PSPACE-completeness of *Rush Hour* & Modeling Computation as Games

MFoCS Seminar

Markel Zubia Aldaburu
Advisor: Anton Golov

2022

Overview

1. Introduction
2. Rush Hour
   - What is Rush Hour?
   - Components
   - NP-hardness
   - PSPACE-completeness
3. Constraint Logic
   - Description
   - Components
   - PSPACE-completeness

Modeling Computation as Games

- Computation can be modeled with a game.
- Computation may not occur explicitly in a game. It may rather happen when deciding whether certain task can be completed.

Game vs Automata

Games not automata.

- Similarity: a finite set of rules is followed in each steps.
- Difference:
  - Turing machines have infinite tapes, so the state is unbounded.
  - Games are finite, yet they can model any computation.

Complexity

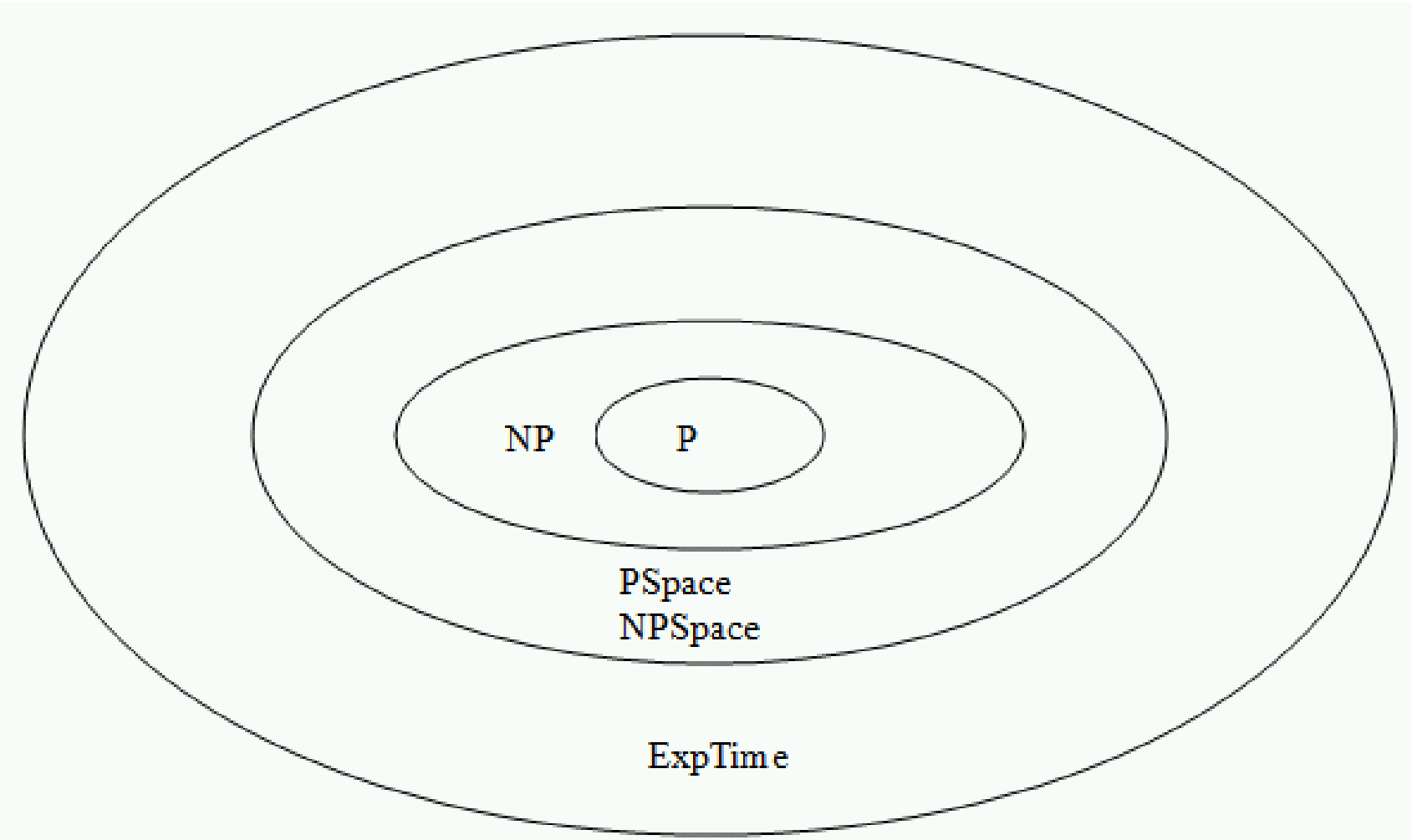


Figure: Complexity hierarchy.

Rush Hour

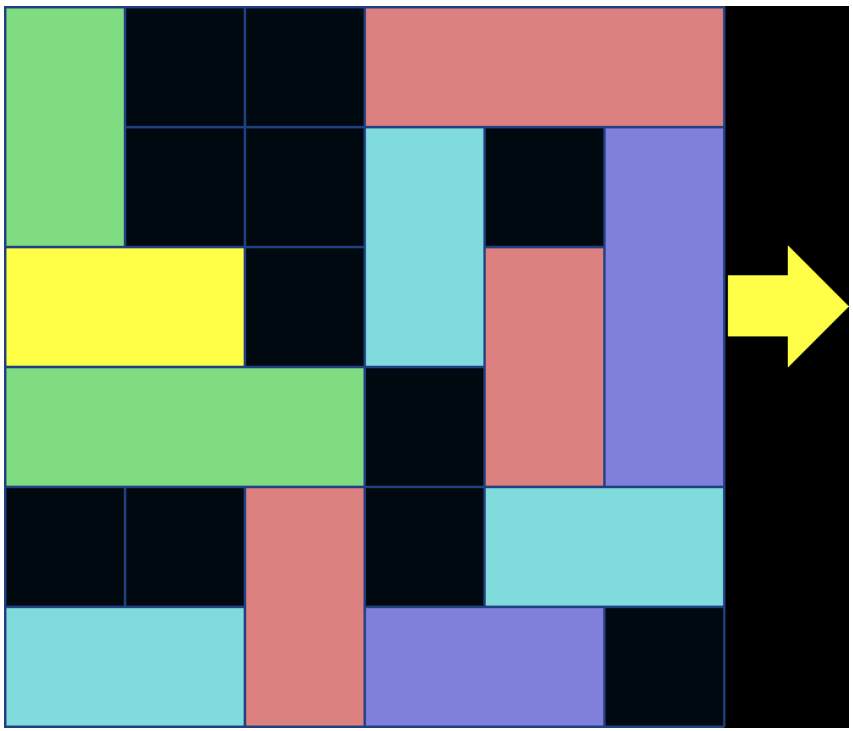


Figure: Instance of rush hour.

An example

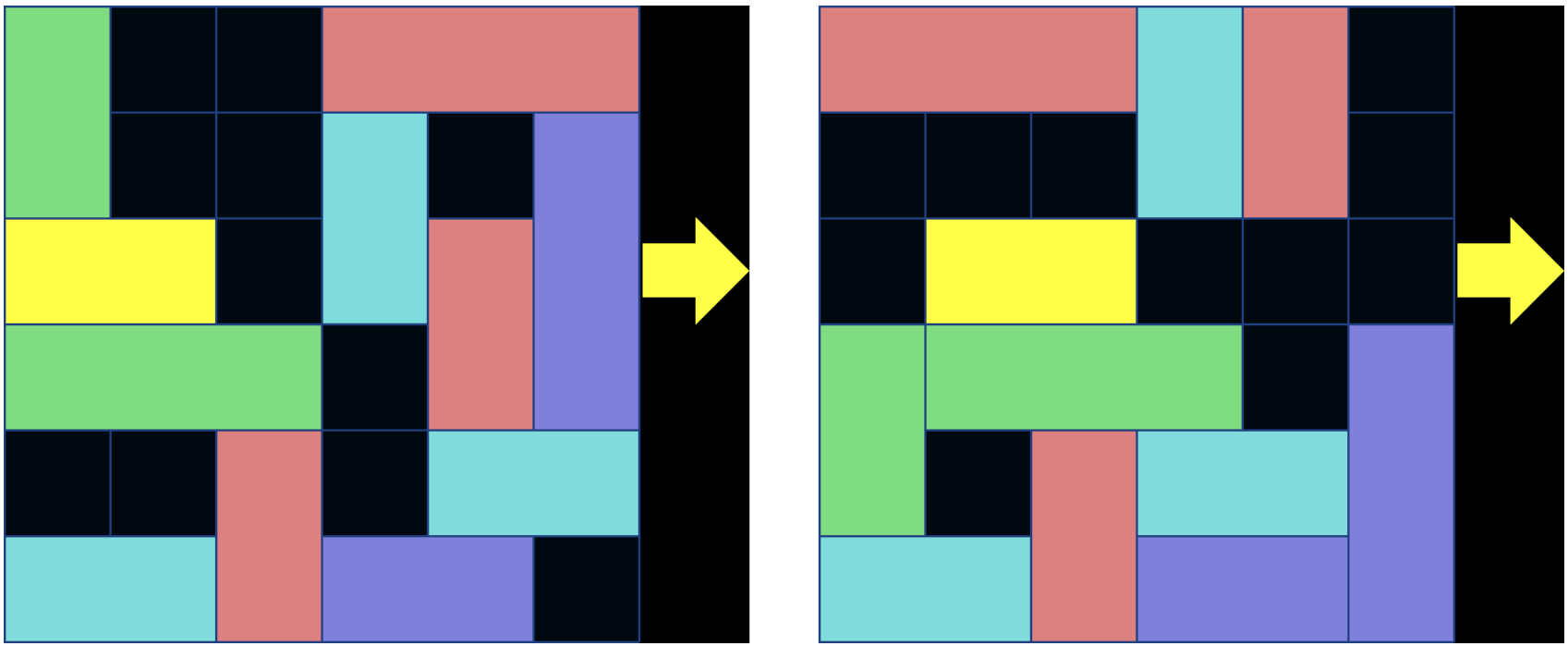


Figure: Example game.

Generalized Rush Hour (GRH)

Two modifications:

1. The grid is a rectangle of arbitrary size.
2. The exit can be at any location on the perimeter.

Intuition

- GRH is in PSPACE.
- Not so clear whether GRH in NP.

Exponential solutions

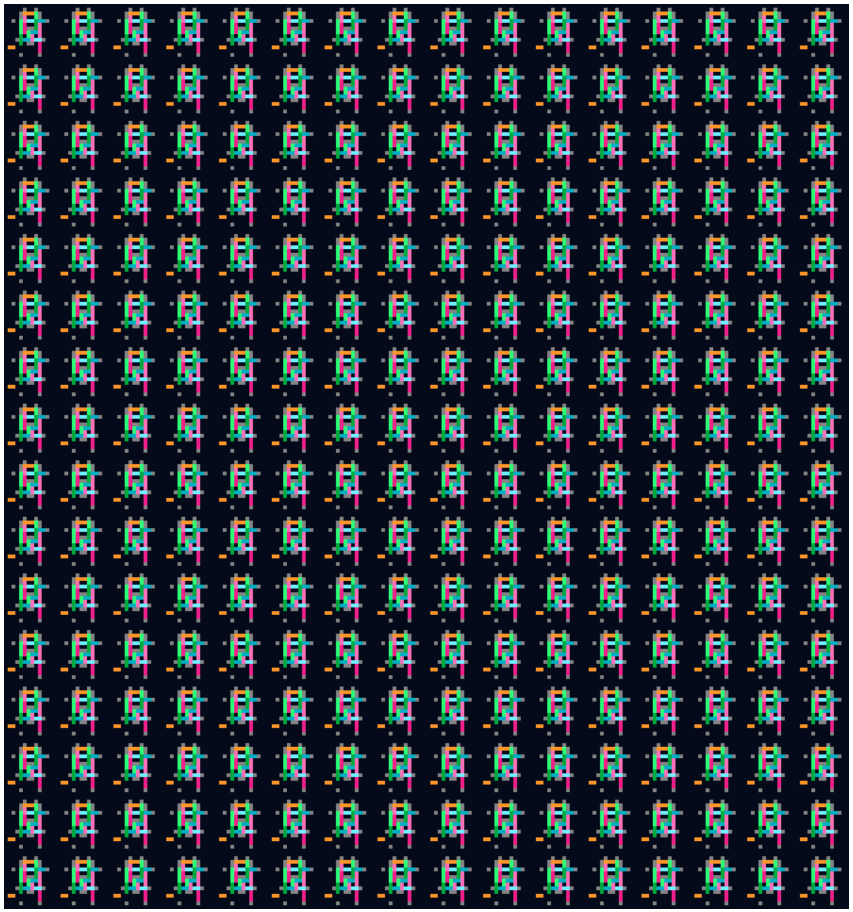


Figure: Solution of 2^{256} steps.

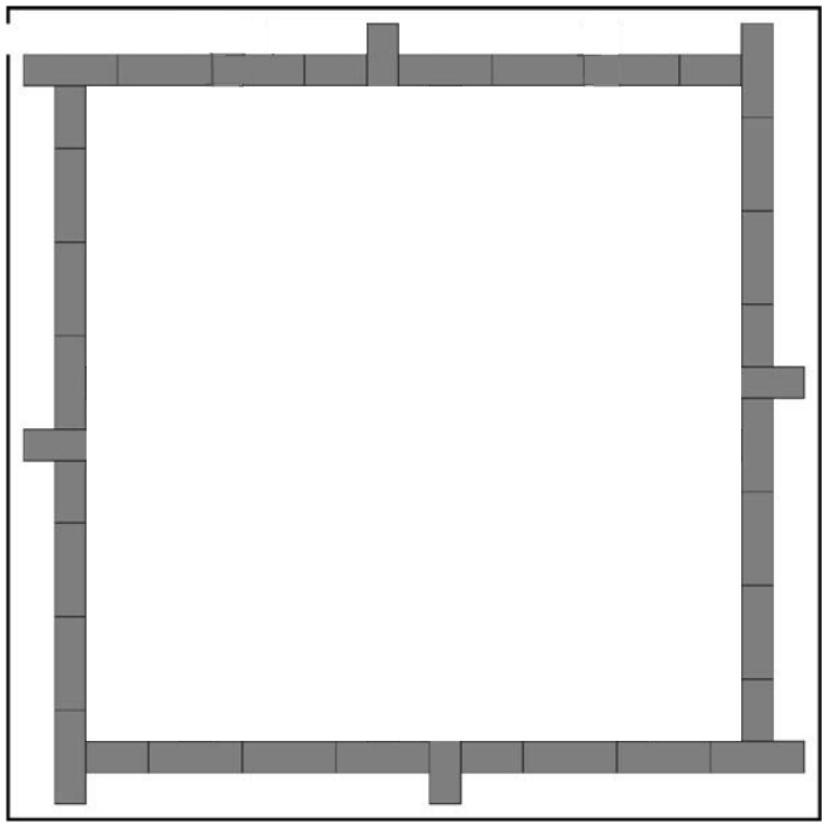
Proofs

Claims[1]:

1. GRH is NP-hard.
2. GRH is PSPACE-complete.

[1] Gary William Flake, Eric B. Baum. Rush Hour is PSPACE-complete, or "Why you should generously tip parking lot attendants", 2002.

Inductive constraints of cells



Crossover block

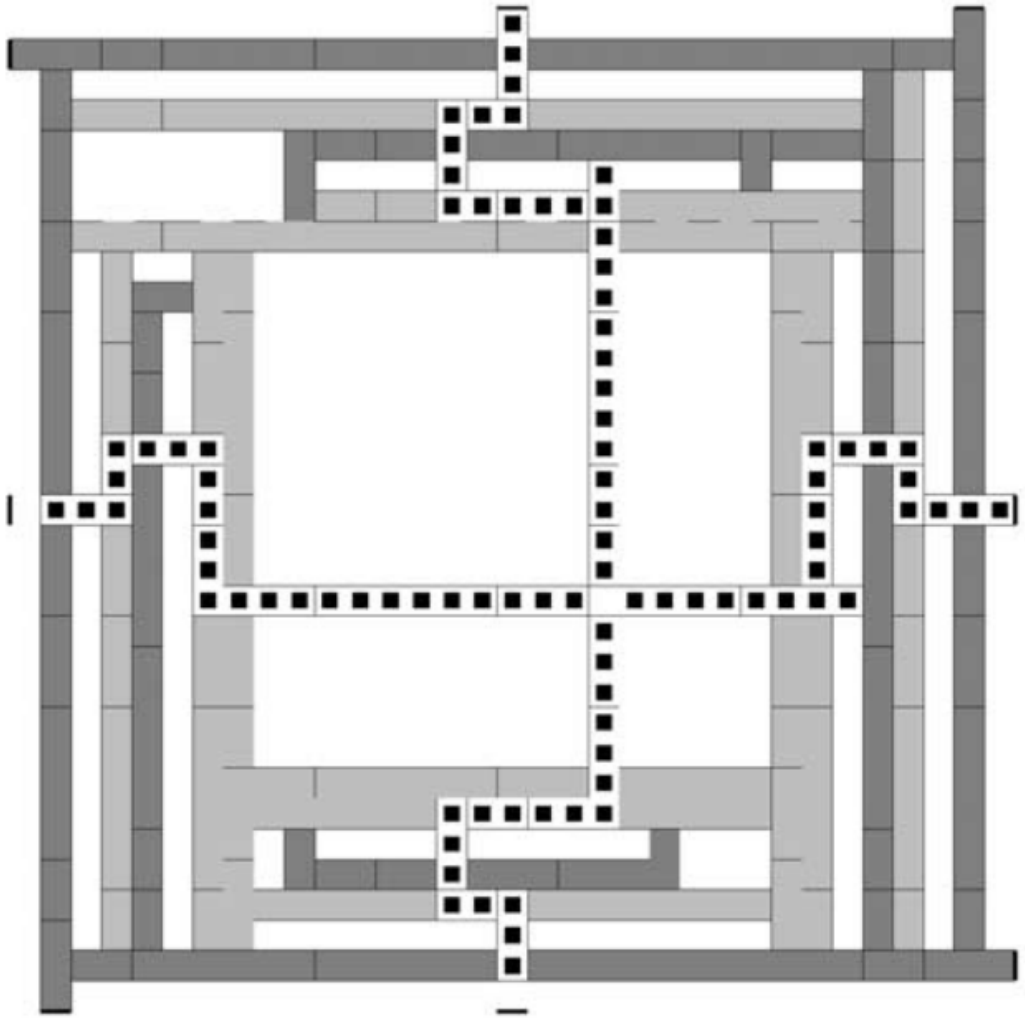


Figure: Crossover block. It allows the intersection of two trigger lines. This lets us have non-plannar circuits.

Both block

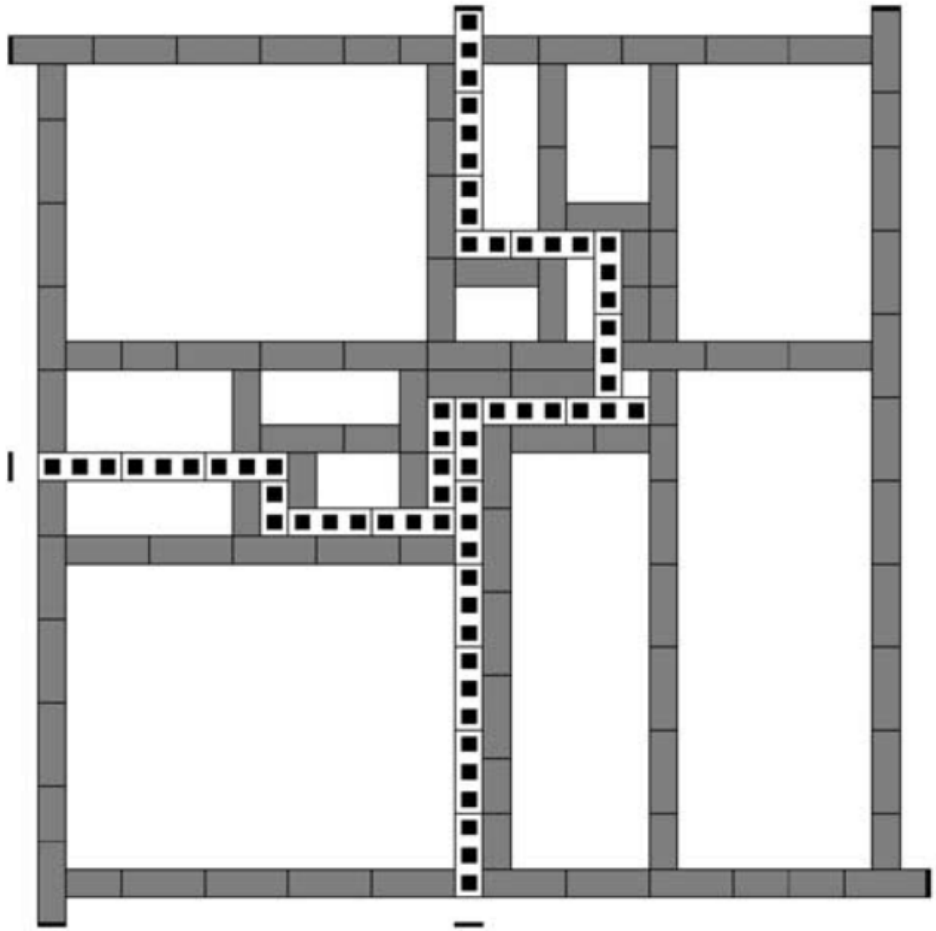


Figure: Both block. Requires both inputs (bottom-left) to be open.

Either block

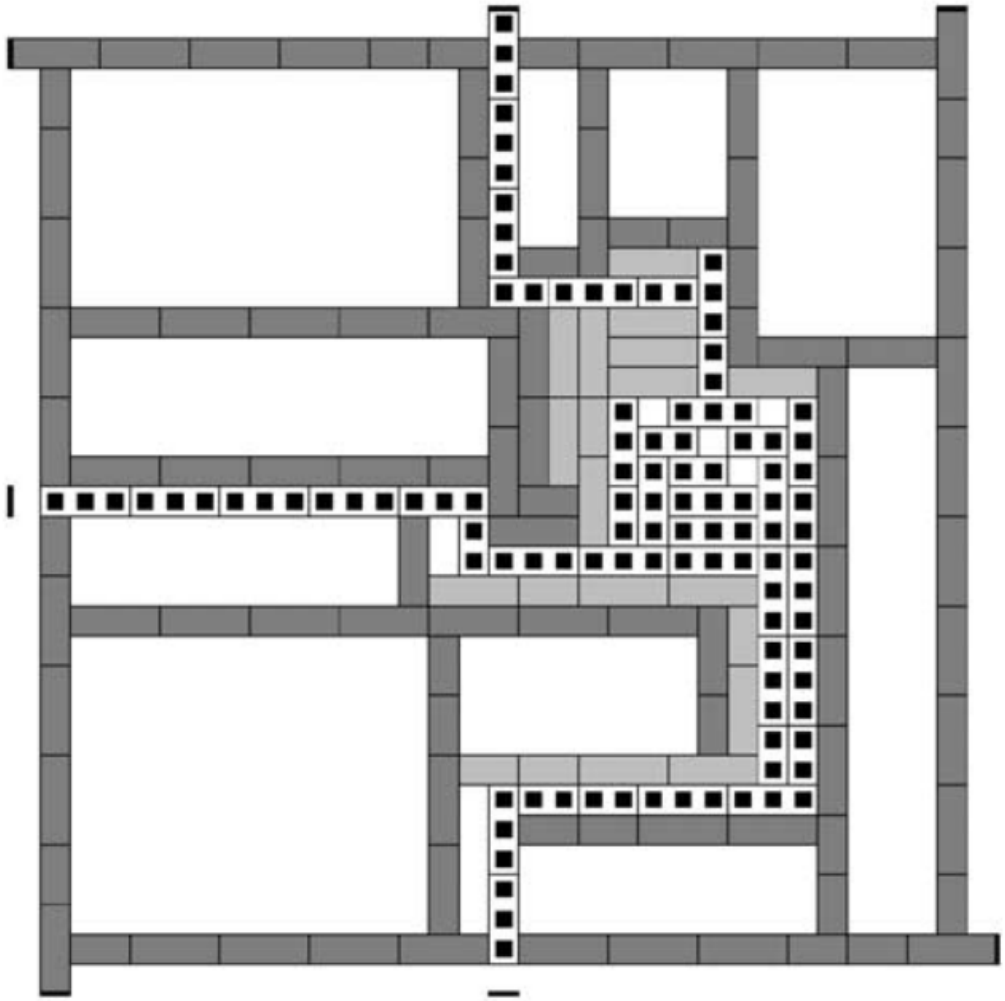


Figure: Either block. Requires either one of the inpputs to be open.

GRH logic

- X^+ and X^- indicate whether X is opened or closed.
- $\dot{\wedge}$ operator represents the output of a both block.

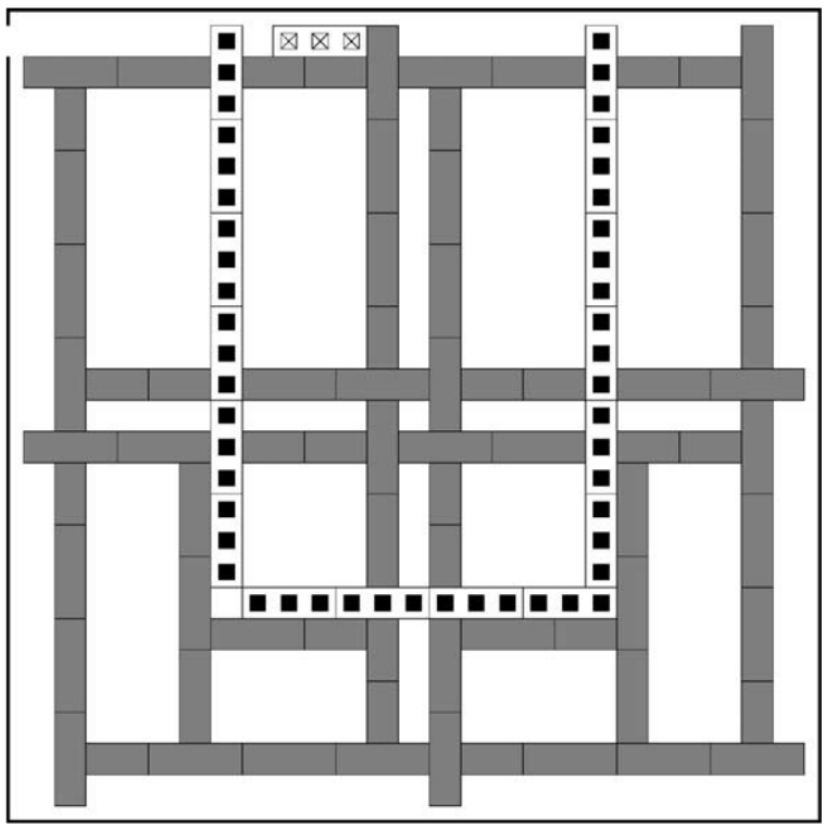
$$Z = X \dot{\wedge} Y \equiv \begin{cases} Z^+ & \text{if } X^+ \wedge Y^+ \\ Z^- & \text{if } X^- \vee Y^- \end{cases}$$

- $\dot{\vee}$ operator represents the output of an either block.

$$Z = X \dot{\vee} Y \equiv \begin{cases} Z^+ & \text{if } X^+ \vee Y^+ \\ Z^- & \text{if } X^- \wedge Y^- \end{cases}$$

GRH logic

- Cannot map open and closed to true and false.
- Cannot have inverter gates.
- Instead map Boolean values to a pair of trigger lines.



GRH logic

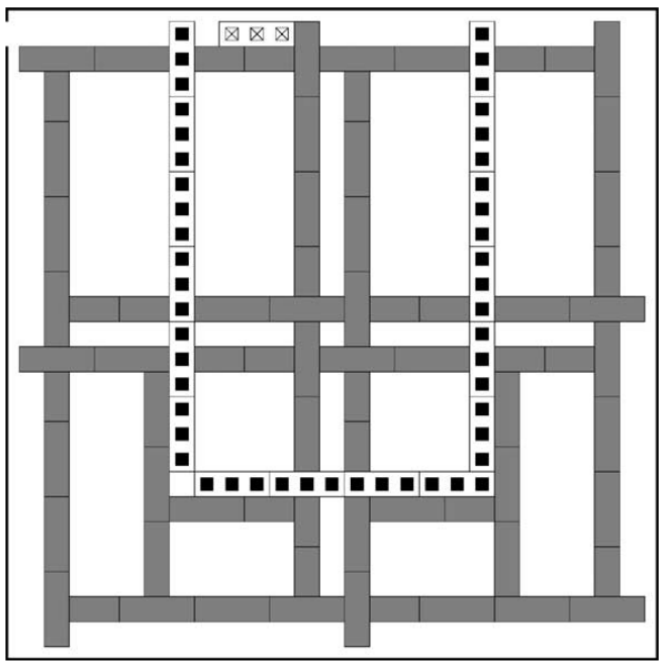
- Dual rail state $\bar{X} = \langle X_T, X_F \rangle$.
- Four possible states:
  - true $\equiv \langle X_T^+, X_F^- \rangle$
  - false $\equiv \langle X_T^-, X_F^+ \rangle$
  - null $\equiv \langle X_T^-, X_F^- \rangle$
  - tautology $\equiv \langle X_T^+, X_F^+ \rangle$
- tautology is prohibited from our constructions.
- null is allowed for transitions between true and false.

GRH logic

- $\bar{X} \bar{\wedge} \bar{Y} \equiv \langle X_T \dot{\wedge} Y_T, X_F \dot{\vee} Y_F \rangle$
- $\bar{X} \bar{\vee} \bar{Y} \equiv \langle X_T \dot{\vee} Y_T, X_F \dot{\wedge} Y_F \rangle$
- $\bar{\neg} \bar{X} \equiv \langle X_F, X_T \rangle$

GRH is NP-hard

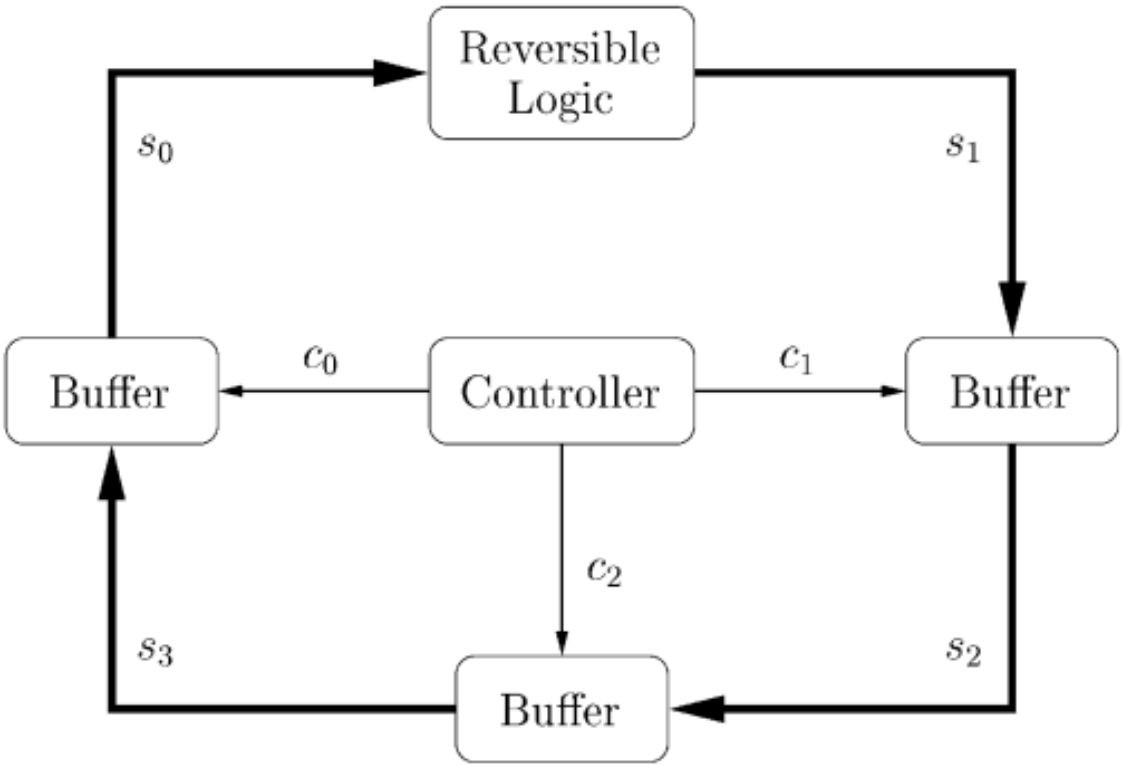
- SAT can be recuced to GRH to prove that GRH is NP-hard.
- Build a GRH configuration for the SAT clause using $\bar{\wedge}, \bar{\vee}, \bar{\neg}$.
- For the variables use the block below.



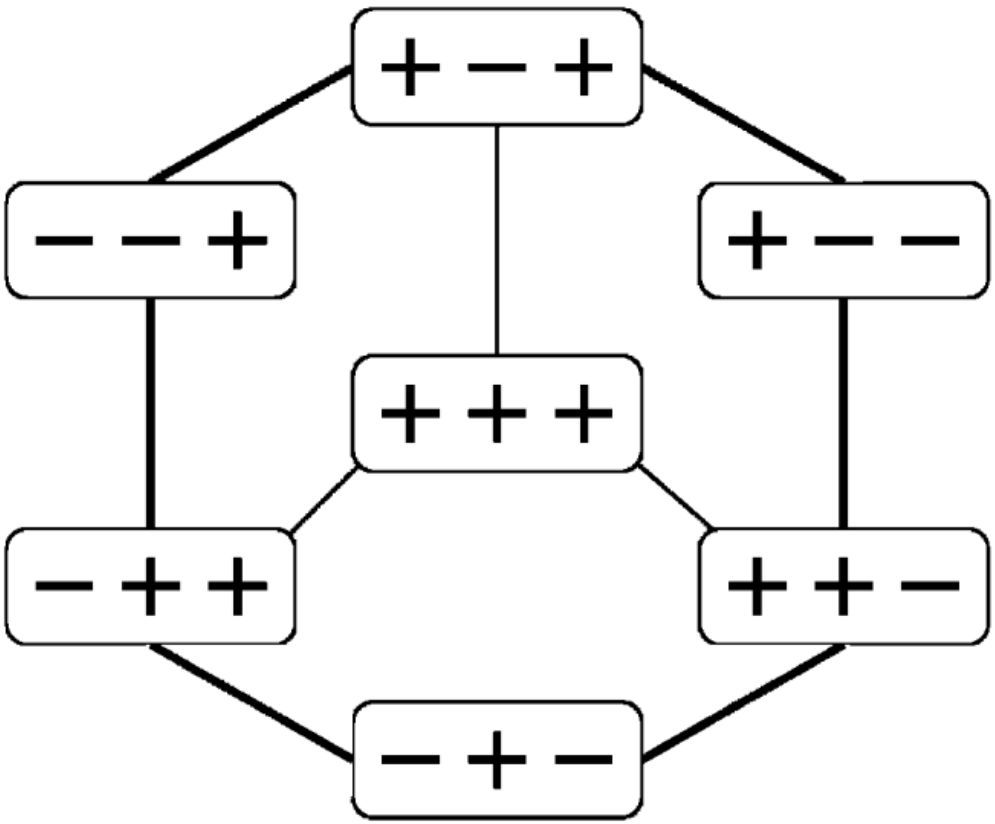
GRH is PSPACE-complete

- Construct a Lazy Reversible Dual-rail machine (LRD).
- LRD can emulate a reversible finite memory random access machine (RAM).
- Equivalent to running arbitrary code with polinomially bounded space.

Lazy Reversible Dual-rail machine



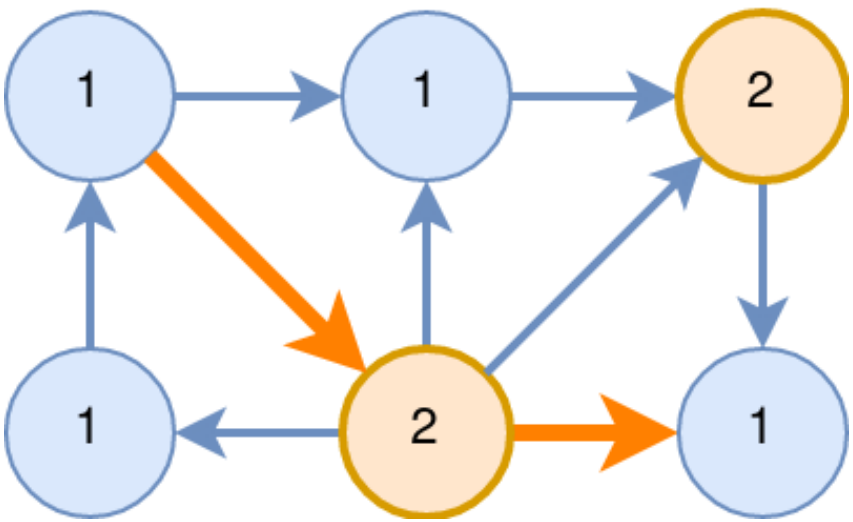
Controller



What is Constraint Logic?

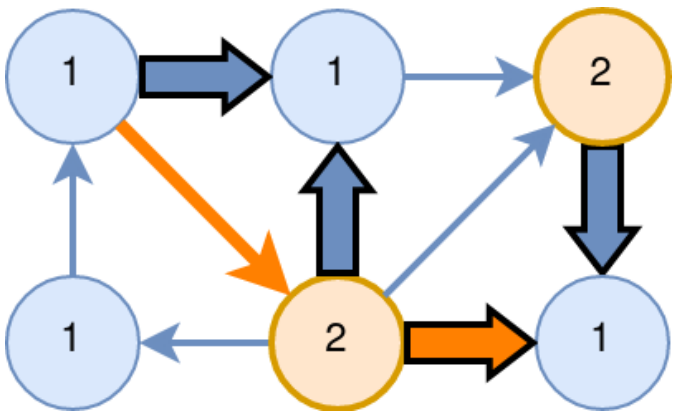
Constraint Logic [2]

- Game family.
- Useful for proving completness of other games.

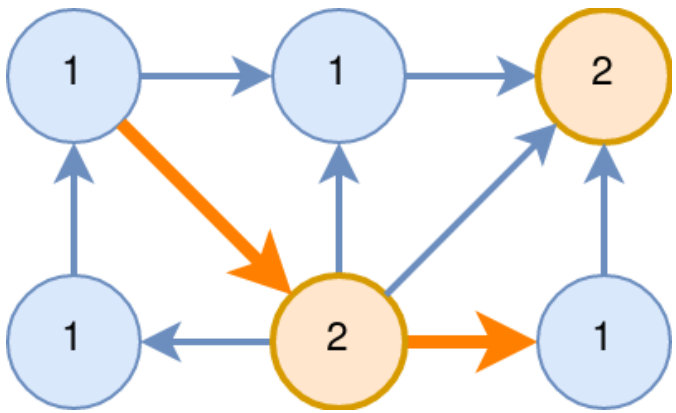


[2] Erik D. Demaine, Robert A. Hearn. Constraint Logic: A Uniform Framework for Modeling Computation as Games, 2008.

Rules

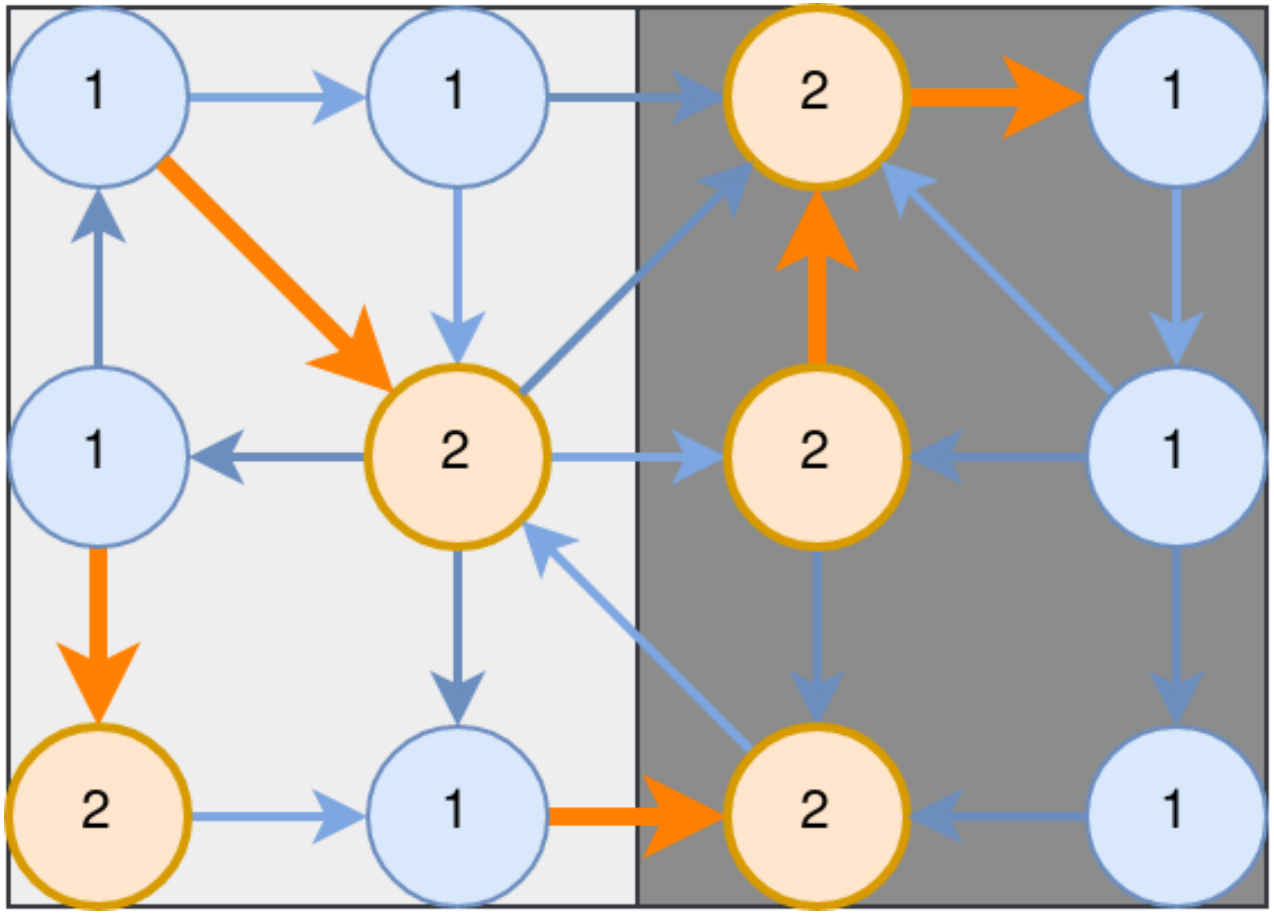


(a) Legal moves

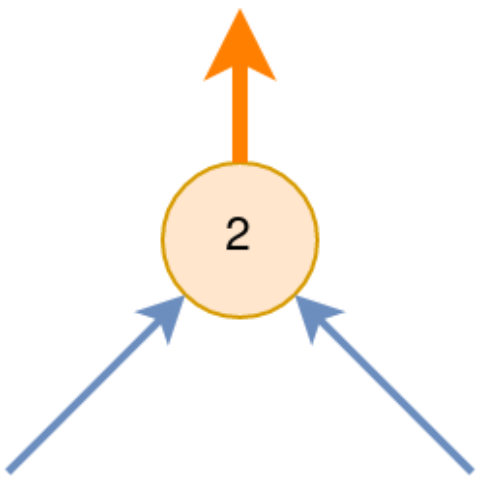


(b) Legal state

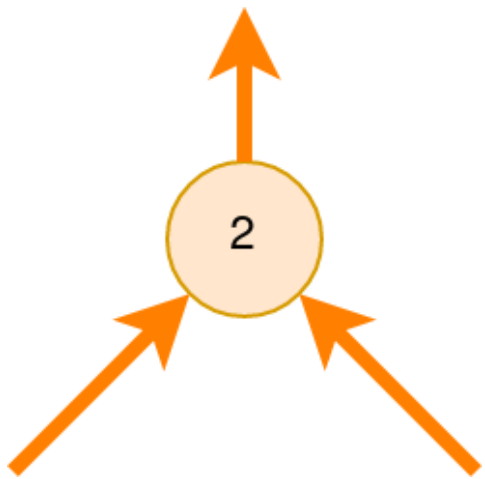
Two player game



Gadgets

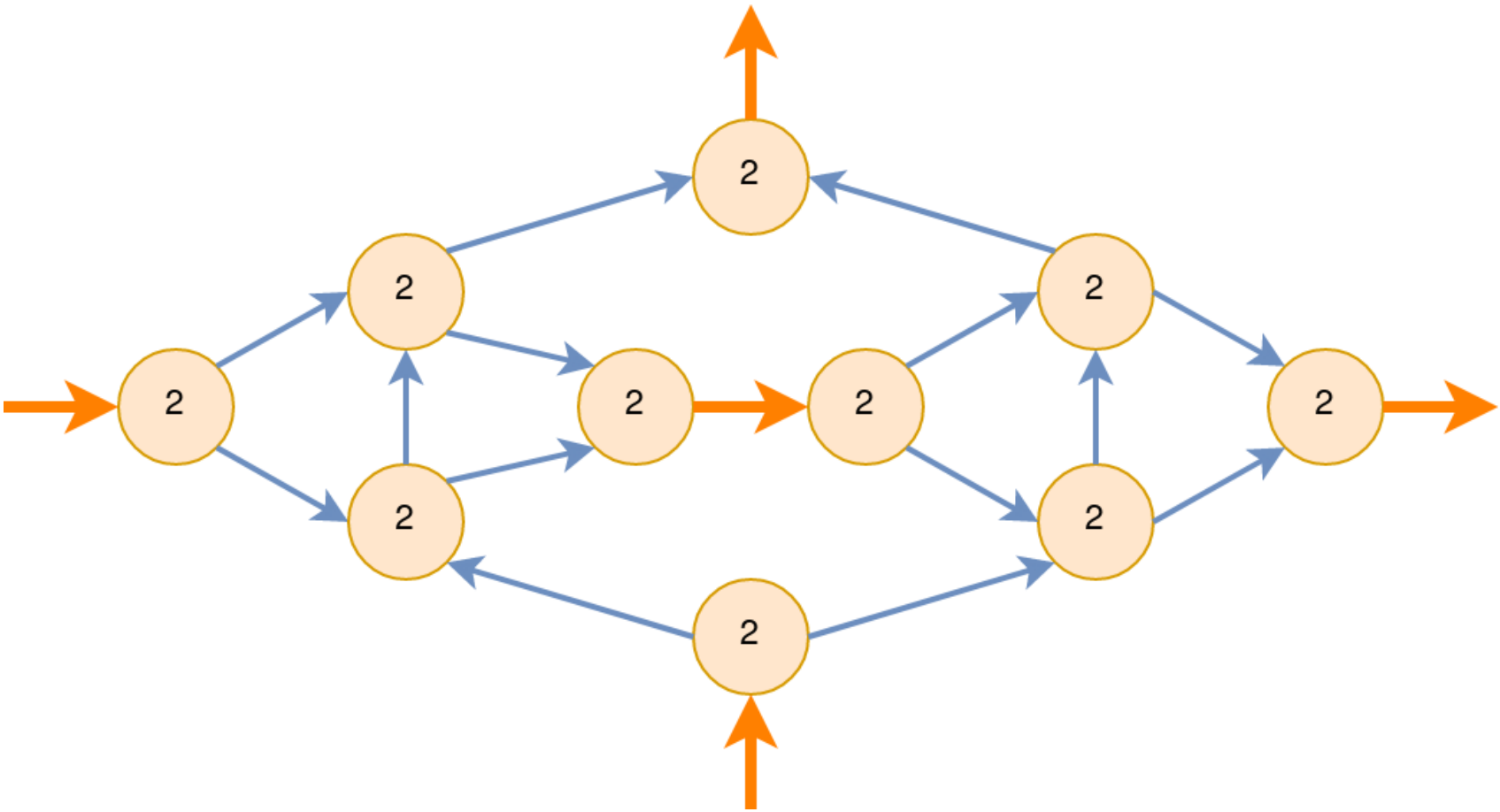


(a) and gadget



(b) or gadget

Crossover gadget



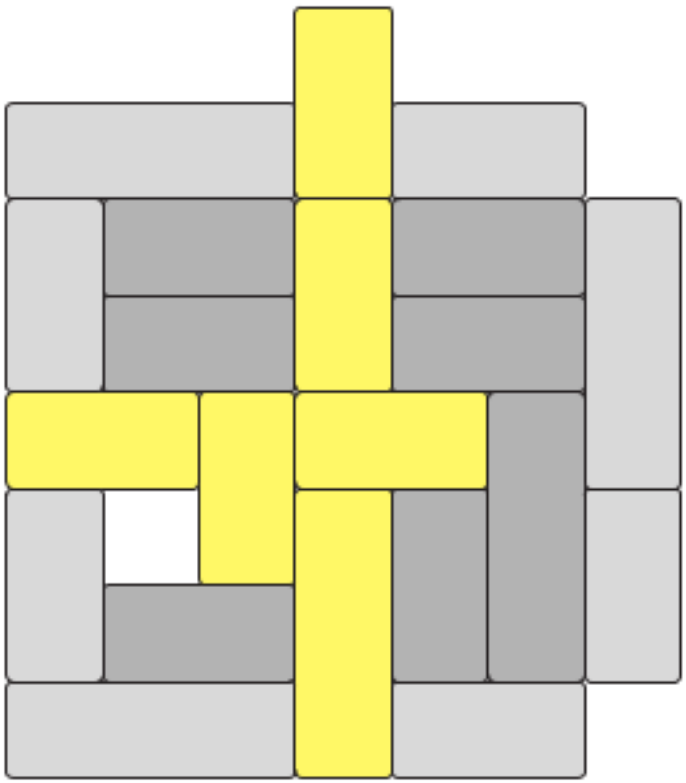
Complexities of different Constraint Logics

	Zero player	*One player*	*Two player*	*Team*
Bounded	P	NP	PSPACE	NEXPTIME
Unbounded	PSPACE	PSPACE	EXPTIME	Undecidable

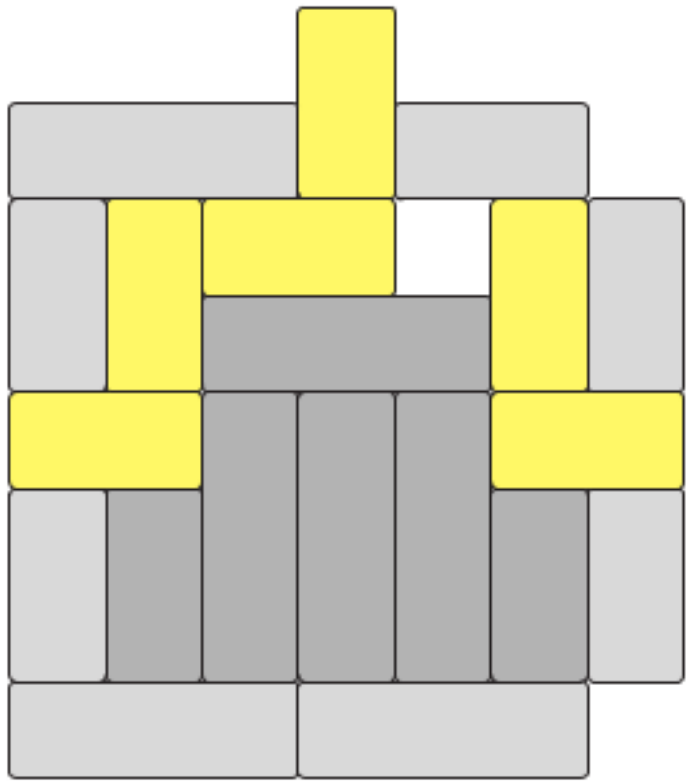
One player unbounded Constraint Logic is PSPACE-complete

- Given constraint graph G, is there a sequence of moves that eventually reverses edge e?
- Build LRD machine as in Rush Hour.
- Problem is PSPACE-complete.

Reduction



(a) AND



(b) OR

Thank you!