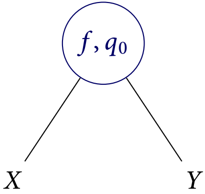
Proof: From Tree to Alternating Word Automata





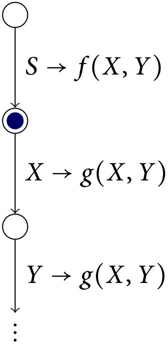
Proof: From Tree to Alternating Word Automata

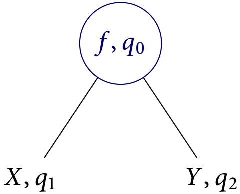




existential: pick transition

Proof: From Tree to Alternating Word Automata

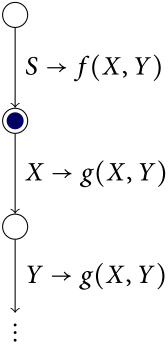


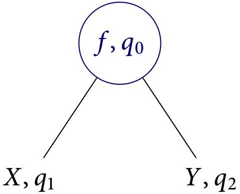


existential: pick transition

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

Proof: From Tree to Alternating Word Automata



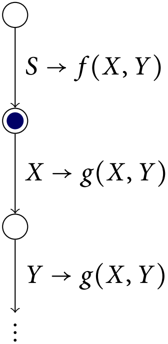


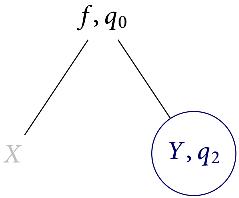
existential: pick transition

universal: left or right

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

Proof: From Tree to Alternating Word Automata



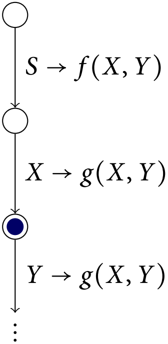


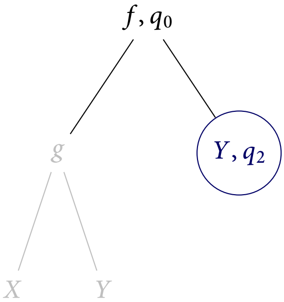
existential: pick transition

universal: left or right

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

Proof: From Tree to Alternating Word Automata





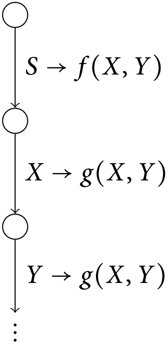
existential: pick transition

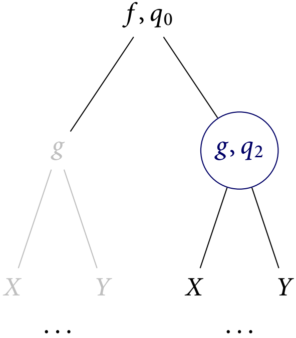
universal: left or right

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

ignore

Proof: From Tree to Alternating Word Automata





existential: pick transition

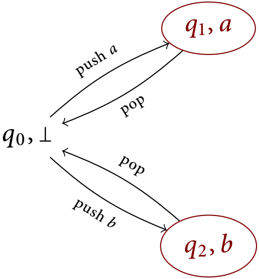
universal: left or right

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

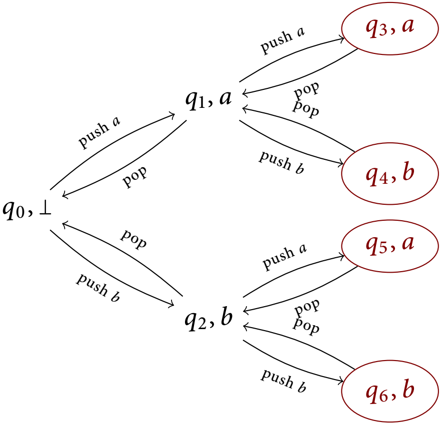
ignore

Application: Pushdown Parity Games

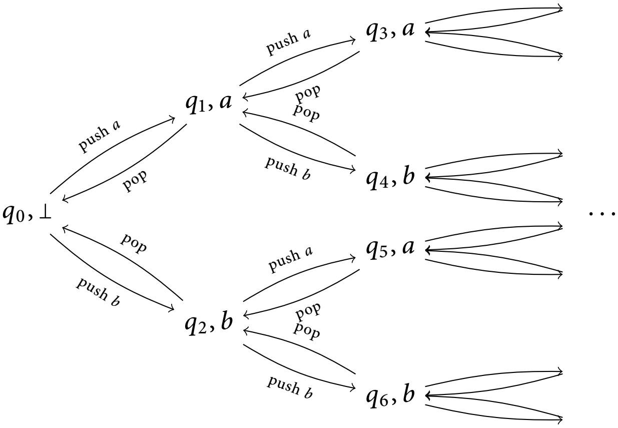
Application: Pushdown Parity Games



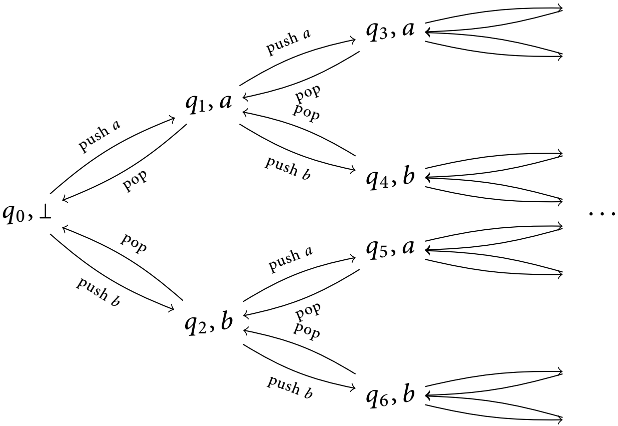
Application: Pushdown Parity Games



Application: Pushdown Parity Games

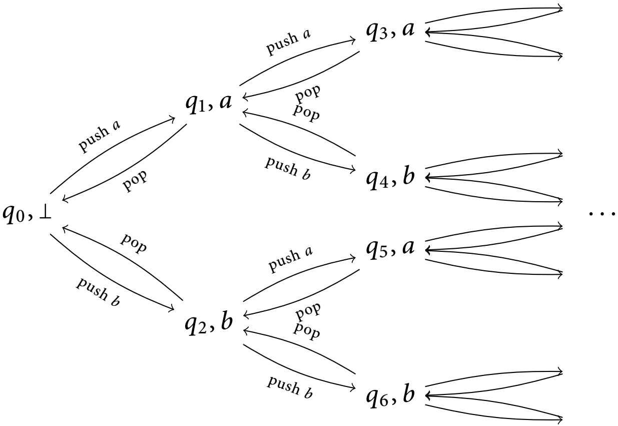


Application: Pushdown Parity Games



We can **define** the **parity winning condition** over states in MSO, **or**

Application: Pushdown Parity Games



We can **define** the **parity winning condition** over states in MSO, **or**

Play a Game on the Resulting Graph

(Higher-order Hypergraph Rewriting Games)

Application of Higher-order Games

Problem: $\exists x \; \forall y \; R(x, y)$ with an automaton ($\rightsquigarrow$ MSO formula) recognizing R

Application of Higher-order Games

Problem: $\exists x \, \forall y \, R(x, y)$ with an automaton ($\rightsquigarrow$ MSO formula) recognizing R

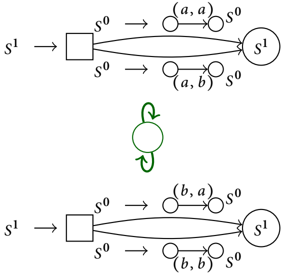
Solution: build the word $x_0 \otimes y_0$, **the Verifier picks x_0, the Falsifier y_0.**

Application of Higher-order Games

Problem: $\exists x\ \forall y\ R(x, y)$ with an automaton ($\rightsquigarrow$ MSO formula) recognizing R

Solution: build the word $x_0 \otimes y_0$, **the Verifier picks x_0, the Falsifier y_0.**

Higher-order game

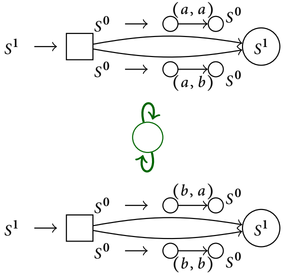


Application of Higher-order Games

Problem: $\exists x\ \forall y\ R(x, y)$ with an automaton ($\rightsquigarrow$ MSO formula) recognizing R

Solution: build the word $x_0 \otimes y_0$, **the Verifier picks x_0, the Falsifier y_0.**

Higher-order game



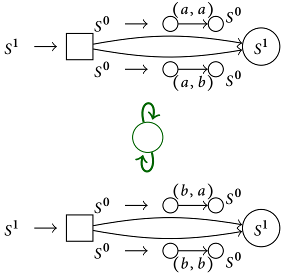
Constructed game

Application of Higher-order Games

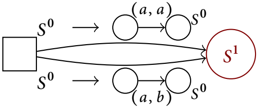
Problem: $\exists x\ \forall y\ R(x, y)$ with an automaton ($\rightsquigarrow$ MSO formula) recognizing R

Solution: build the word $x_0 \otimes y_0$, **the Verifier picks x_0, the Falsifier y_0.**

Higher-order game



Constructed game

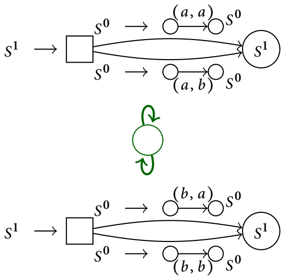


Application of Higher-order Games

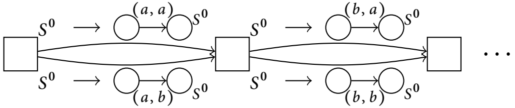
Problem: $\exists x\ \forall y\ R(x, y)$ with an automaton ($\rightsquigarrow$ MSO formula) recognizing R

Solution: build the word $x_0 \otimes y_0$, **the Verifier picks x_0, the Falsifier y_0.**

Higher-order game



Constructed game

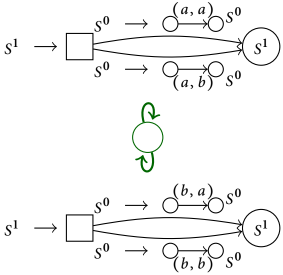


Application of Higher-order Games

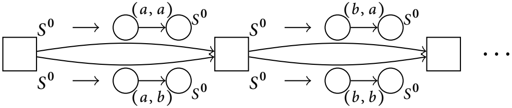
Problem: $\exists x \; \forall y \; R(x,y)$ with an automaton ($\rightsquigarrow$ MSO formula) recognizing R

Solution: build the word $x_0 \otimes y_0$, **the Verifier picks x_0**, **the Falsifier y_0.**

Higher-order game



Constructed game



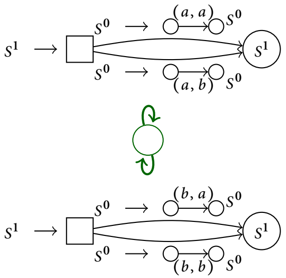
Constructed hypergraph

Application of Higher-order Games

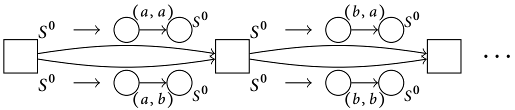
Problem: $\exists x\ \forall y\ R(x, y)$ with an automaton ($\rightsquigarrow$ MSO formula) recognizing R

Solution: build the word $x_0 \otimes y_0$, **the Verifier picks x_0**, **the Falsifier y_0**.

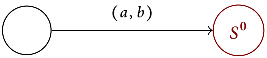
Higher-order game



Constructed game



Constructed hypergraph

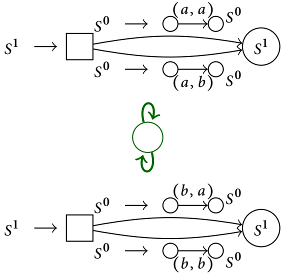


Application of Higher-order Games

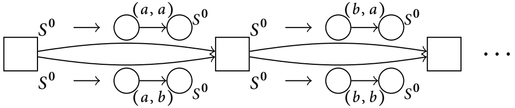
Problem: $\exists x\ \forall y\ R(x, y)$ with an automaton ($\rightsquigarrow$ MSO formula) recognizing R

Solution: build the word $x_0 \otimes y_0$, **the Verifier picks x_0**, **the Falsifier y_0.**

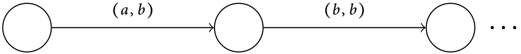
Higher-order game



Constructed game



Constructed hypergraph



Summary

Hypergraph Rewriting Games are Fun!

Definition Problems

- Hypergraph rewriting **without pushouts?**
- Can we really **use variables?**
- Why all non-terminals are separated? What if **only corresponding ones?**

Existing Tools

- **Minor ordering** which is a **well-quasi-order**
- **Vertex replacement construction**
- MSO-to-MSO **interpretations** and **automata**
- Standard **ω-regular games**

Summary

Hypergraph Rewriting Games are Fun!

Definition Problems

- Hypergraph rewriting **without pushouts?**
- Can we really **use variables?**
- Why all non-terminals are separated? What if **only corresponding ones?**

Existing Tools

- **Minor ordering** which is a **well-quasi-order**
- **Vertex replacement construction**
- MSO-to-MSO **interpretations** and **automata**
- Standard **ω-regular games**

Thank You