Games Played with Hypergraphs

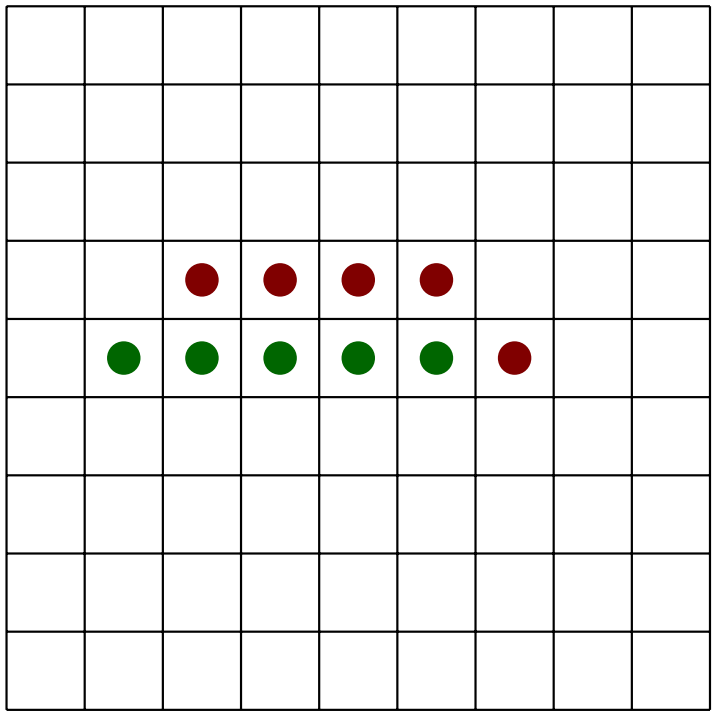
Łukasz Kaiser

Mathematische Grundlagen der Informatik
RWTH Aachen

Games 2008, Warsaw

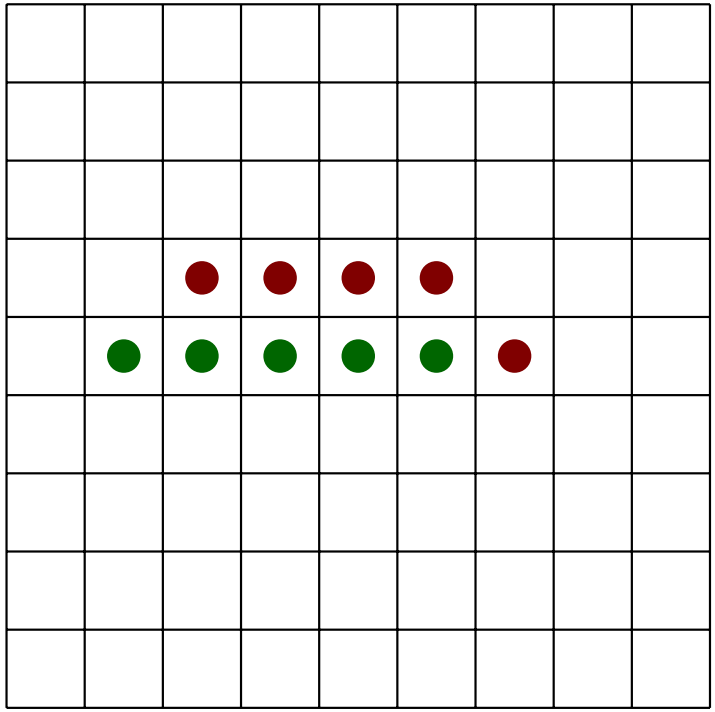
How People Play Games

Gomoku (five in a row)

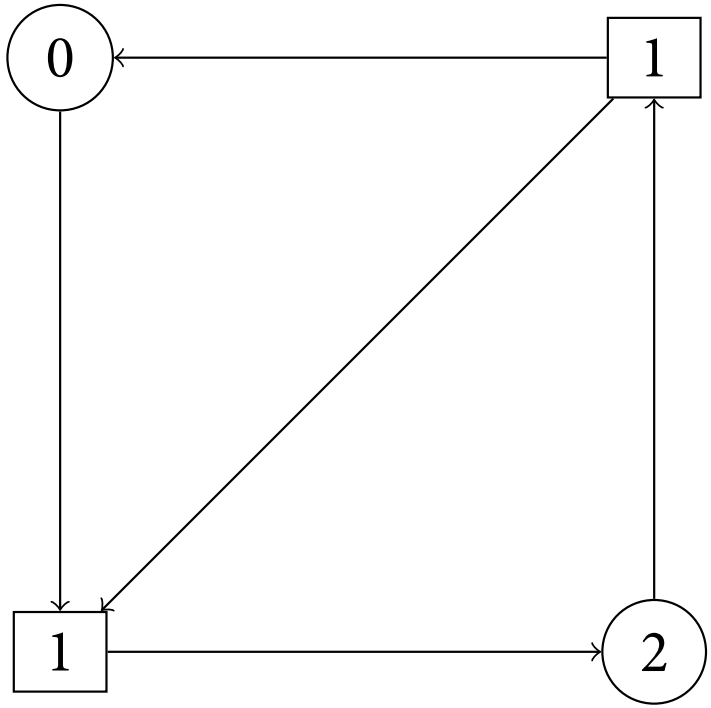


How People Play Games

Gomoku (five in a row)

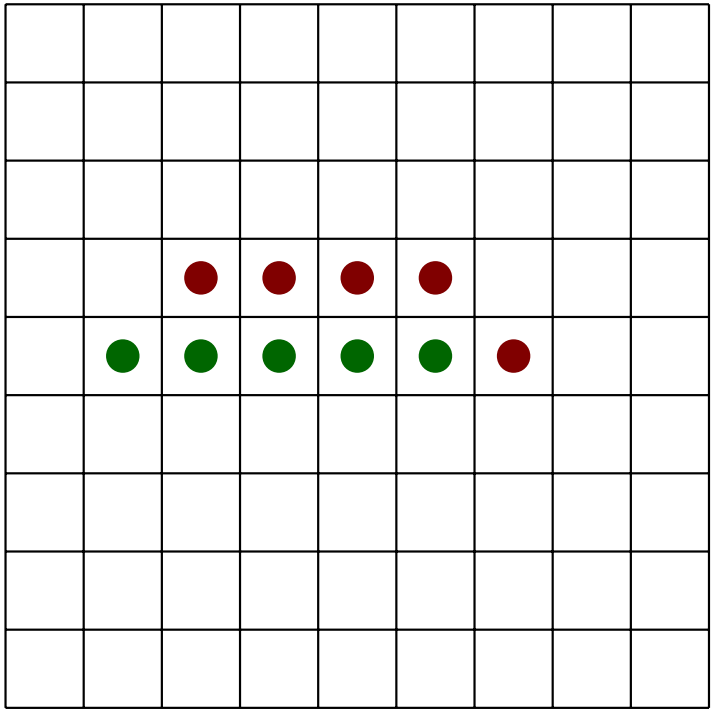


Parity game

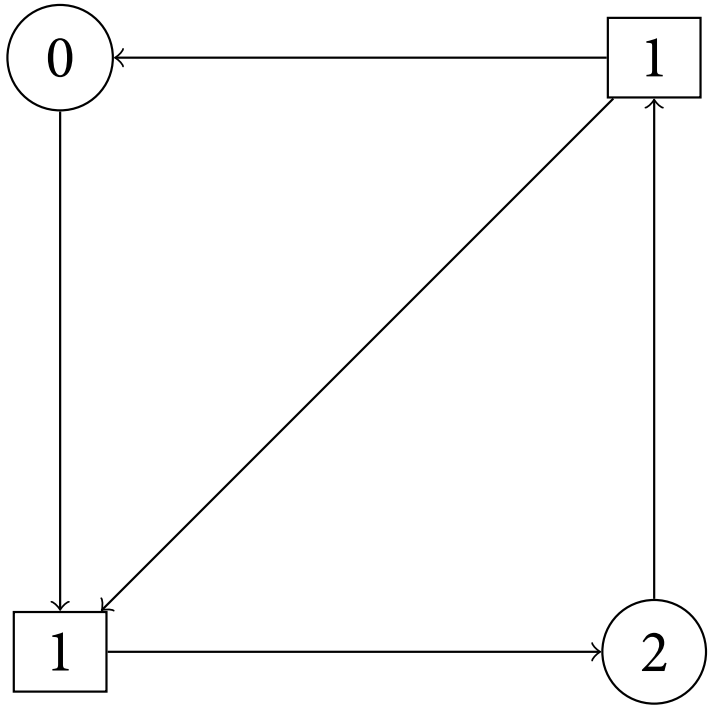


How People Play Games

Gomoku (five in a row)



Parity game



When state space is infinite, we often (again) consider its structure

- to construct **algorithms**
- to prove **interesting properties**

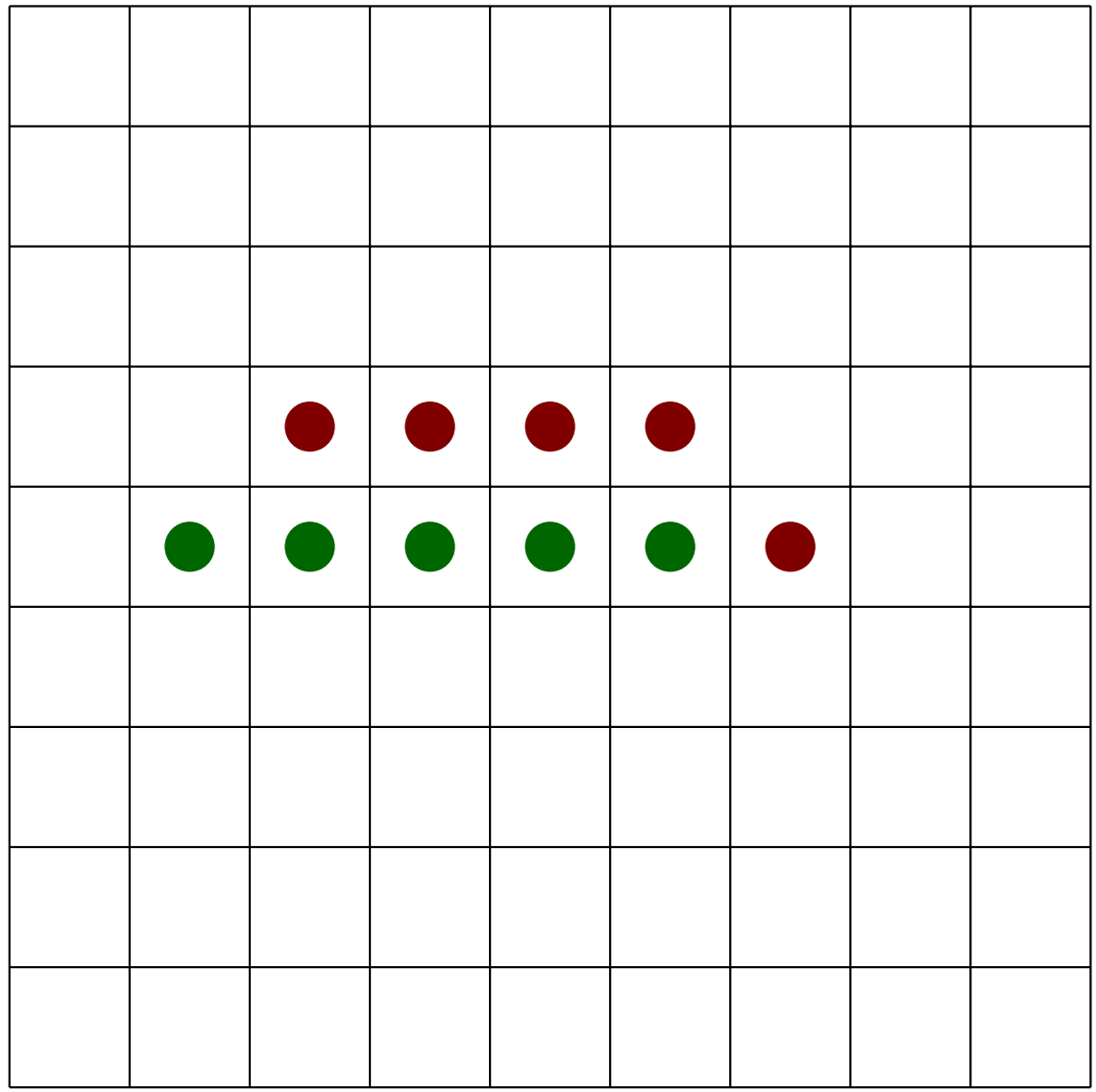
Example: pushdown games

A Natural Game Presentation

Gomoku Rules

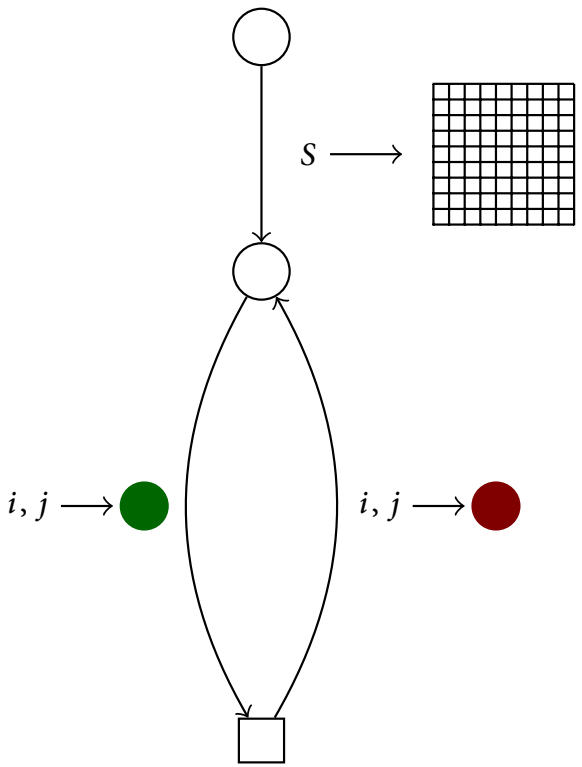
- 9×9 board
- Player 0 puts
- Player 1 puts
- Players alternate

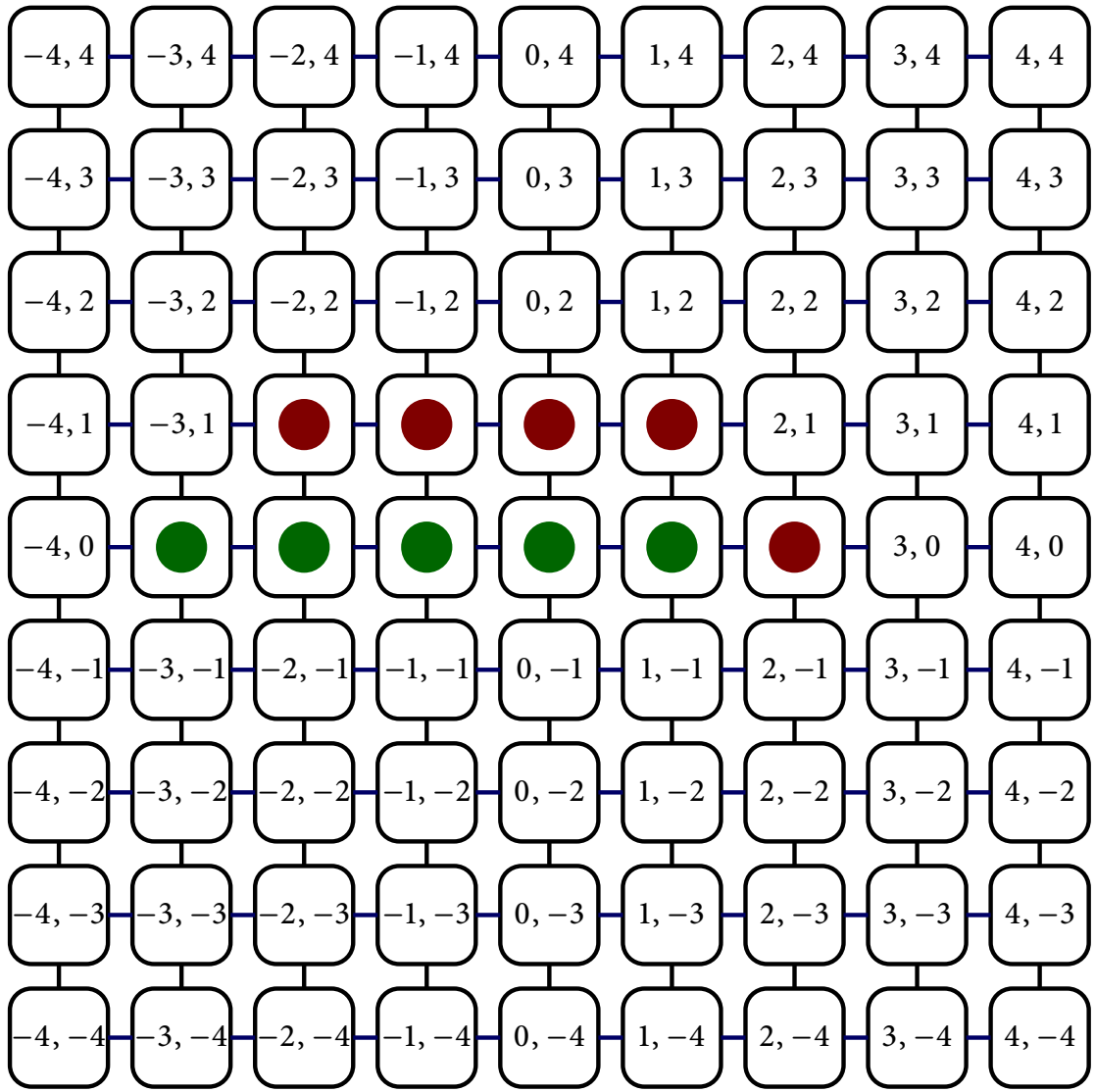
Gomoku Board



A Natural Game Presentation

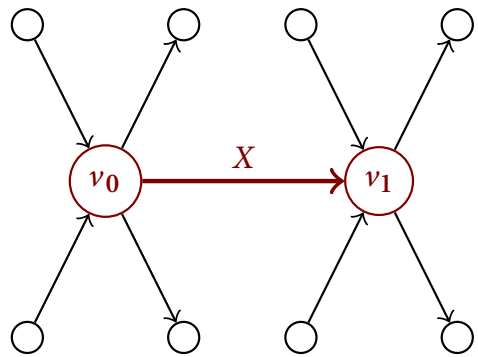
Representing the Rules and the Gomoku Board





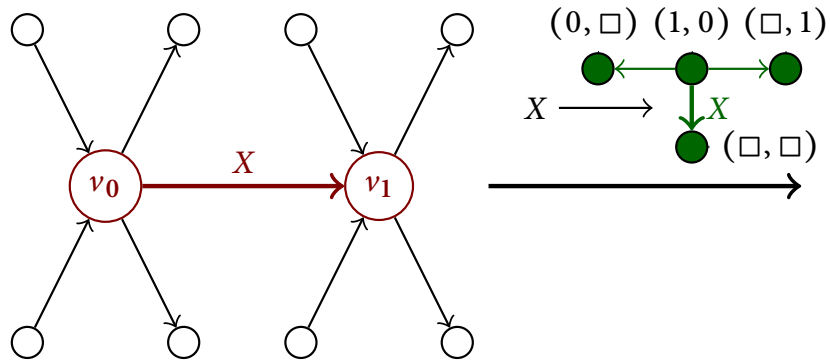
Games Played with Hypergraphs

Handle Rewrite Rules (Courcelle, Engelfriet, Rozenberg, 1991)



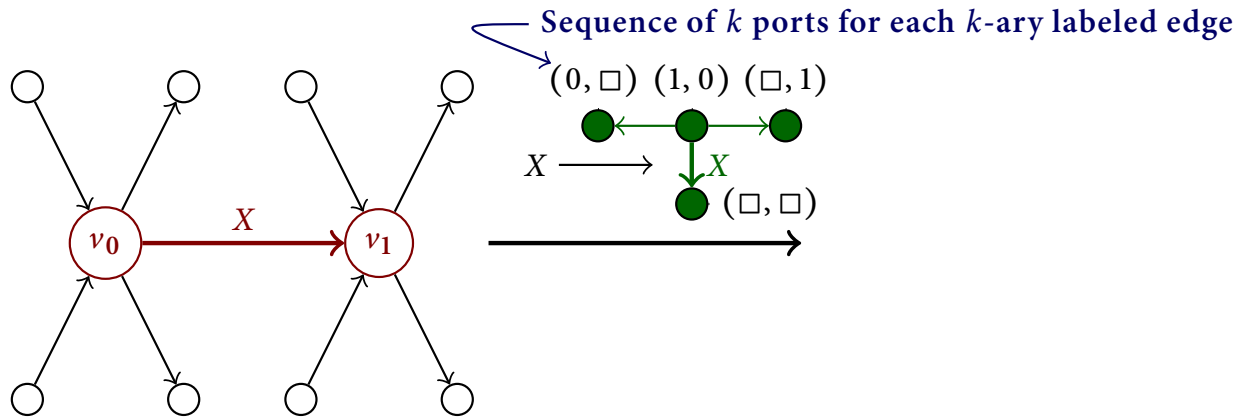
Games Played with Hypergraphs

Handle Rewrite Rules (Courcelle, Engelfriet, Rozenberg, 1991)



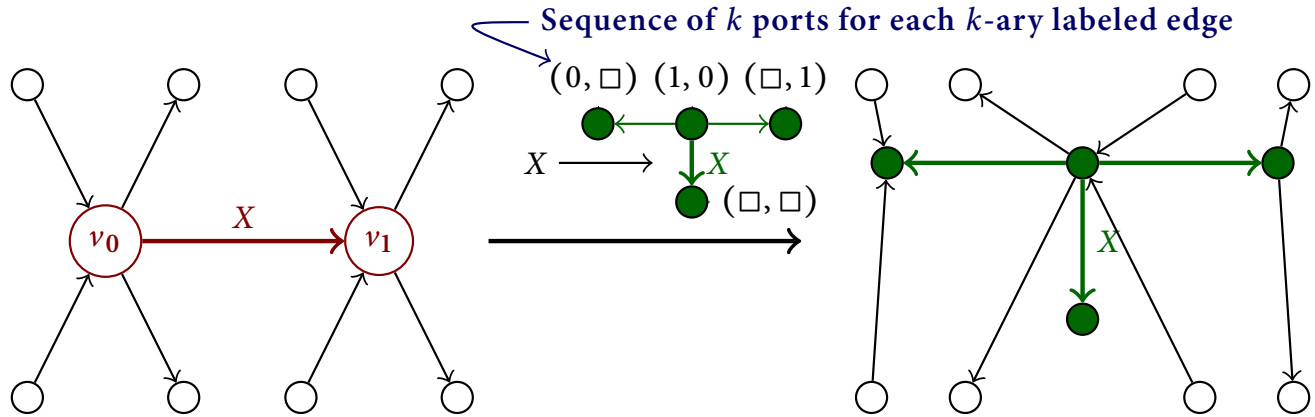
Games Played with Hypergraphs

Handle Rewrite Rules (Courcelle, Engelfriet, Rozenberg, 1991)



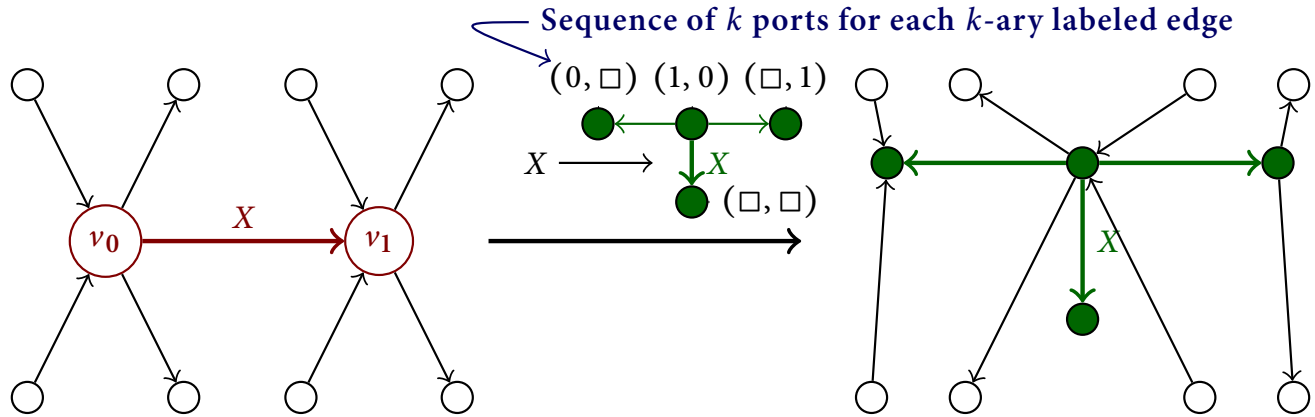
Games Played with Hypergraphs

Handle Rewrite Rules (Courcelle, Engelfriet, Rozenberg, 1991)



Games Played with Hypergraphs

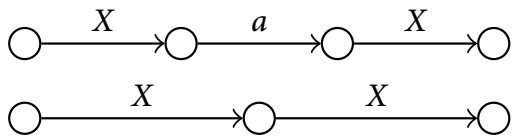
Handle Rewrite Rules (Courcelle, Engelfriet, Rozenberg, 1991)



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

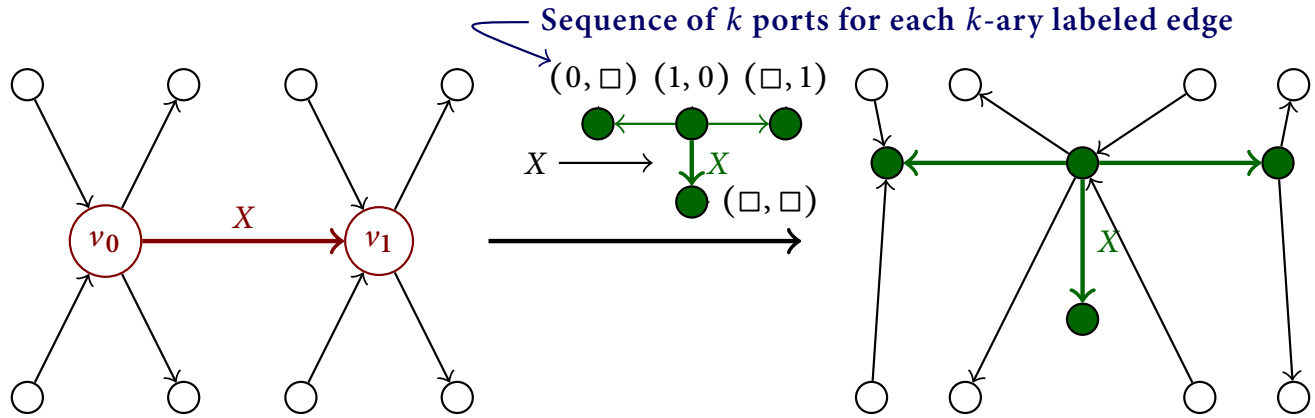
Separated:

Not Separated:



Games Played with Hypergraphs

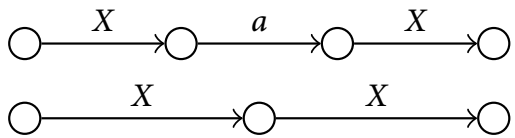
Handle Rewrite Rules (Courcelle, Engelfriet, Rozenberg, 1991)



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

Separated:

Not Separated:



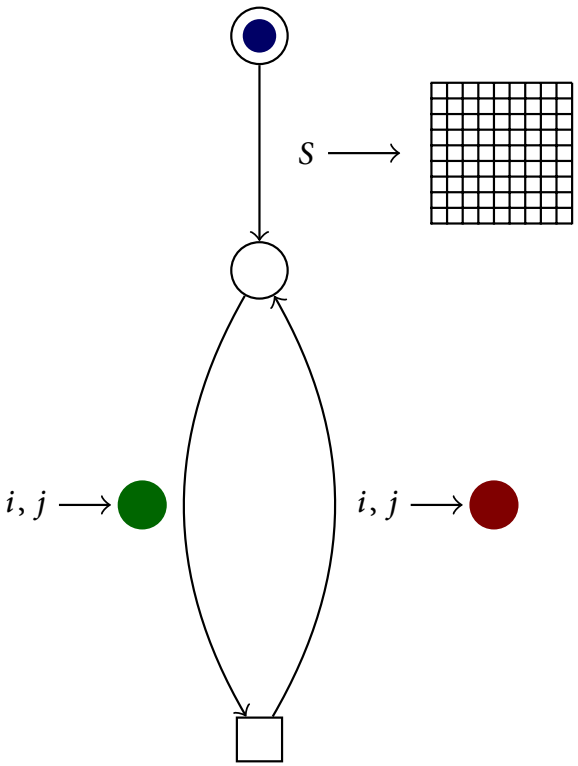
Rewriting Sequences and Limit Hypergraphs

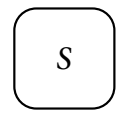
- $G[X \to H]$ is G with **all*** occurrences of X rewritten to H

  * if players **pick** positions: **undecidable**, see **Active context-free games**, thanks to **Anca Muscholl**

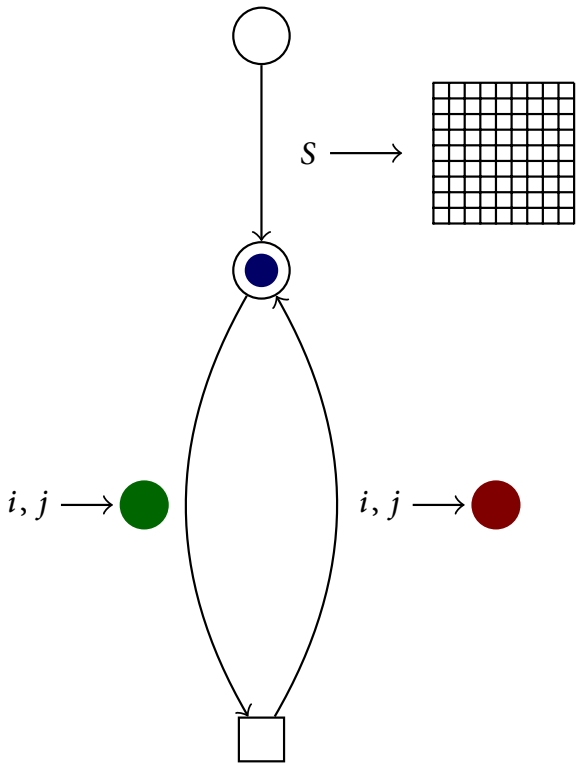
- **Limit** of $G_0 \to G_1 \to G_3 \to \ldots$: $(\bigcup_{n \in \mathbb{N}} \bigcap_{i \geq n} V_i, \bigcup_{n \in \mathbb{N}} \bigcap_{i \geq n} E_i)$

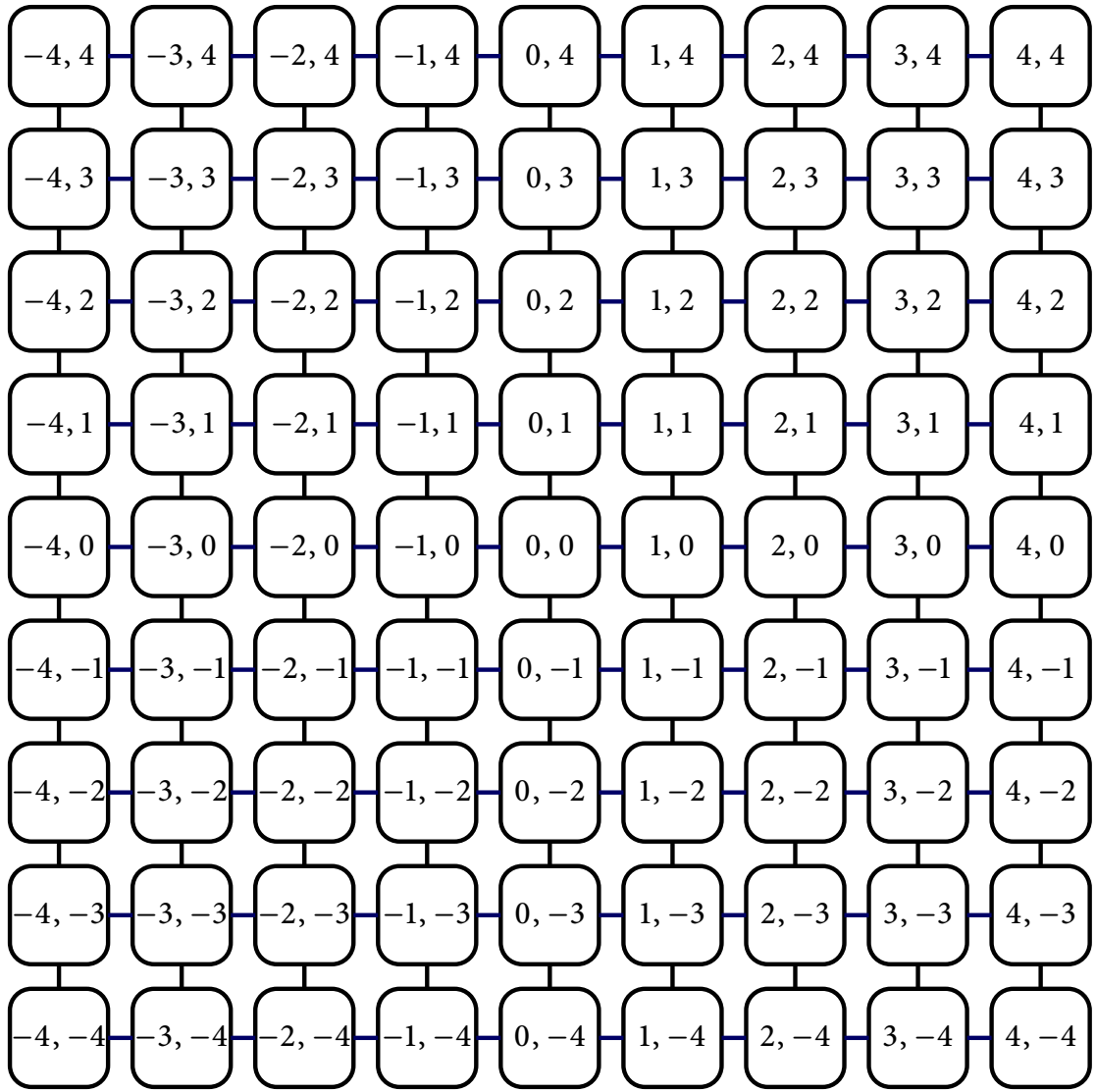
Example: Playing Gomoku



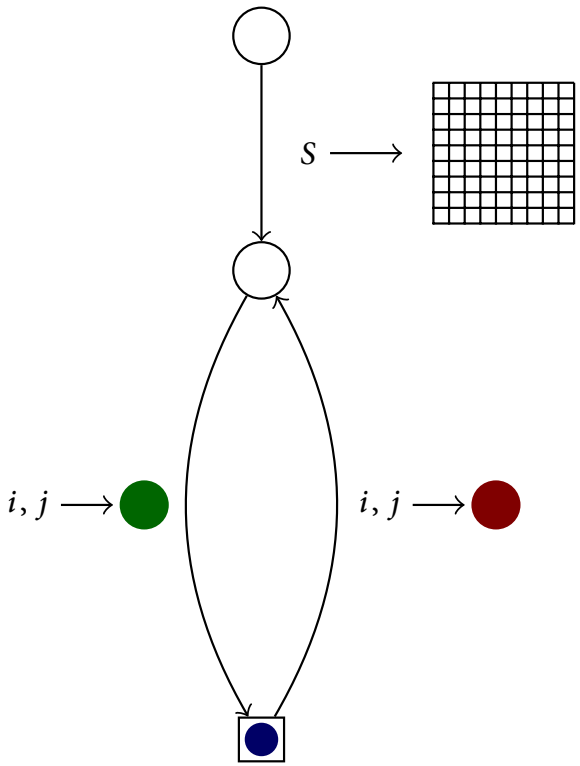


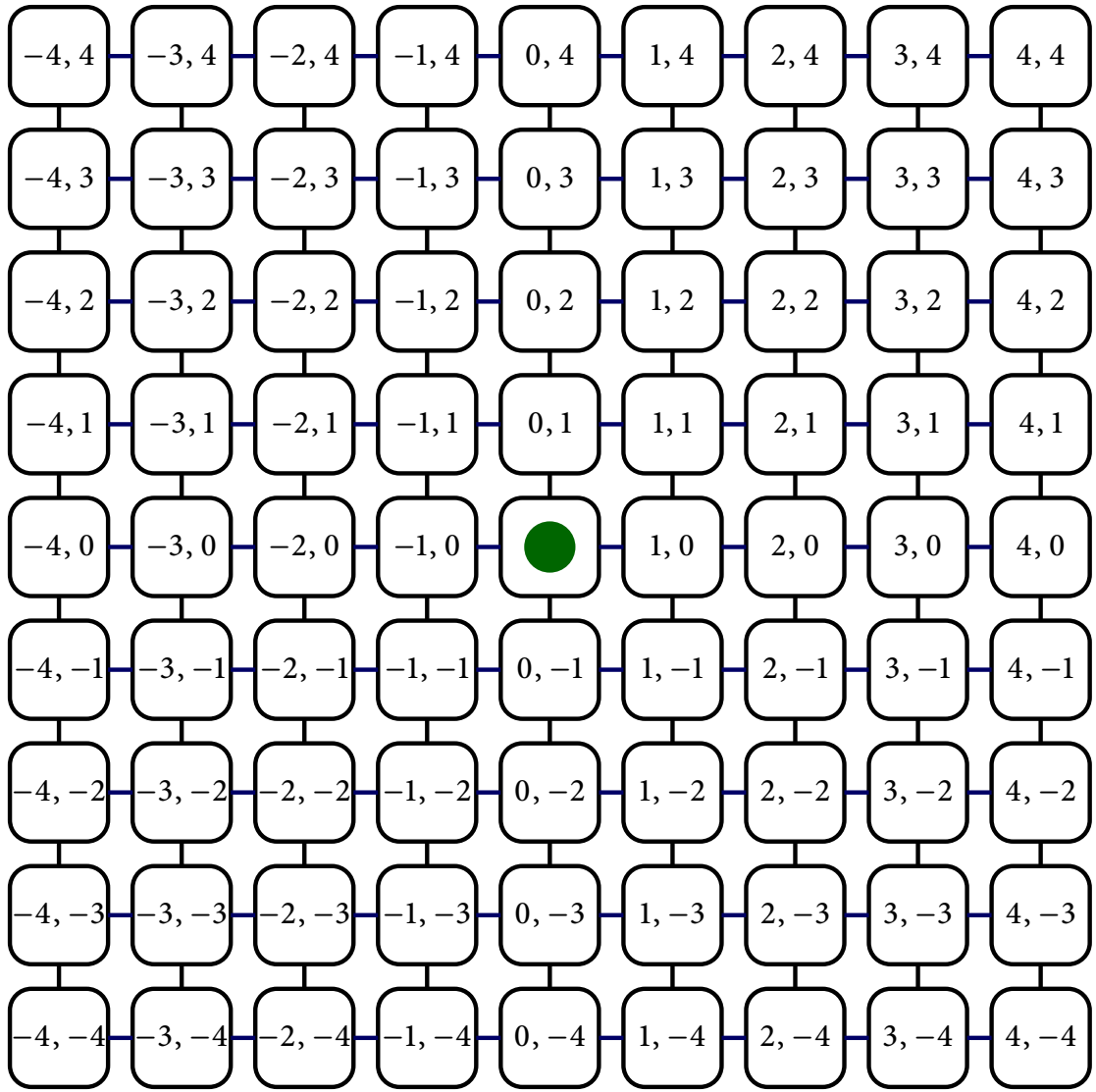
Example: Playing Gomoku



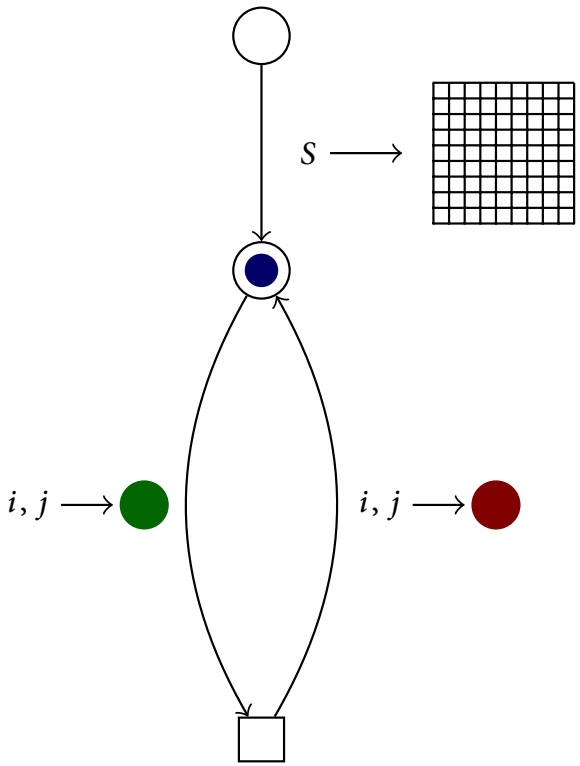


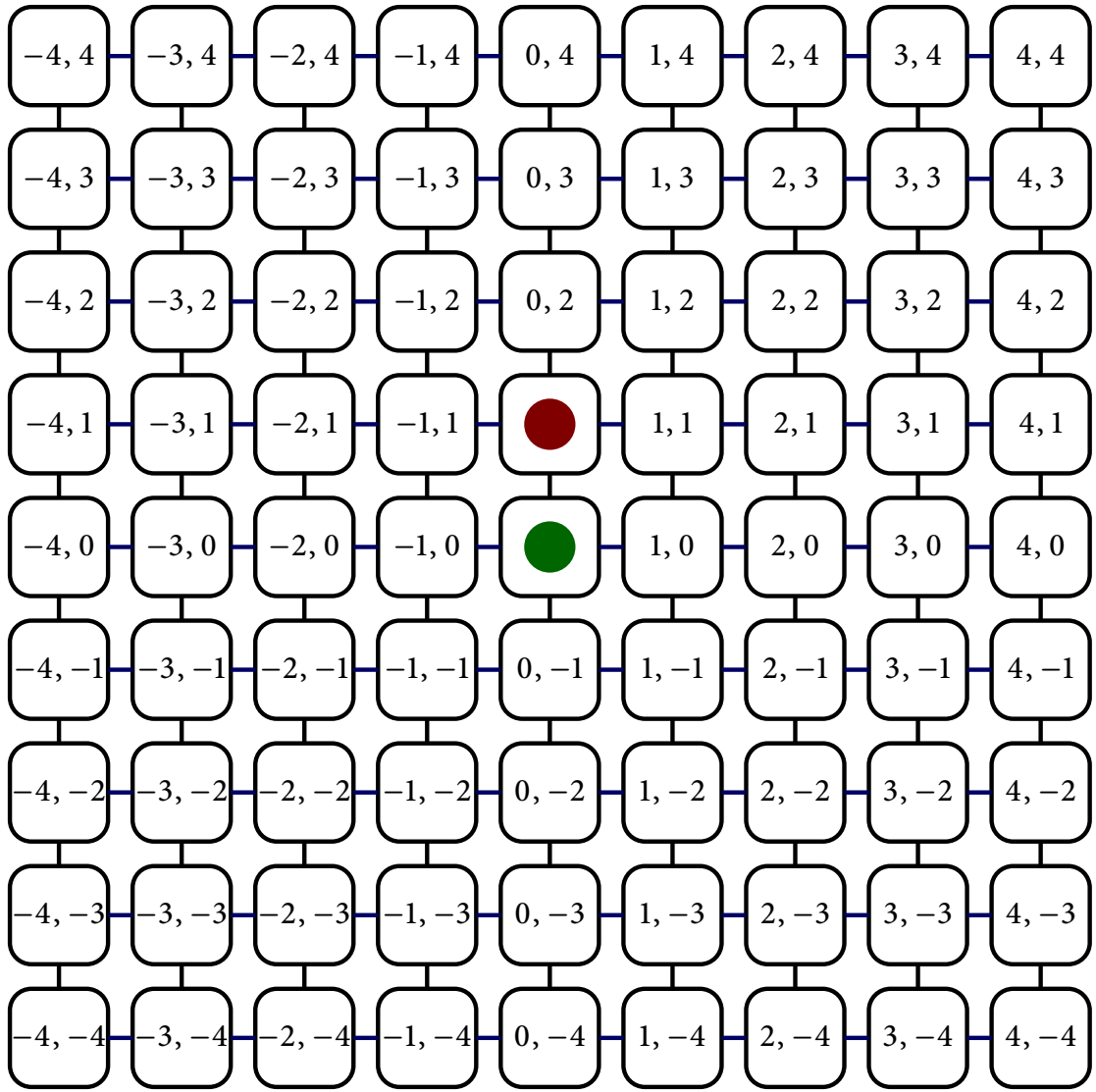
Example: Playing Gomoku



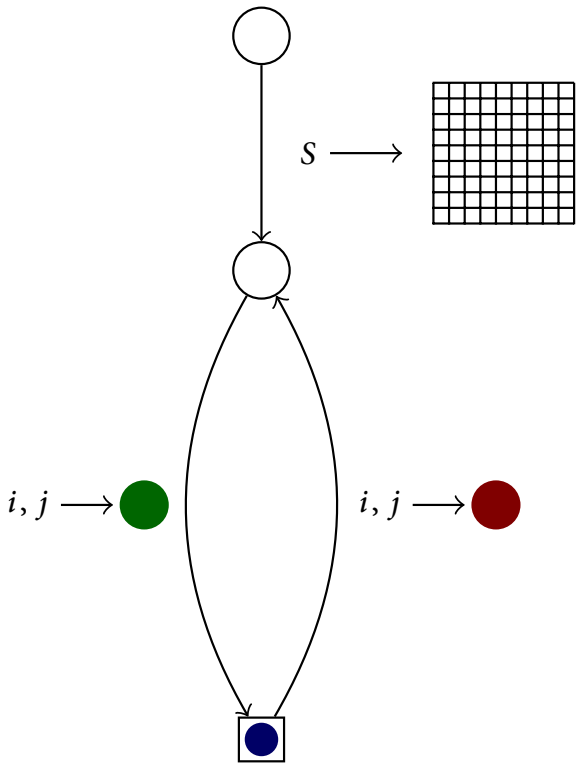


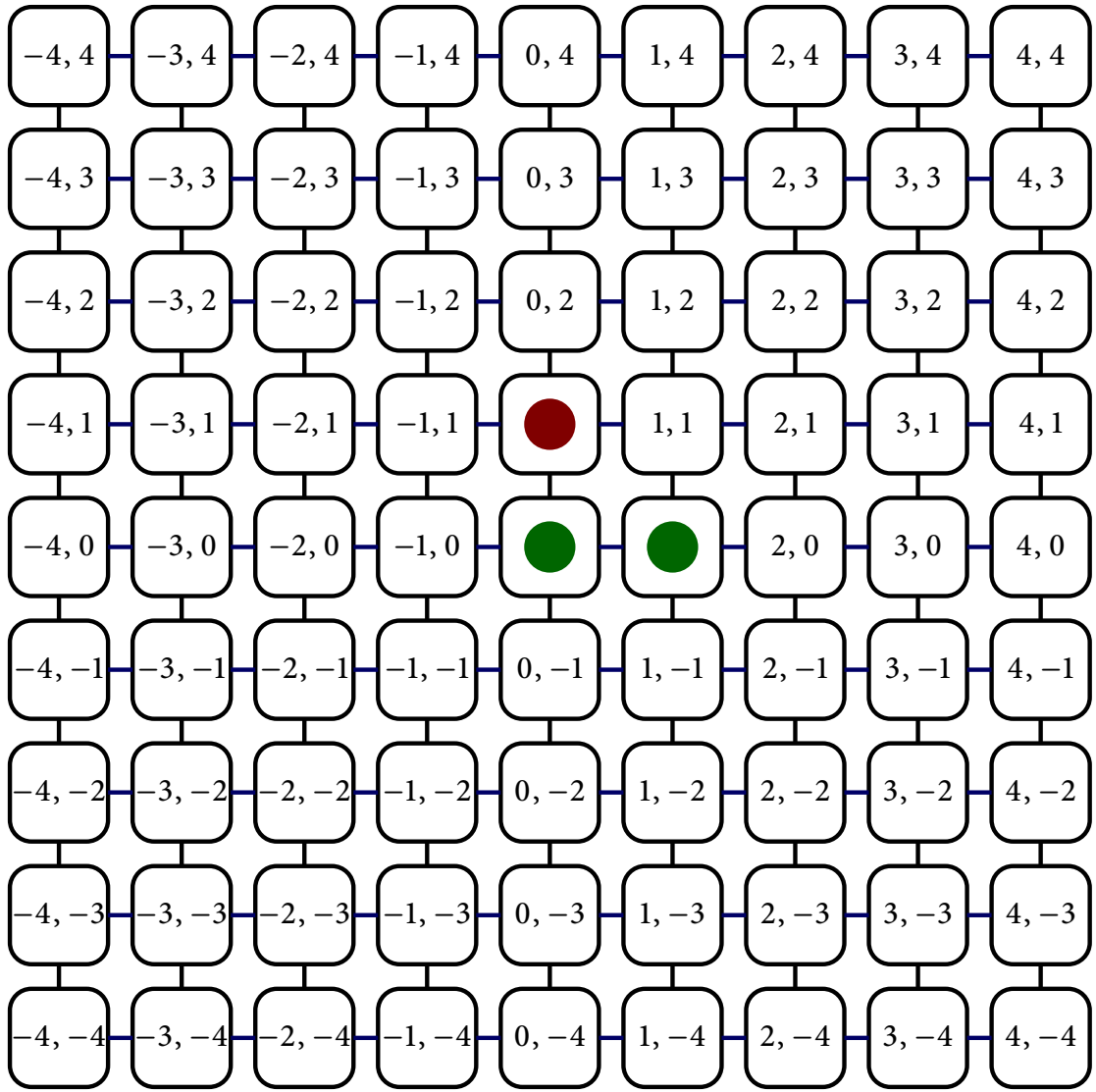
Example: Playing Gomoku



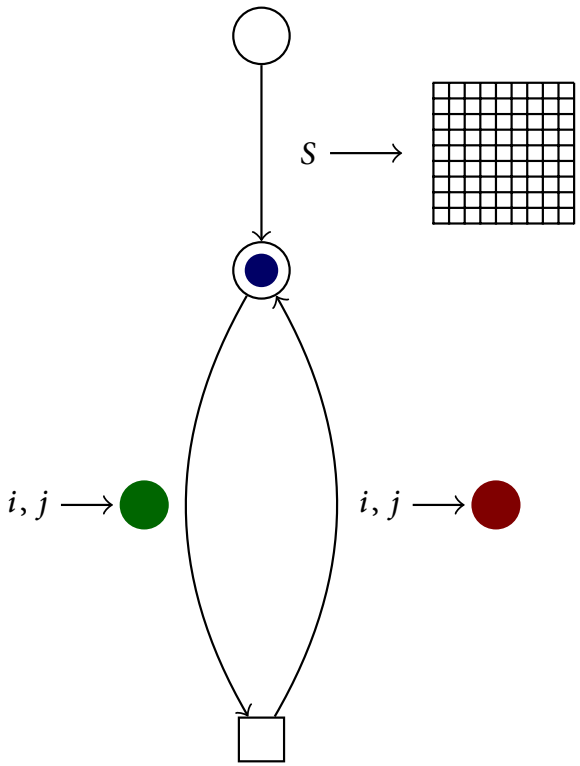


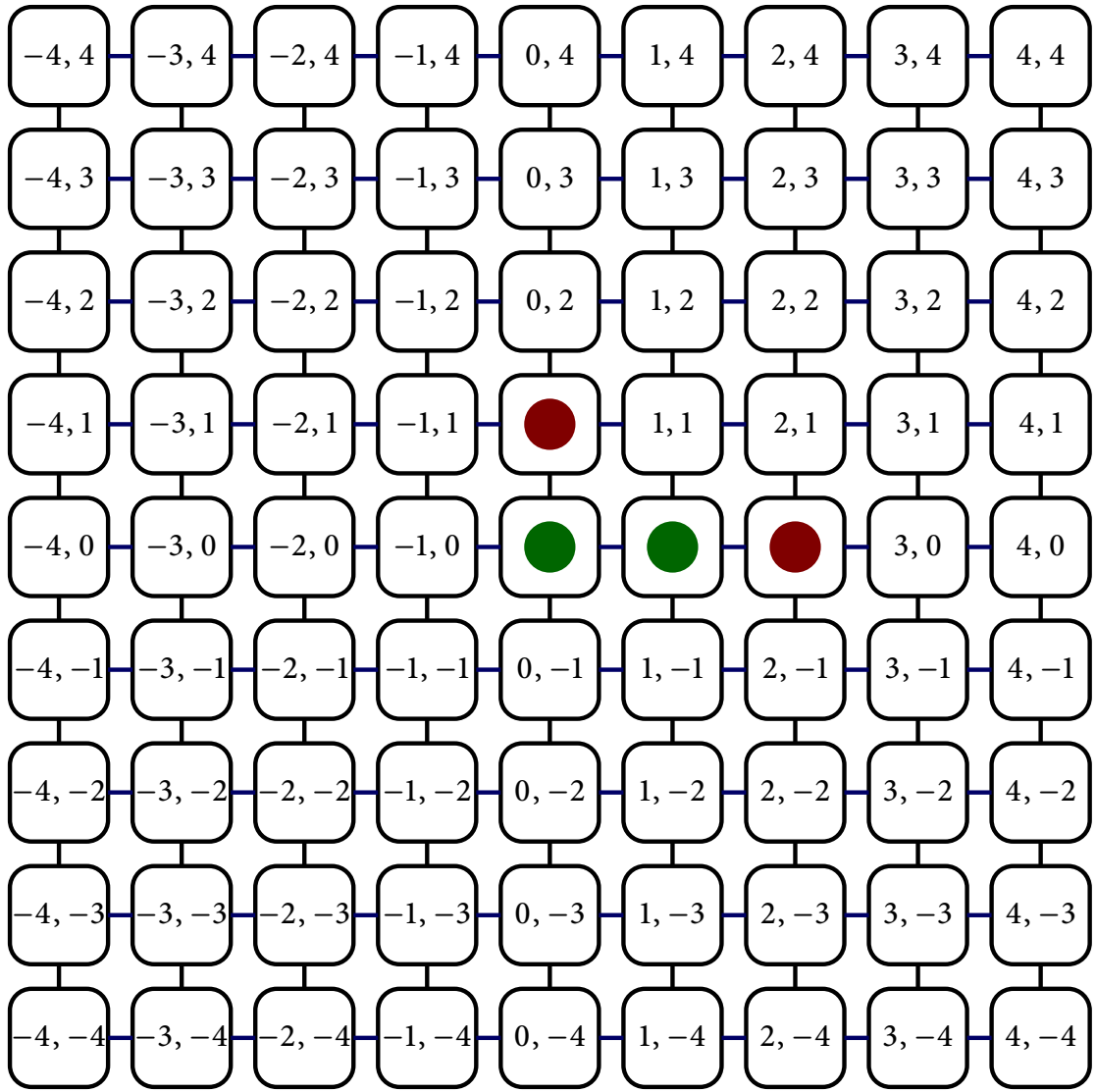
Example: Playing Gomoku



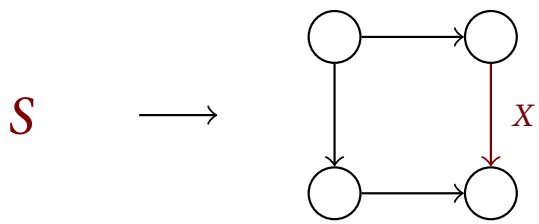


Example: Playing Gomoku

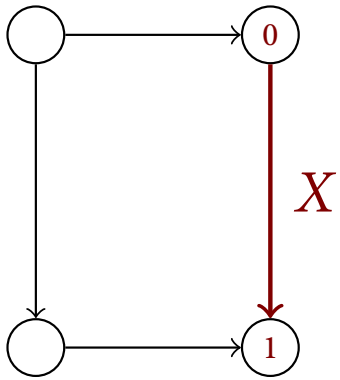


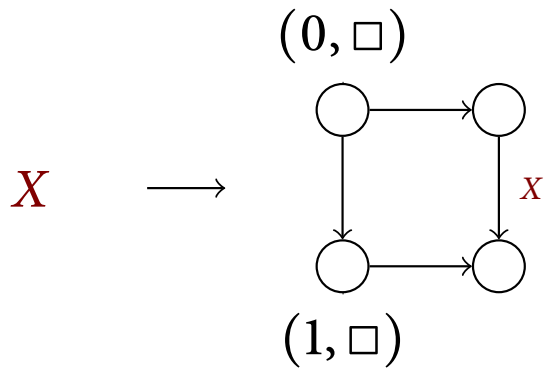


Example: Constructing a Ladder

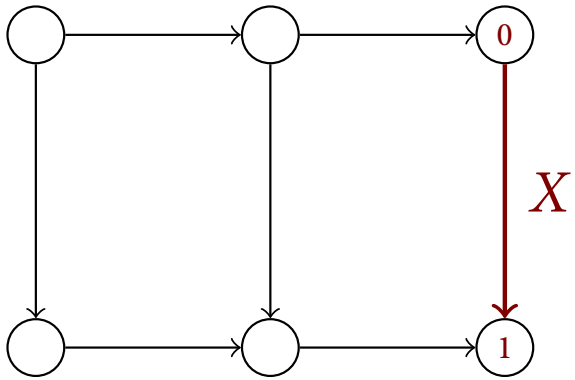


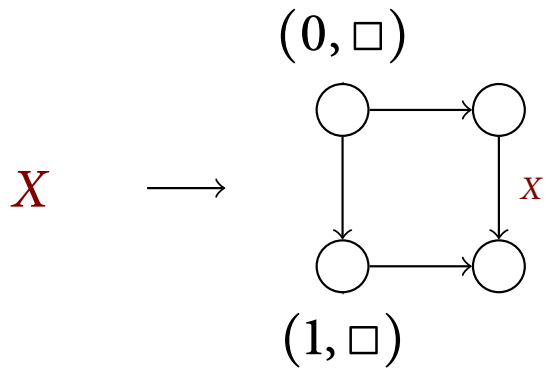
Example: Constructing a Ladder



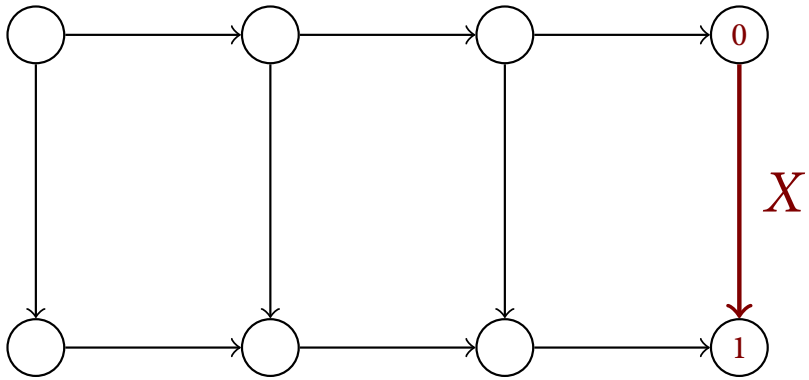


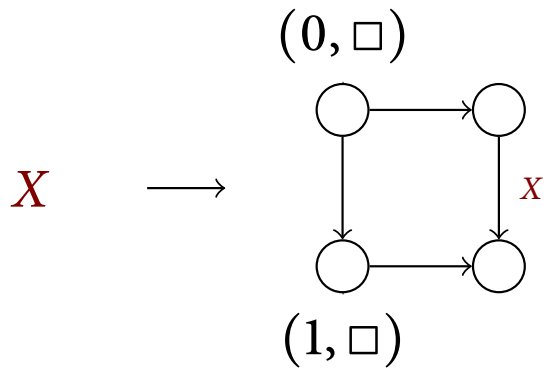
Example: Constructing a Ladder



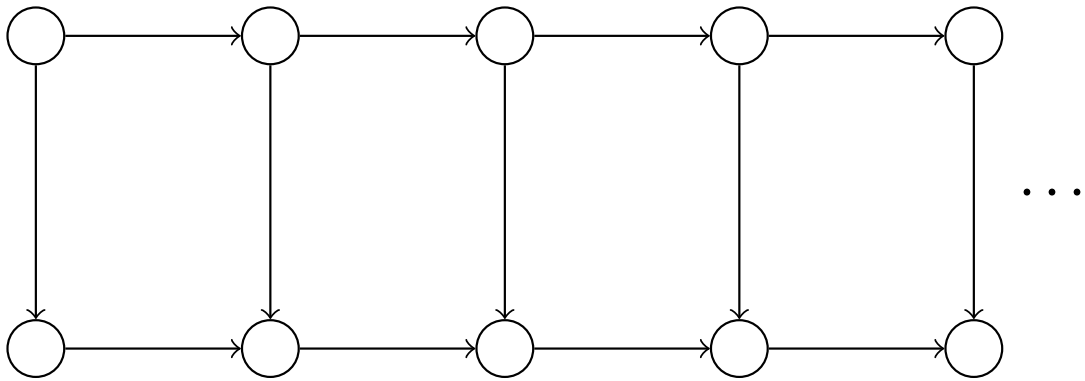


Example: Constructing a Ladder





Example: Constructing a Ladder



Main Result

Theorem

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

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

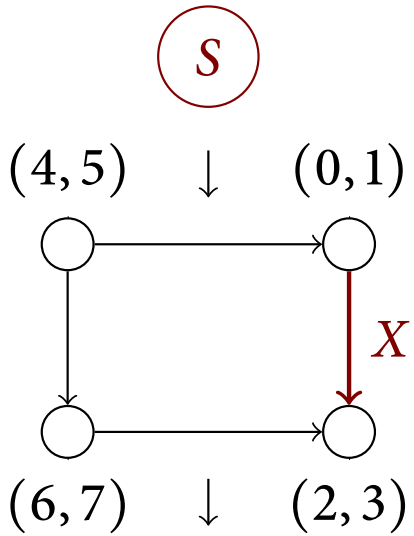
Corollary

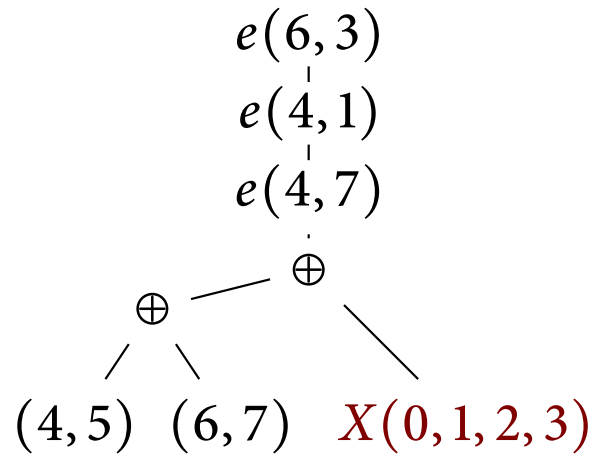
Establishing the winner of finite separated handle rewriting games is decidable.

Proof: Separated Graph Rewriting as a Tree

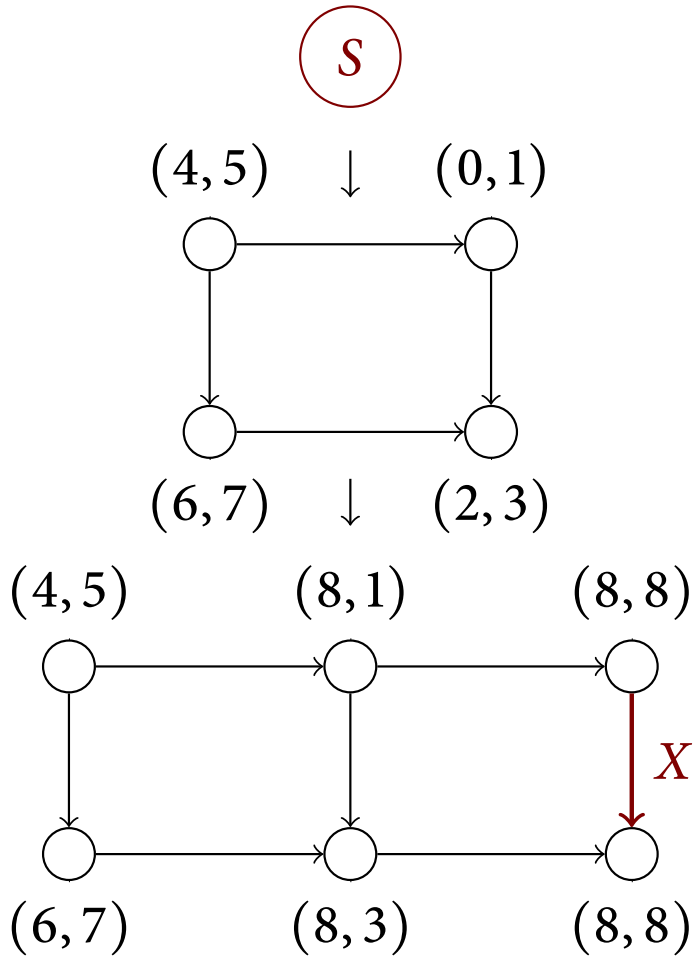
S

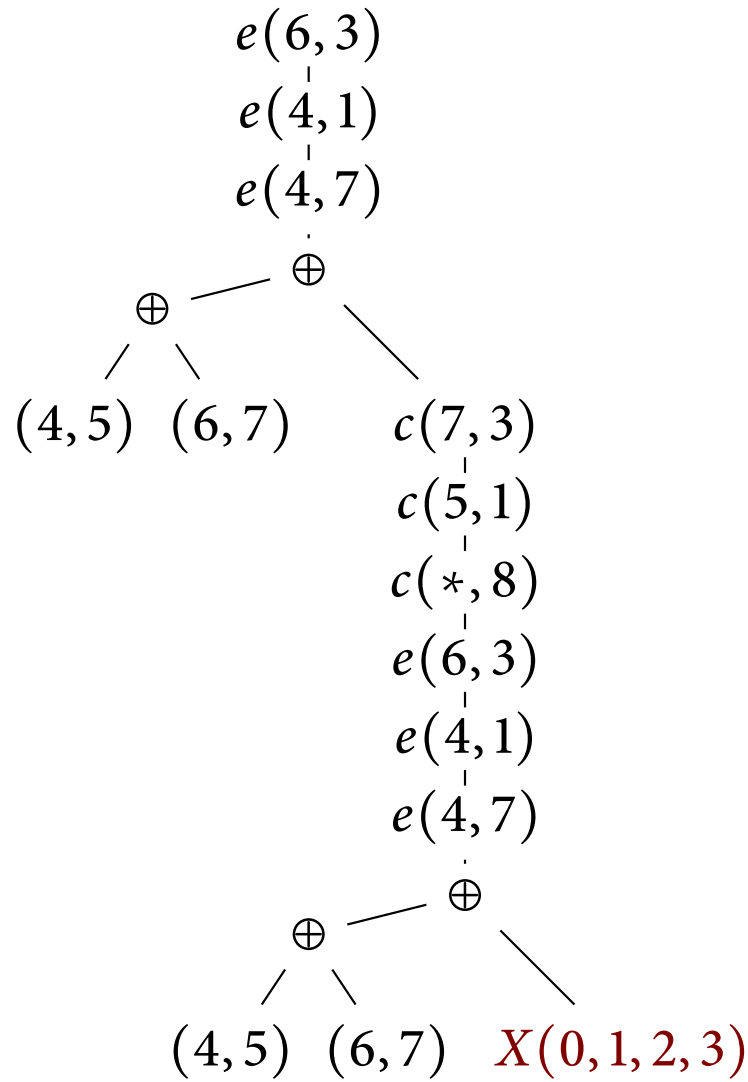
Proof: Separated Graph Rewriting as a Tree



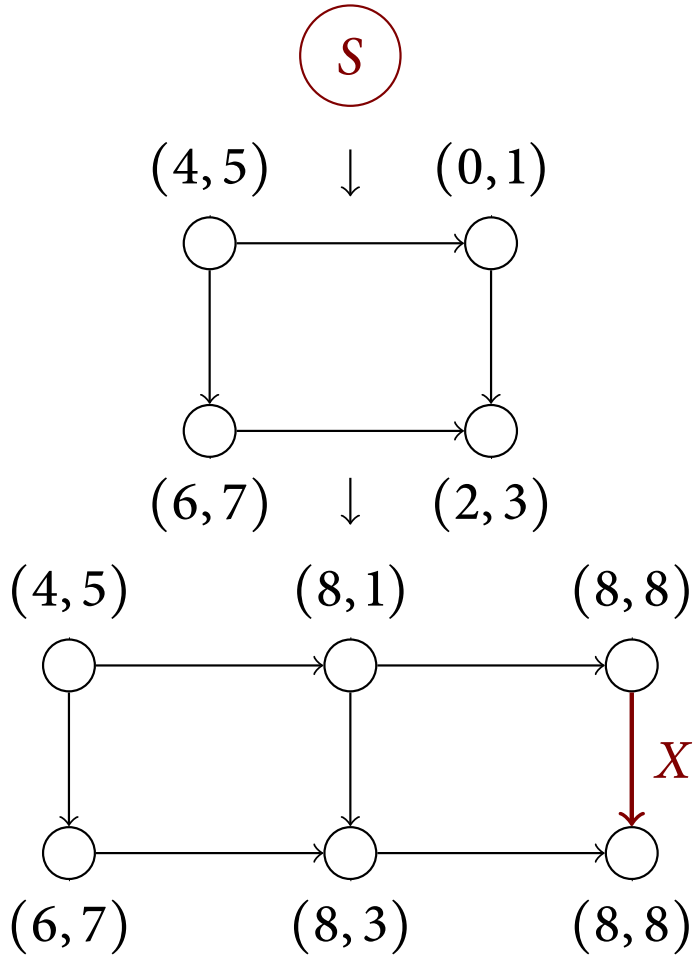


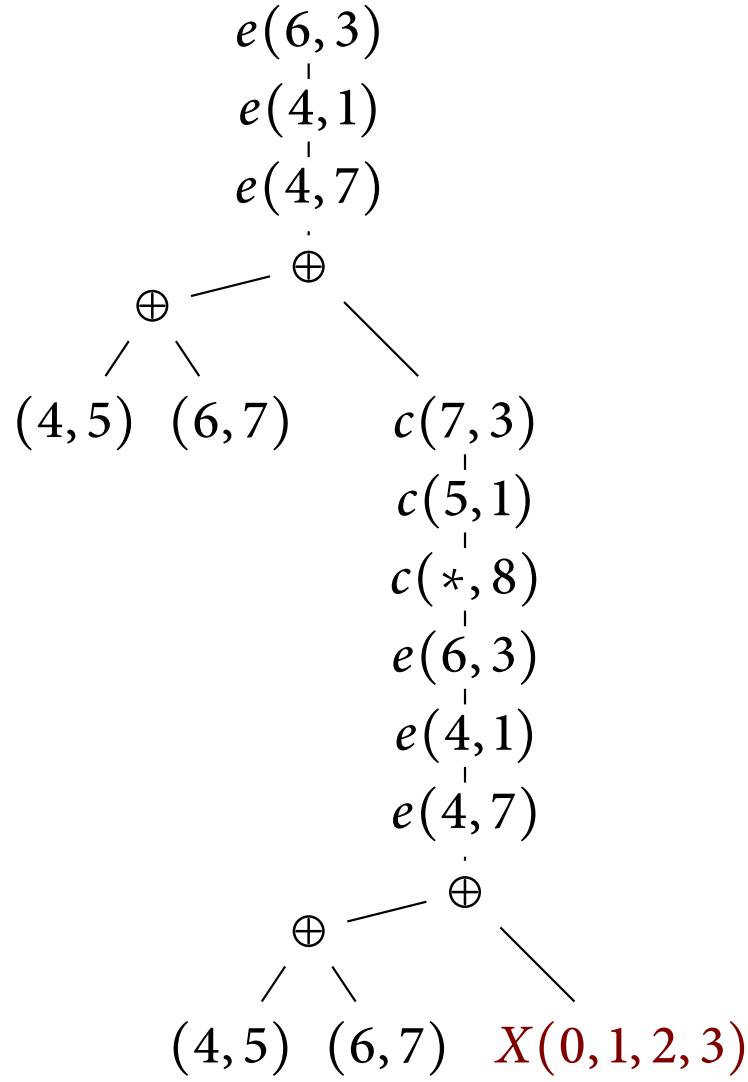
Proof: Separated Graph Rewriting as a Tree





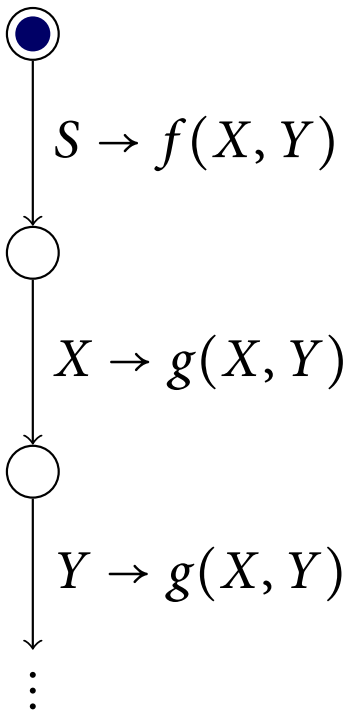
Proof: Separated Graph Rewriting as a Tree



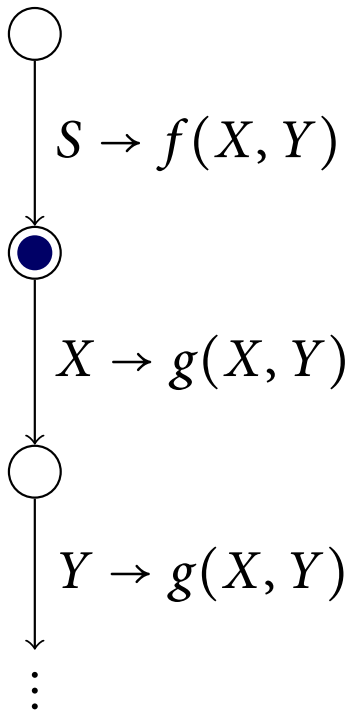


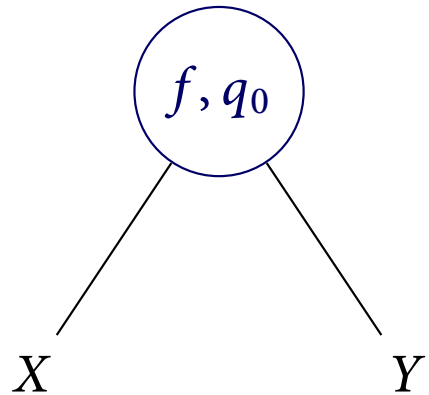
MSO-to-MSO **interpretation:** $\varphi \to \psi$

Proof: From Tree to Alternating Word Automata

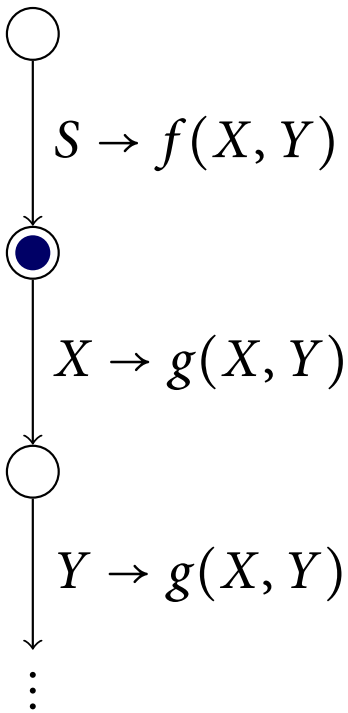


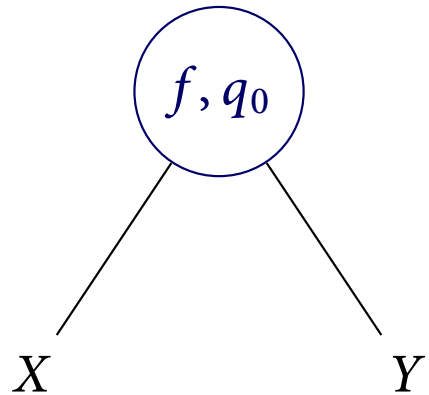
Proof: From Tree to Alternating Word Automata





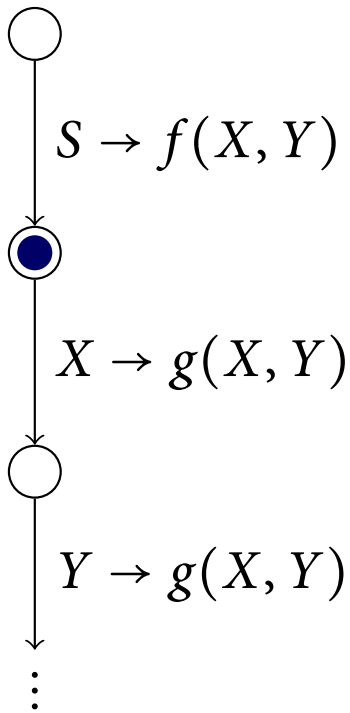
Proof: From Tree to Alternating Word Automata

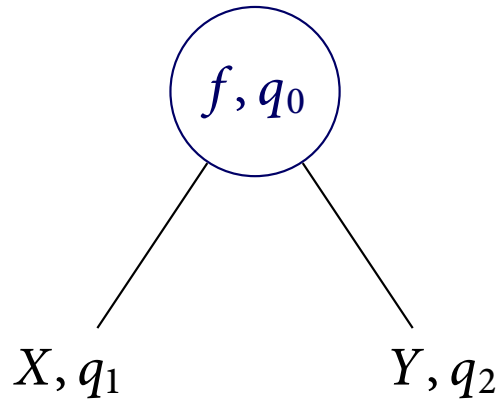




existential: pick transition

Proof: From Tree to Alternating Word Automata

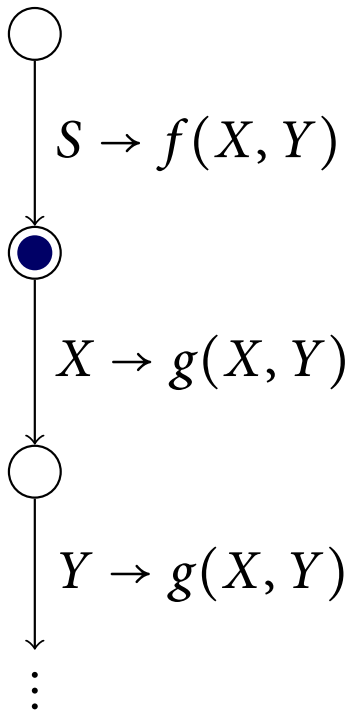


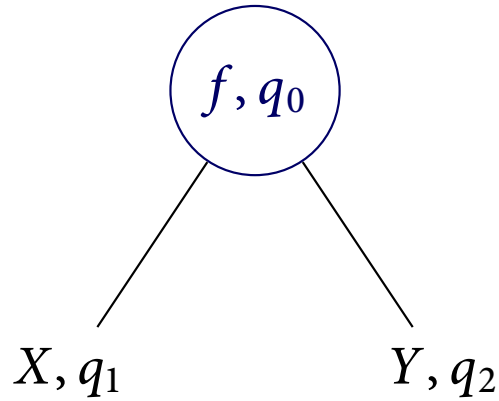


existential: pick transition

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

Proof: From Tree to Alternating Word Automata



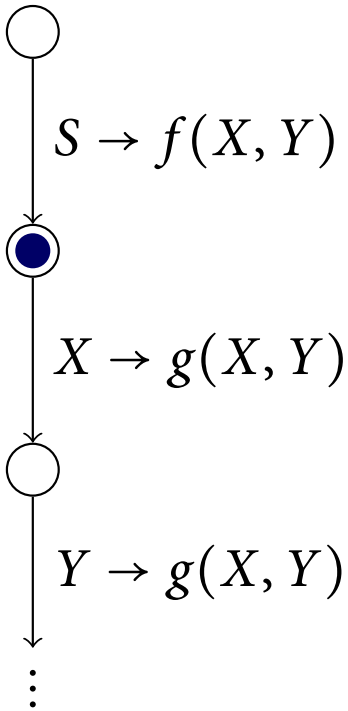


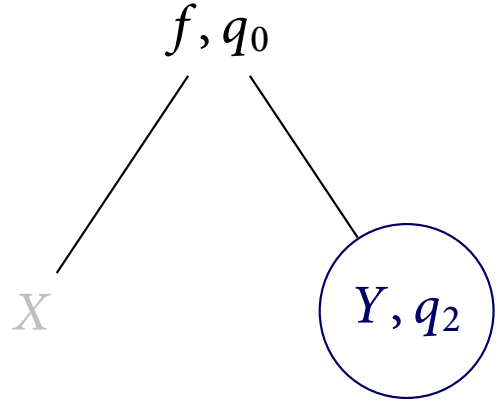
existential: pick transition

universal: left or right

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

Proof: From Tree to Alternating Word Automata



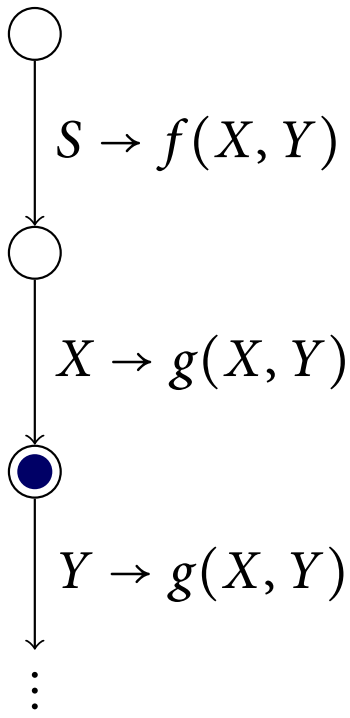


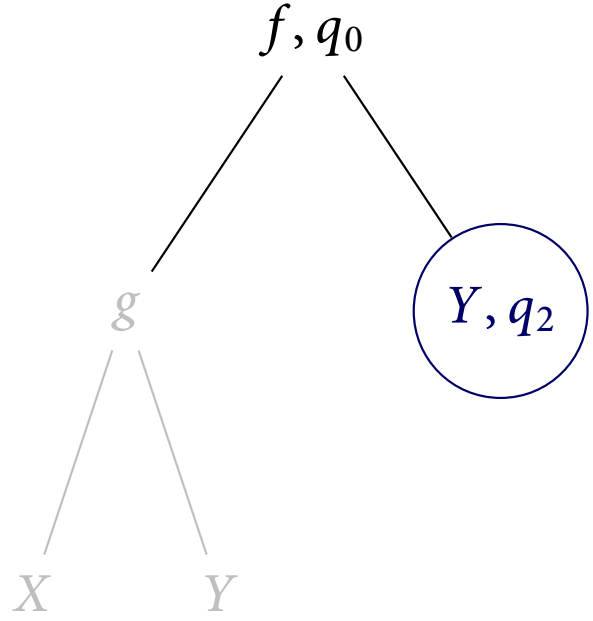
existential: pick transition

universal: left or right

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

Proof: From Tree to Alternating Word Automata





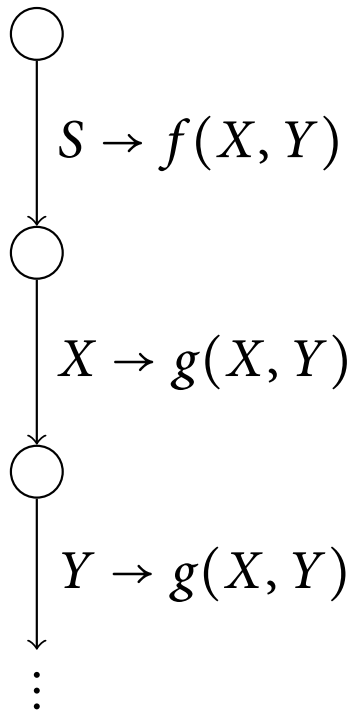
existential: pick transition

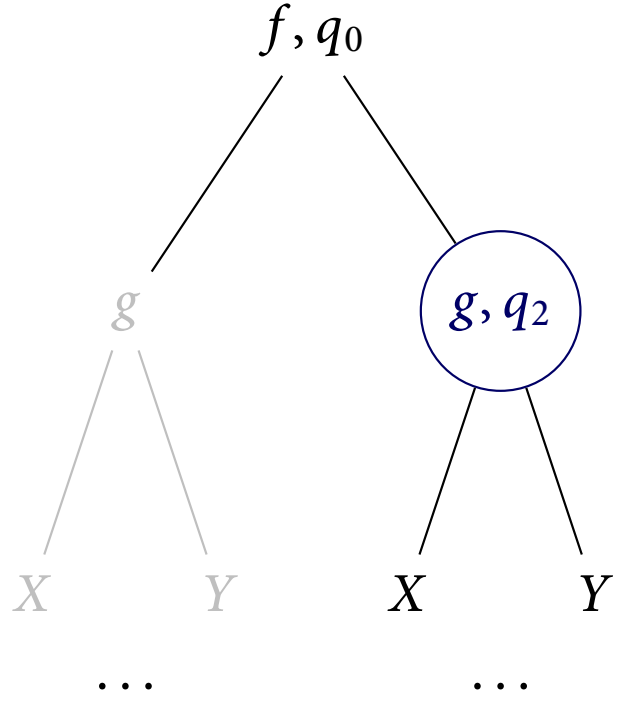
universal: left or right

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

ignore

Proof: From Tree to Alternating Word Automata





existential: pick transition

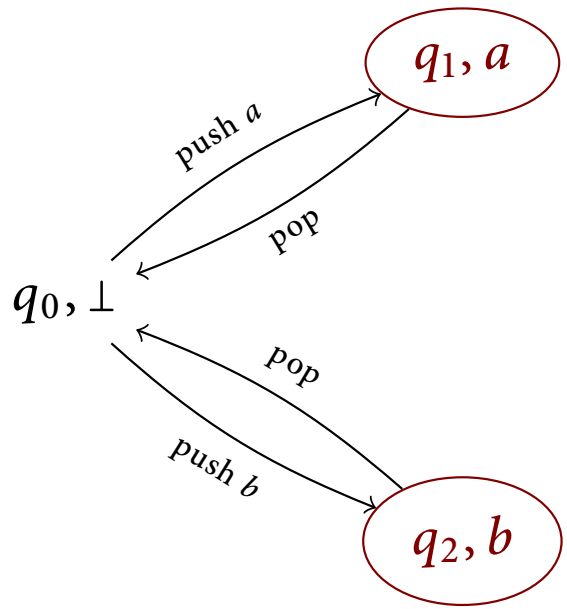
universal: left or right

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

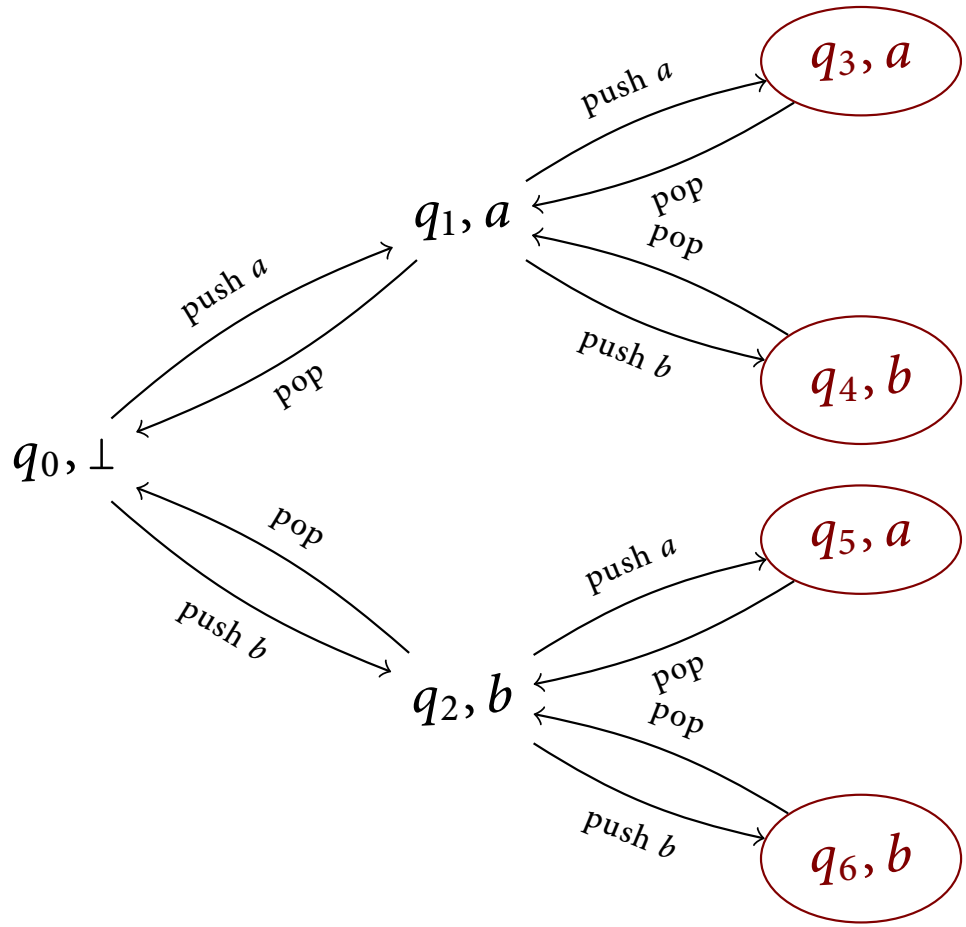
ignore

Application: Pushdown Parity Games

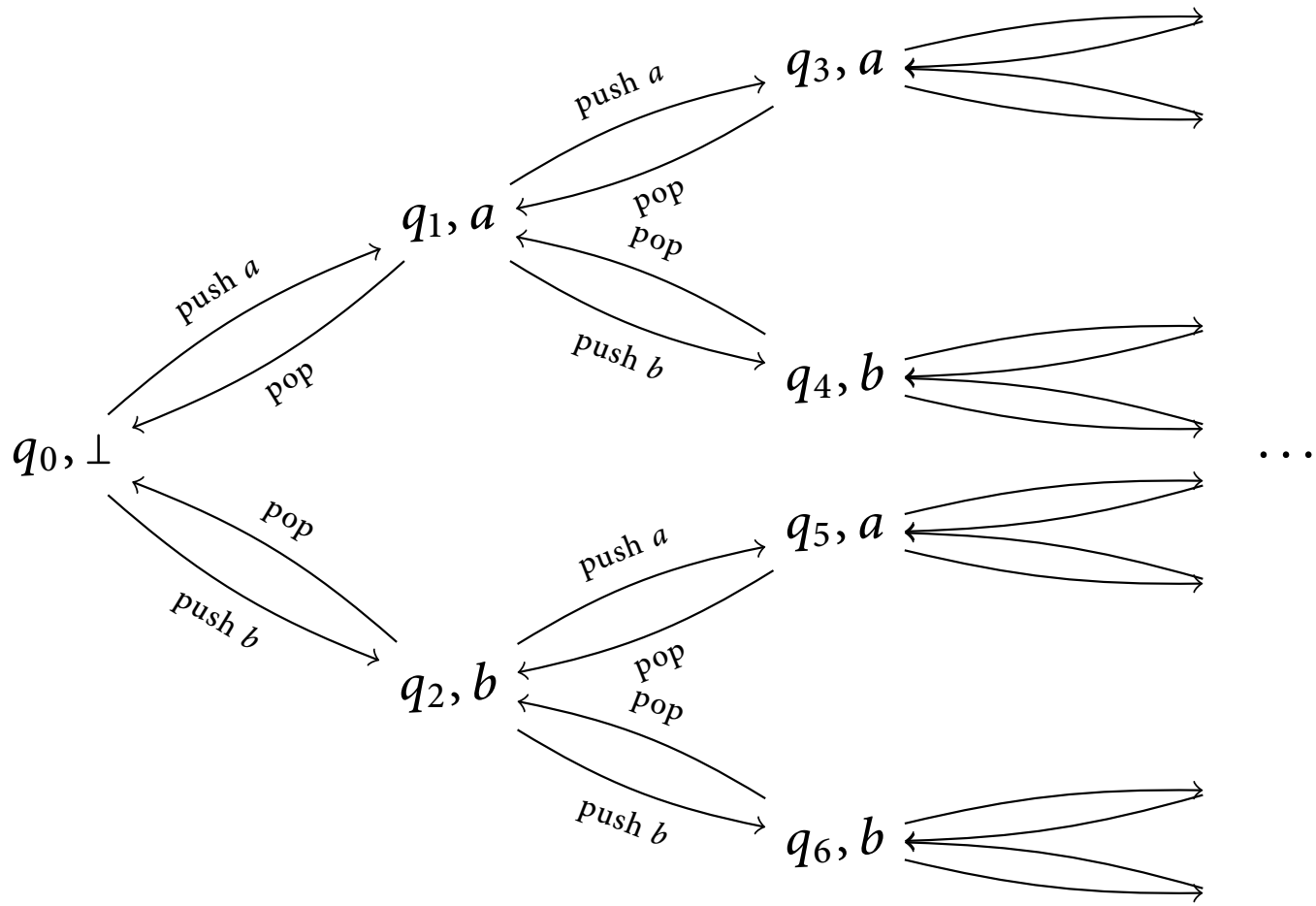
Application: Pushdown Parity Games



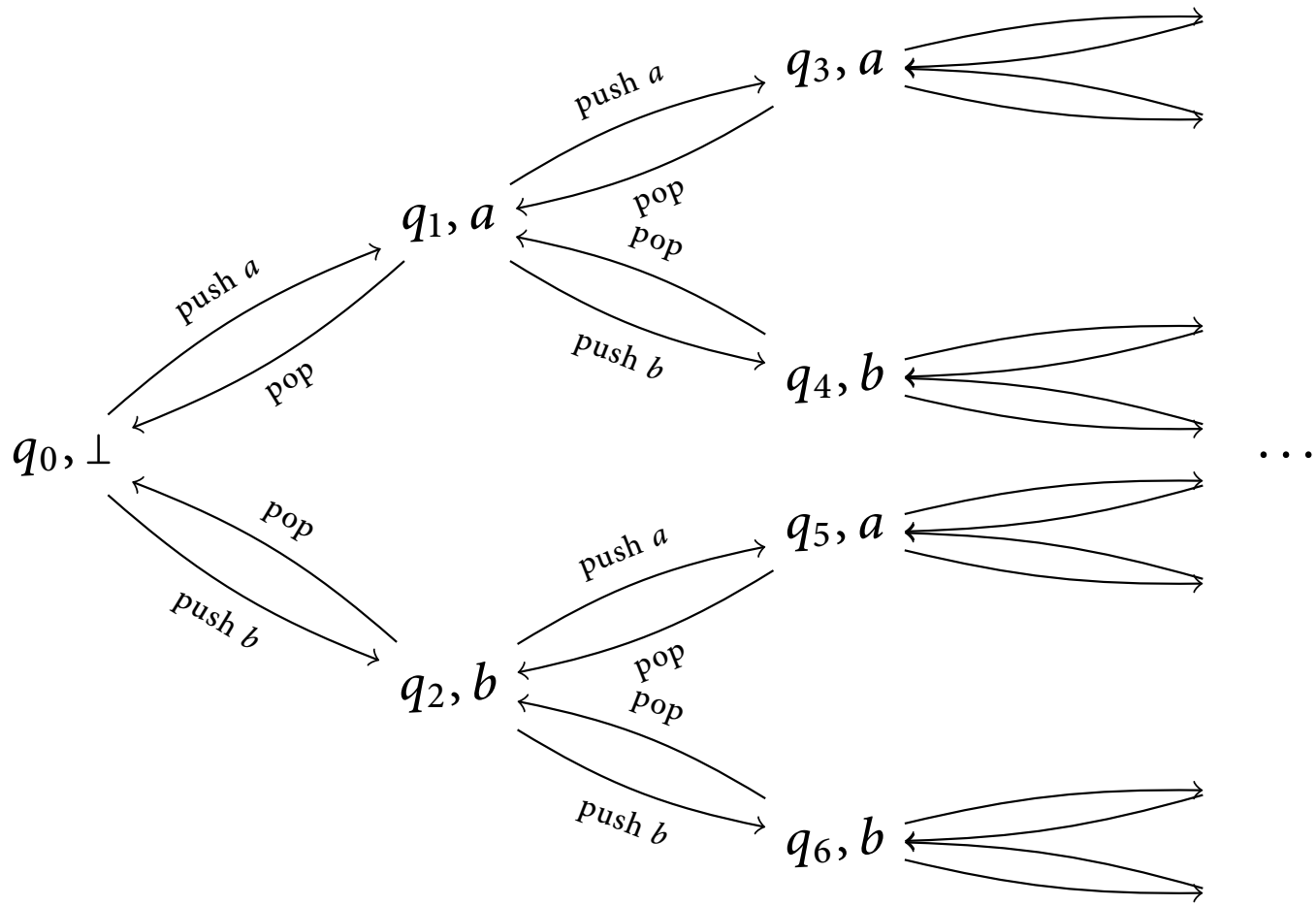
Application: Pushdown Parity Games



Application: Pushdown Parity Games

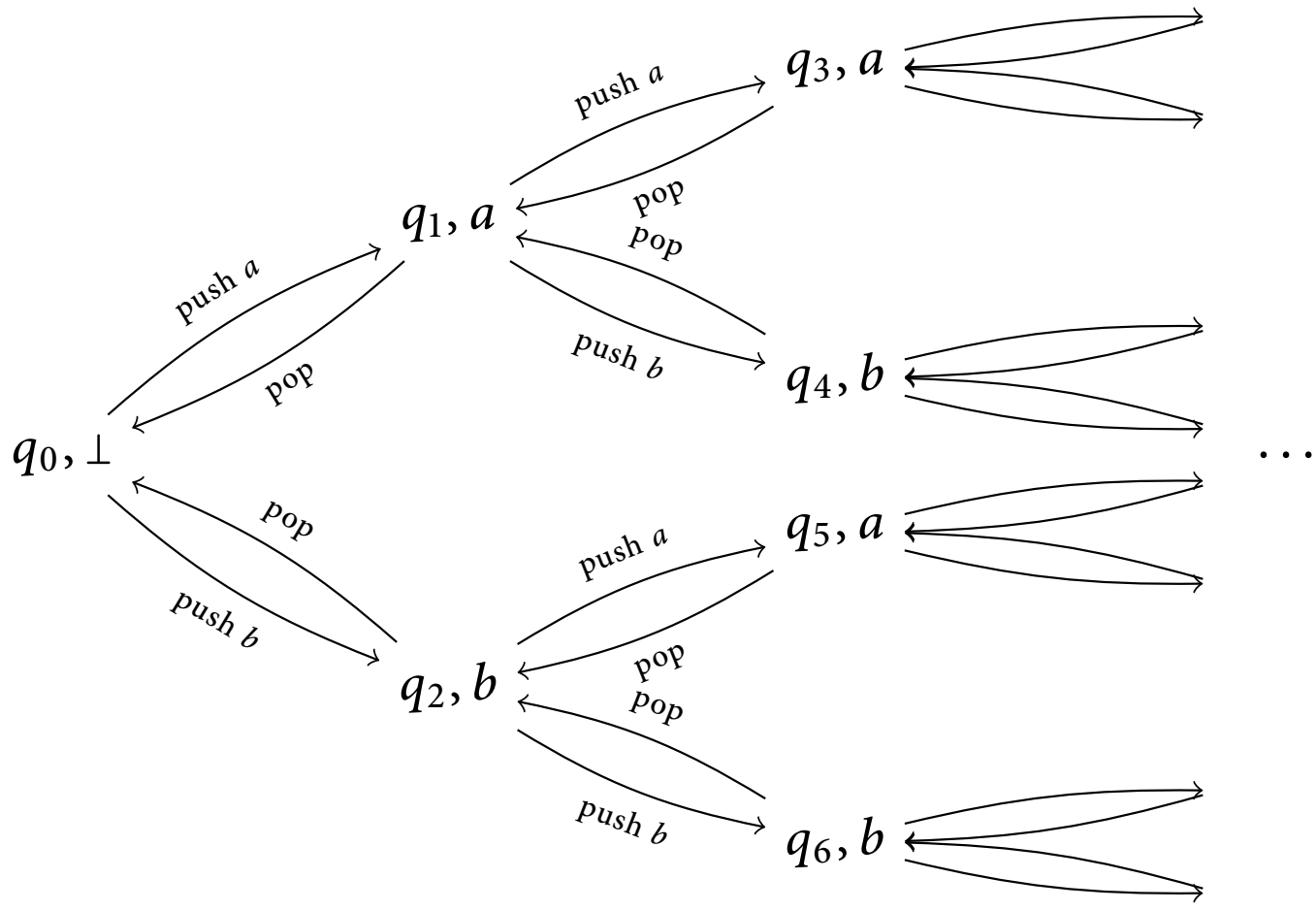


Application: Pushdown Parity Games



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

Application: FO on Automatic Structures

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

Application: FO on Automatic Structures

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

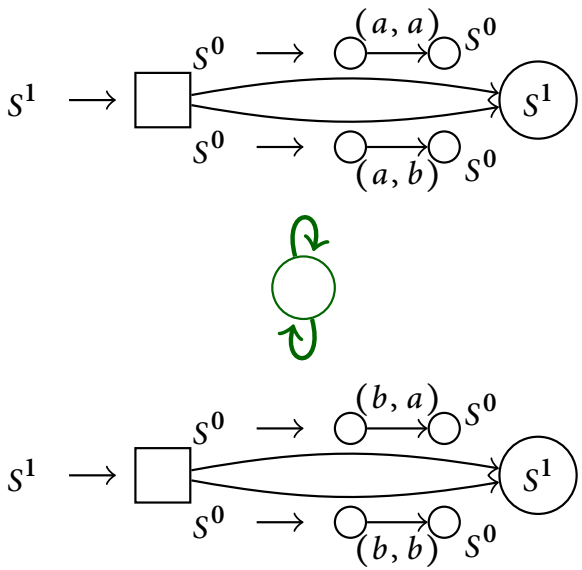
Solution: build the word $x_0 \otimes y_0$, **the Verifier picks** x_0, **the Falsifier** y_0.

Application: FO on Automatic Structures

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

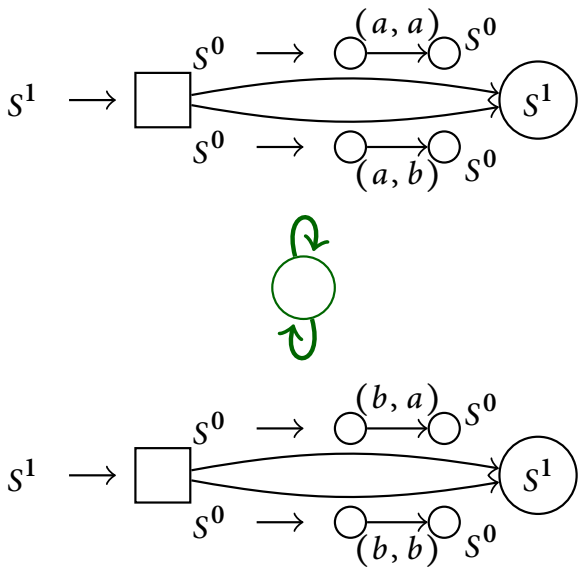


Application: FO on Automatic Structures

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



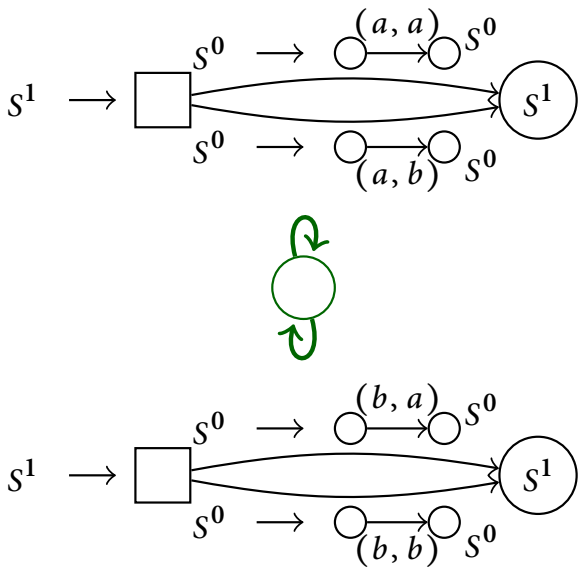
Constructed game

Application: FO on Automatic Structures

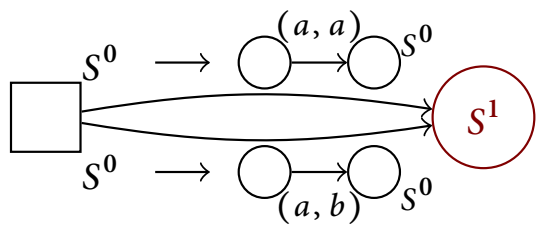
Problem: $\exists x\ \forall y\ R(x, y)$ with an automaton ($\rightsquigarrow$ MSO formula) recognizing R

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

Higher-order game



Constructed game

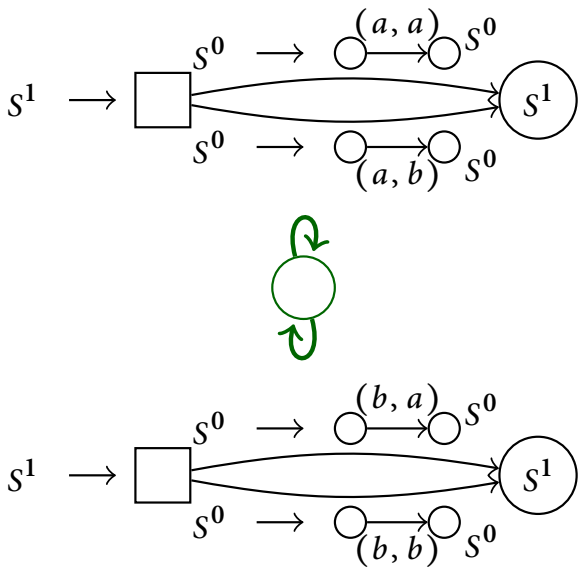


Application: FO on Automatic Structures

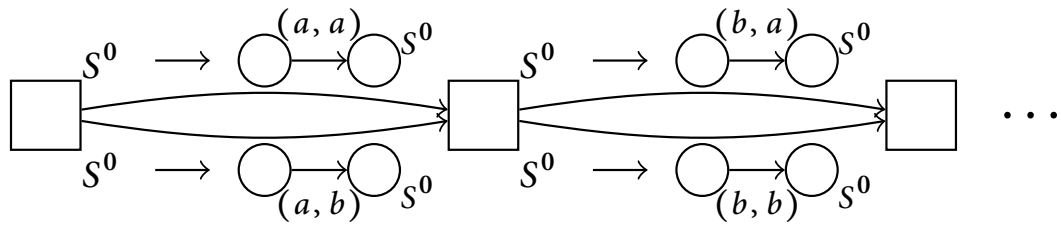
Problem: $\exists x\ \forall y\ R(x, y)$ with an automaton ($\rightsquigarrow$ MSO formula) recognizing R

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

Higher-order game



Constructed game

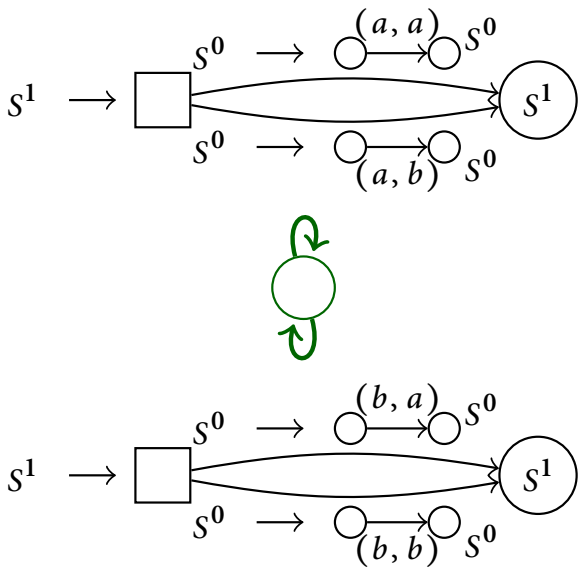


Application: FO on Automatic Structures

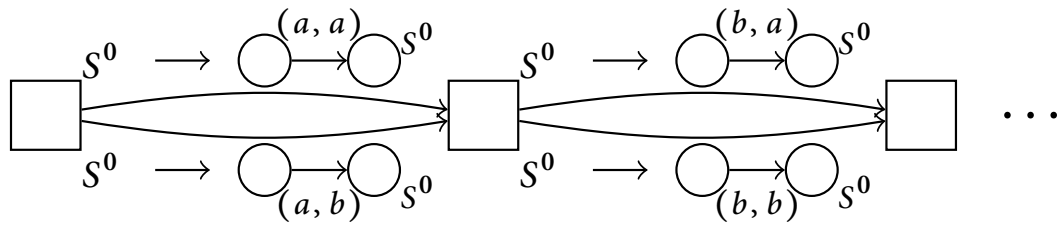
Problem: $\exists x\ \forall y\ R(x, y)$ with an automaton ($\rightsquigarrow$ MSO formula) recognizing R

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

Higher-order game



Constructed game



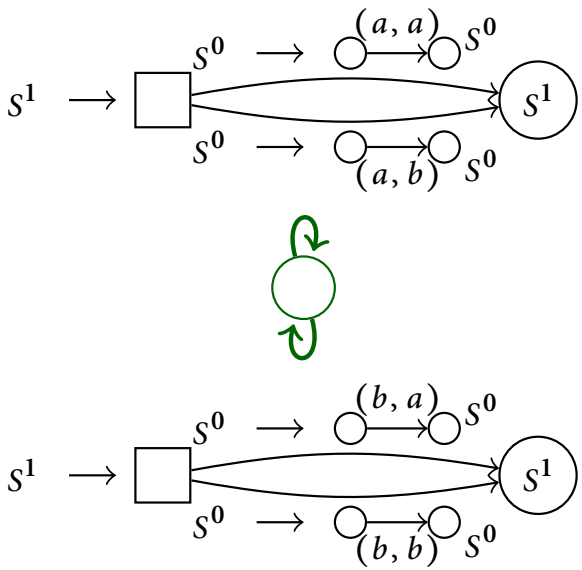
Constructed hypergraph

Application: FO on Automatic Structures

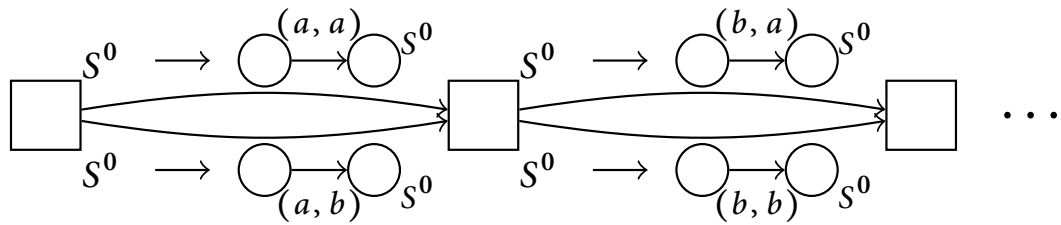
Problem: $\exists x\ \forall y\ R(x, y)$ with an automaton ($\rightsquigarrow$ MSO formula) recognizing R

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

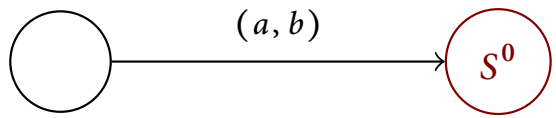
Higher-order game



Constructed game



Constructed hypergraph

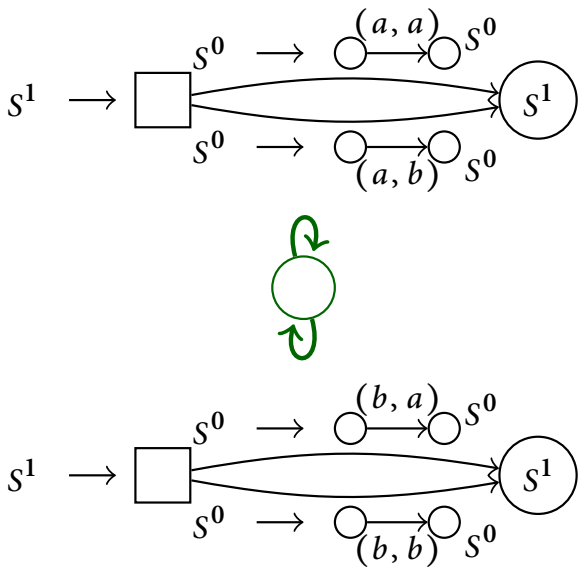


Application: FO on Automatic Structures

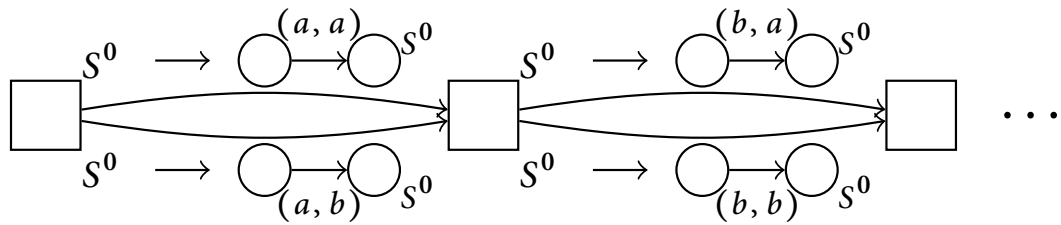
Problem: $\exists x\ \forall y\ R(x, y)$ with an automaton ($\rightsquigarrow$ MSO formula) recognizing R

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

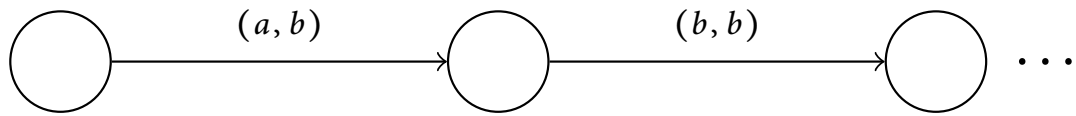
Higher-order game



Constructed game



Constructed hypergraph



Summary

Nice properties of separated handle rewriting games

- Reduce to **ω-regular games** for MSO winning conditions
- Establishing the winner is **decidable**
- Generate known **classes of graphs**, simulate **pushdown games**
- Playing **repeatedly**, e.g. for **model checking** on automatic structures
- Many ways to **generalize** these games

Summary

Nice properties of separated handle rewriting games

- Reduce to ω-regular games for MSO winning conditions
- Establishing the winner is **decidable**
- Generate known **classes of graphs**, simulate **pushdown games**
- Playing **repeatedly**, e.g. for **model checking** on automatic structures
- Many ways to **generalize** these games

Questions

(1) What if players pick **multiple options** for non-terminal symbols?

(2) How about constructing **stochastic** games?

(3) Which rewriting corresponds to **asynchronous product*** ?

* which leads to graphs with decidable FO[R] (T. Colcombet), thanks to C. Löding for asking

Summary

Nice properties of separated handle rewriting games

- Reduce to ω-regular games for MSO winning conditions
- Establishing the winner is **decidable**
- Generate known **classes of graphs**, simulate **pushdown games**
- Playing **repeatedly**, e.g. for **model checking** on automatic structures
- Many ways to **generalize** these games

Questions

(1) What if players pick **multiple options** for non-terminal symbols?

(2) How about constructing **stochastic** games?

(3) Which rewriting corresponds to **asynchronous product*** ?

* which leads to graphs with decidable FO[R] (T. Colcombet), thanks to C. Löding for asking

Thank You