Complexity Measures in Propositional Resolution

An Overview of Some Results of the DFG Project

Florian Wörz

Universität Ulm

Institute of Theoretical Computer Science
Institute of Artificial Intelligence

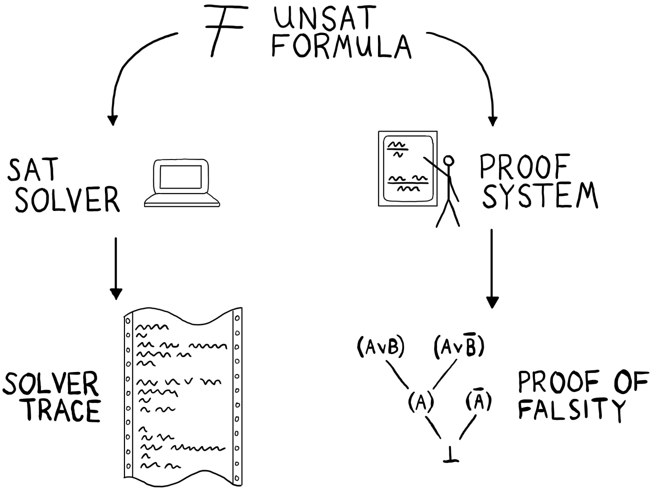
February 14, 2022

Overview

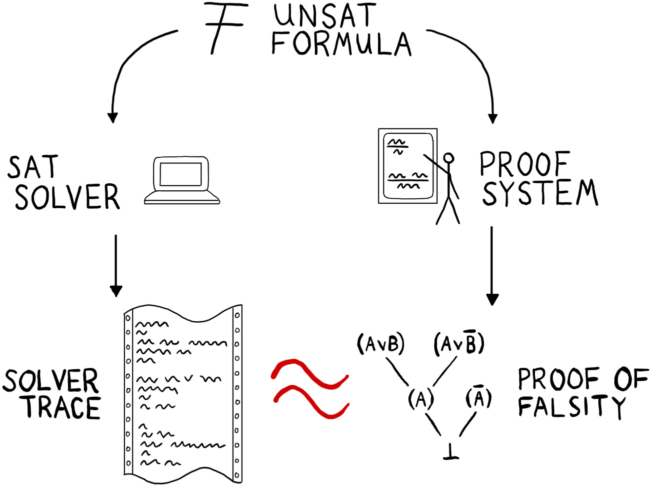
1. Some Basics of Proof Complexity
2. Resolution of Graph Isomorphism Formulas
3. Reversible Pebbling and Resolution Space
4. Interesting Open Research Problems

Some Basics of Proof Complexity

Why Study Proof Complexity?



Why Study Proof Complexity?



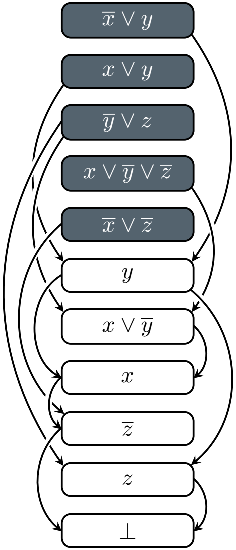
The Proof System Resolution

Resolution Rule:

$$\frac{A \vee x \qquad B \vee \overline{x}}{A \vee B}$$

Distinction by Cases: [Galesi & Thapen]

$$\frac{A_1 \vee \overline{x_1} \quad \dots \quad A_m \vee \overline{x_m}}{B \vee A_1 \vee \cdots \vee A_m} \quad \text{if} \quad (B \vee x_1 \vee \cdots \vee x_m) \in F$$



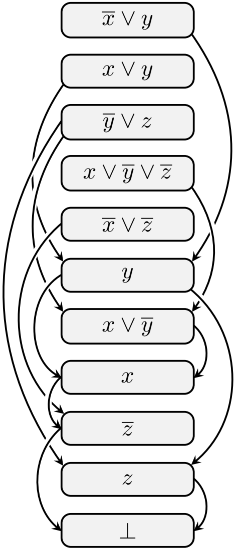
Complexity Measures for Resolution

Size
: # clauses

Width
: # literals in largest clause

Narrow Width
: exclude all axioms

Clause Space
: max # clauses in memory



Complexity Measures for Resolution

Size

clauses (here: 11)

Width

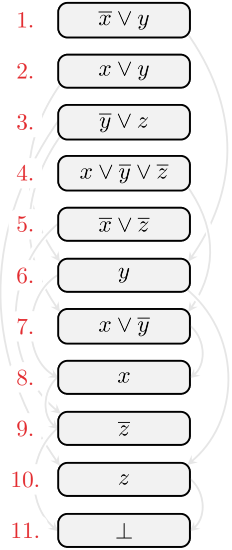
literals in largest clause

Narrow Width

exclude all axioms

Clause Space

max # clauses in memory



Complexity Measures for Resolution

Size

clauses (here: 11)

Width

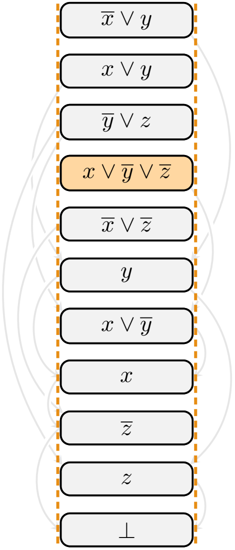
literals in largest clause (here: 3)

Narrow Width

exclude all axioms

Clause Space

max # clauses in memory



Complexity Measures for Resolution

Size

clauses (here: 11)

Width

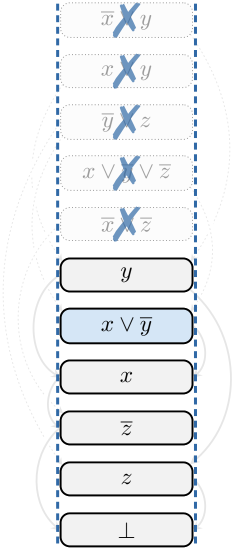
literals in largest clause (here: 3)

Narrow Width

exclude all axioms (here: 2)

Clause Space

max # clauses in memory



Complexity Measures for Resolution

Size

$\#$ clauses (here: 11)

Width

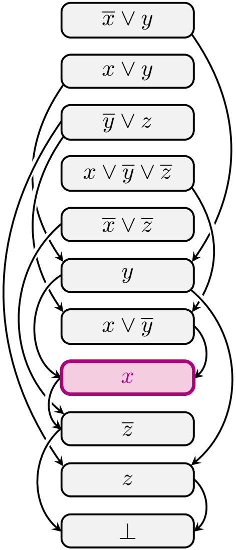
$\#$ literals in largest clause (here: 3)

Narrow Width

exclude all axioms (here: 2)

Clause Space

max $\#$ clauses in memory



Complexity Measures for Resolution

Size

$\#$ clauses (here: 11)

Width

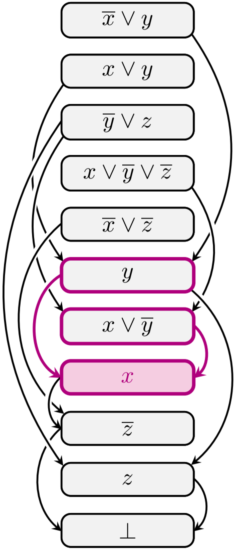
$\#$ literals in largest clause (here: 3)

Narrow Width

exclude all axioms (here: 2)

Clause Space

max $\#$ clauses in memory



Complexity Measures for Resolution

Size

$\#$ clauses (here: 11)

Width

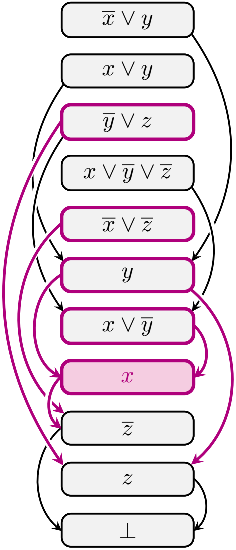
$\#$ literals in largest clause (here: 3)

Narrow Width

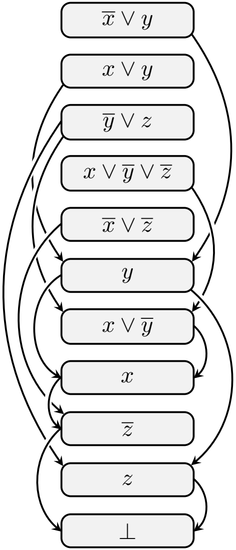
exclude all axioms (here: 2)

Clause Space

max $\#$ clauses in memory (here: 5 at time 8)



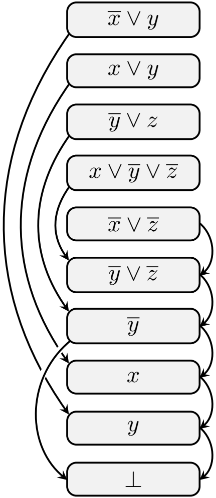
Complexity Measures for Resolution—What we really care about



For each complexity measure $\mathscr{C}$:

Take minimum over all refutations π

$$\mathscr{C}(F \vdash \bot) := \min_{\pi : F \vdash \bot} \mathscr{C}(\pi)$$

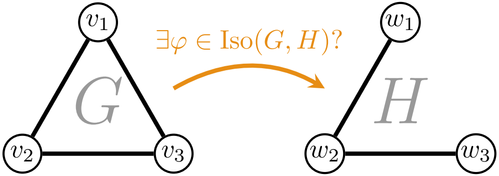


Resolution of Graph Isomorphism Formulas

Computer Science Logic 2022, Göttingen

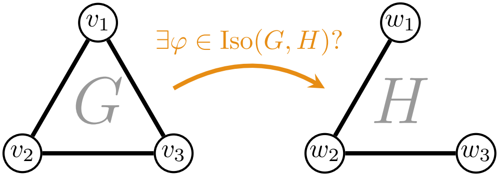
Topic in this Section: Graph Isomorphism Formulas

Take the Graph Isomorphism Problem...



Topic in this Section: Graph Isomorphism Formulas

Take the Graph Isomorphism Problem…



…and encode it as the formula $\mathrm{ISO}(G,H)$:

- **Type 1 clauses:** consider all vertices

$$\forall i \in [n] : (x_{i,1} \vee x_{i,2} \vee \cdots \vee x_{i,n})$$
$$\forall j \in [n] : (x_{1,j} \vee x_{2,j} \vee \cdots \vee x_{n,j})$$

- **Type 2 clauses:** function + injective

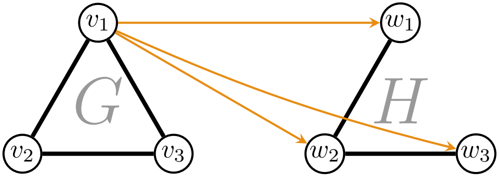
$$\forall i,j,k \in [n] \text{ with } j \neq k : (\overline{x_{i,j}} \vee \overline{x_{i,k}})$$
$$\forall i,j,k \in [n] \text{ with } i \neq j : (\overline{x_{i,k}} \vee \overline{x_{j,k}})$$

- **Type 3 clauses:** adjacency relation

$$\forall i < j \text{ and } k \neq \ell \text{ with}$$
$$\{v_i, v_j\} \in E_G \Leftrightarrow \{v_k, v_\ell\} \notin E_H : (\overline{x_{i,k}} \vee \overline{x_{j,\ell}})$$

Topic in this Section: Graph Isomorphism Formulas

Take the Graph Isomorphism Problem...



...and encode it as the formula $\mathrm{ISO}(G, H)$:

- **Type 1 clauses:** consider all vertices

$$\forall i \in [n] : (x_{i,1} \vee x_{i,2} \vee \cdots \vee x_{i,n})$$
$$\forall j \in [n] : (x_{1,j} \vee x_{2,j} \vee \cdots \vee x_{n,j})$$

- **Type 2 clauses:** function + injective

$$\forall i, j, k \in [n] \text{ with } j \neq k : (\overline{x_{i,j}} \vee \overline{x_{i,k}})$$
$$\forall i, j, k \in [n] \text{ with } i \neq j : (\overline{x_{i,k}} \vee \overline{x_{j,k}})$$

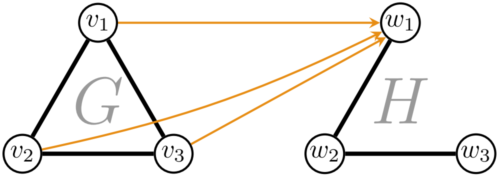
- **Type 3 clauses:** adjacency relation

$\forall i < j$ and $k \neq \ell$ with

$$\{v_i, v_j\} \in E_G \Leftrightarrow \{v_k, v_\ell\} \notin E_H : (\overline{x_{i,k}} \vee \overline{x_{j,\ell}})$$

Topic in this Section: Graph Isomorphism Formulas

Take the Graph Isomorphism Problem...



...and encode it as the formula $\mathrm{ISO}(G, H)$:

- **Type 1 clauses:** consider all vertices

$$\forall i \in [n] : (x_{i,1} \vee x_{i,2} \vee \cdots \vee x_{i,n})$$
$$\forall j \in [n] : (x_{1,j} \vee x_{2,j} \vee \cdots \vee x_{n,j})$$

- **Type 2 clauses:** function + injective

$$\forall i, j, k \in [n] \text{ with } j \neq k : (\overline{x_{i,j}} \vee \overline{x_{i,k}})$$
$$\forall i, j, k \in [n] \text{ with } i \neq j : (\overline{x_{i,k}} \vee \overline{x_{j,k}})$$

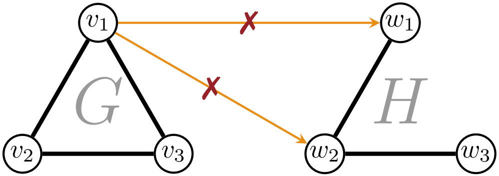
- **Type 3 clauses:** adjacency relation

$\forall i < j$ and $k \neq \ell$ with

$\{v_i, v_j\} \in E_G \Leftrightarrow \{v_k, v_\ell\} \notin E_H : (\overline{x_{i,k}} \vee \overline{x_{j,\ell}})$

Topic in this Section: Graph Isomorphism Formulas

Take the Graph Isomorphism Problem...



...and encode it as the formula $\mathrm{ISO}(G, H)$:

- **Type 1 clauses:** consider all vertices

$$\forall i \in [n] : (x_{i,1} \vee x_{i,2} \vee \cdots \vee x_{i,n})$$
$$\forall j \in [n] : (x_{1,j} \vee x_{2,j} \vee \cdots \vee x_{n,j})$$

- **Type 2 clauses:** function + injective

$$\forall i, j, k \in [n] \text{ with } j \neq k : (\overline{x_{i,j}} \vee \overline{x_{i,k}})$$
$$\forall i, j, k \in [n] \text{ with } i \neq j : (\overline{x_{i,k}} \vee \overline{x_{j,k}})$$

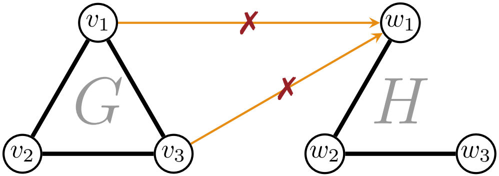
- **Type 3 clauses:** adjacency relation

$\forall i < j$ and $k \neq \ell$ with

$$\{v_i, v_j\} \in E_G \Leftrightarrow \{v_k, v_\ell\} \notin E_H : (\overline{x_{i,k}} \vee \overline{x_{j,\ell}})$$

Topic in this Section: Graph Isomorphism Formulas

Take the Graph Isomorphism Problem...



...and encode it as the formula $\mathrm{ISO}(G, H)$:

- **Type 1 clauses:** consider all vertices

$$\forall i \in [n] : (x_{i,1} \vee x_{i,2} \vee \cdots \vee x_{i,n})$$
$$\forall j \in [n] : (x_{1,j} \vee x_{2,j} \vee \cdots \vee x_{n,j})$$

- **Type 2 clauses:** function + injective

$\forall i, j, k \in [n]$ with $j \neq k : (\overline{x_{i,j}} \vee \overline{x_{i,k}})$

$\forall i, j, k \in [n]$ with $i \neq j : (\overline{x_{i,k}} \vee \overline{x_{j,k}})$

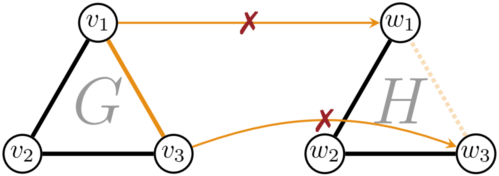
- **Type 3 clauses:** adjacency relation

$\forall i < j$ and $k \neq \ell$ with

$\{v_i, v_j\} \in E_G \Leftrightarrow \{v_k, v_\ell\} \notin E_H : (\overline{x_{i,k}} \vee \overline{x_{j,\ell}})$

Topic in this Section: Graph Isomorphism Formulas

Take the Graph Isomorphism Problem…



…and encode it as the formula $\mathrm{ISO}(G, H)$:

- **Type 1 clauses:** consider all vertices

$$\forall i \in [n] : (x_{i,1} \vee x_{i,2} \vee \cdots \vee x_{i,n})$$
$$\forall j \in [n] : (x_{1,j} \vee x_{2,j} \vee \cdots \vee x_{n,j})$$

- **Type 2 clauses:** function + injective

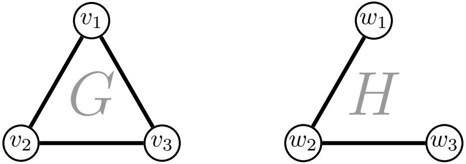
$$\forall i, j, k \in [n] \text{ with } j \neq k : (\overline{x_{i,j}} \vee \overline{x_{i,k}})$$
$$\forall i, j, k \in [n] \text{ with } i \neq j : (\overline{x_{i,k}} \vee \overline{x_{j,k}})$$

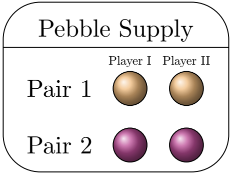
- **Type 3 clauses:** adjacency relation

$\forall i < j$ and $k \neq \ell$ with
$\{v_i, v_j\} \in E_G \Leftrightarrow \{v_k, v_\ell\} \notin E_H : (\overline{x_{i,k}} \vee \overline{x_{j,\ell}})$

Tool From Descriptive Complexity: Immerman's k-pebble game

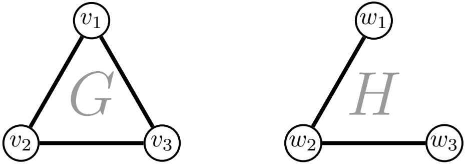
- Player I and Player II have k pebble pairs

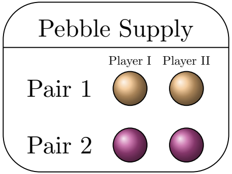




Tool From Descriptive Complexity: Immerman's k-pebble game

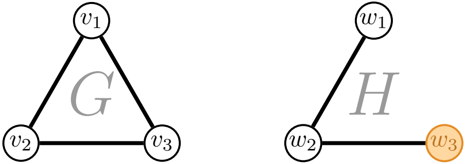
- Player I and Player II have k pebble pairs
- In each round:
  - **Player I chooses:**
    - put a pebble pair back into the box, OR
    - place a new pebble of a pair on any graph

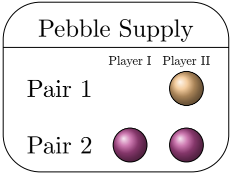




Tool From Descriptive Complexity: Immerman's k-pebble game

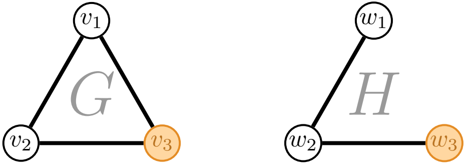
- Player I and Player II have k pebble pairs
- In each round:
  - **Player I chooses:**
    - put a pebble pair back into the box, OR
    - place a new pebble of a pair on any graph

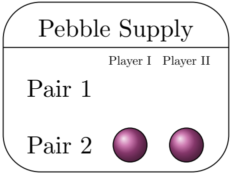




Tool From Descriptive Complexity: Immerman's k-pebble game

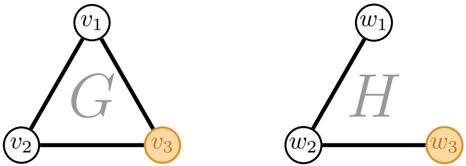
- Player I and Player II have k pebble pairs
- In each round:
  - **Player I chooses:**
    - put a pebble pair back into the box, OR
    - place a new pebble of a pair on any graph
  - **Player II simply reacts.**

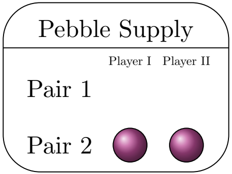




Tool From Descriptive Complexity: Immerman's k-pebble game

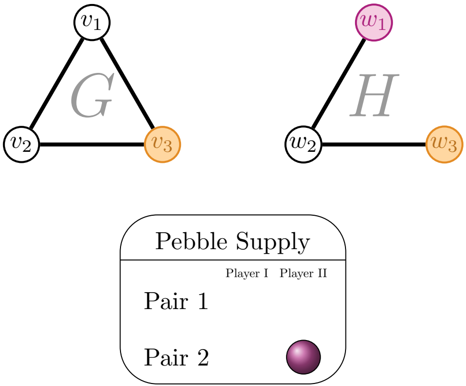
- Player I and Player II have k pebble pairs
- In each round:
  - **Player I chooses:**
    - put a pebble pair back into the box, OR
    - place a new pebble of a pair on any graph
  - **Player II simply reacts.**
- Player II survives if pebbled subgraphs are isomorphic





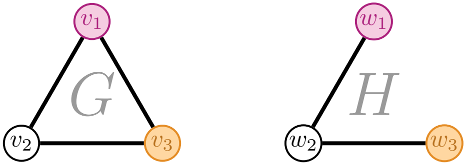
Tool From Descriptive Complexity: Immerman's k-pebble game

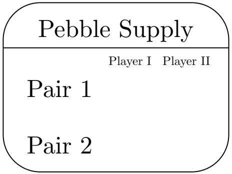
- Player I and Player II have k pebble pairs
- In each round:
  - **Player I chooses:**
    - put a pebble pair back into the box, OR
    - place a new pebble of a pair on any graph
  - **Player II simply reacts.**
- Player II survives if pebbled subgraphs are isomorphic



Tool From Descriptive Complexity: Immerman's k-pebble game

- Player I and Player II have k pebble pairs
- In each round:
  - **Player I chooses:**
    - put a pebble pair back into the box, OR
    - place a new pebble of a pair on any graph
  - **Player II simply reacts.**
- Player II survives if pebbled subgraphs are isomorphic ✗ Player I won!



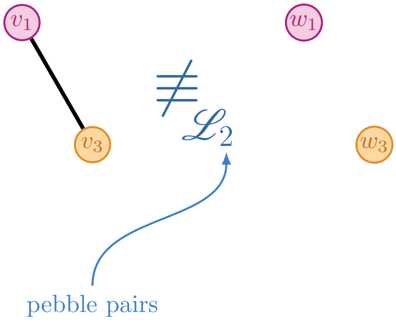


Tool From Descriptive Complexity: Immerman's k-pebble game

- Player I and Player II have k pebble pairs
- In each round:
  - **Player I chooses:**
    - put a pebble pair back into the box, OR
    - place a new pebble of a pair on any graph
  - **Player II simply reacts.**
- Player II survives if pebbled subgraphs are isomorphic ✗ Player I won!

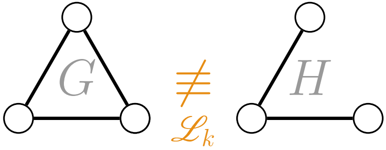
Tool From Descriptive Complexity: Immerman's k-pebble game

- Player I and Player II have k pebble pairs
- In each round:
  - **Player I chooses:**
    - put a pebble pair back into the box, OR
    - place a new pebble of a pair on any graph
  - **Player II simply reacts.**
- Player II survives if pebbled subgraphs are isomorphic ✗ Player I won!



Main Result: Connection between FO and PC

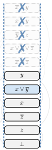
Immerman's Pebble Game
on G and H



$$G \not\equiv_{\mathscr{L}_k} H$$

$$\iff$$

Narrow Width Refutation
of $\mathrm{ISO}(G, H)$



$$\text{N-Width}(\pi) \leq k - 1$$

Implications

For every pair of graphs (G, H) with n vertices each and for every $k \in \mathbb{N}$:

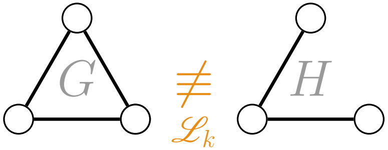
1. $G \not\equiv_{\mathscr{L}_k} H \implies \text{Size}\big(\text{ISO}(G,H) \vdash \bot\big) \leq n^{\mathrm{O}(k)}$

2. $G \equiv_{\mathscr{L}_k} H \implies \begin{cases} \text{Tree-Size}\big(\text{ISO}(G,H) \vdash \bot\big) \geq 2^k \\ \text{CS}\big(\text{ISO}(G,H) \vdash \bot\big) \geq k+1 \end{cases}$

3. $(G,\lambda) \equiv_{\mathscr{L}_k} (H,\mu) \implies \text{Size}\big(\text{ISO}(G,H) \vdash \bot\big) \geq \exp\Big(\Omega\big(\frac{k^2}{\text{sum of color class sizes}}\big)\Big)$

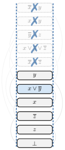
Proof Idea: Use k-witnessing game

Immerman's Pebble Game
on G and H



$\Longleftrightarrow$

Narrow Width Refutation
of $\mathrm{ISO}(G, H)$



$\text{N-Width}(\pi) \leq k - 1$

$\Updownarrow$

k-witnessing game

Spoiler wins on $\mathrm{ISO}(G, H)$

Spoiler vs. Duplicator

They compete in the ***k*-witnessing game** on the formula $\mathrm{ISO}(G, H)$

- Game state is a partial assignment, initially $\alpha_0 = \varepsilon$
- In each round i

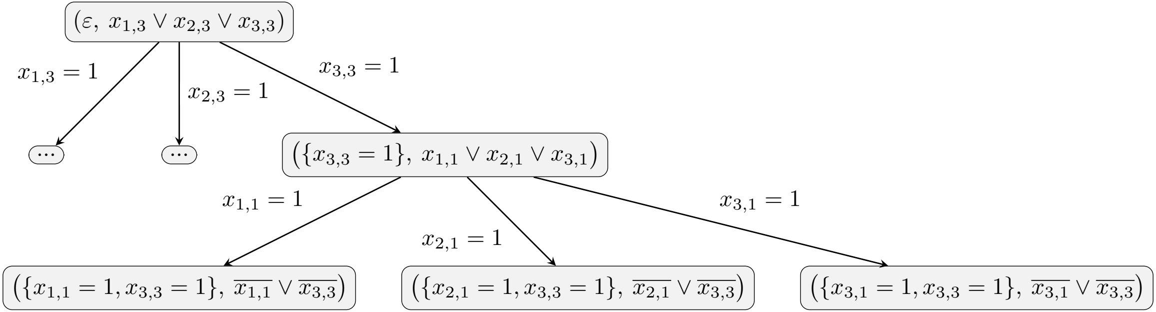
  Spoiler: Chooses a subset $\alpha' \subseteq \alpha_{i-1}$ of size at most $\boldsymbol{k-1}$
  Chooses a Type 1 clause C in $\mathrm{ISO}(G, H)$

  Duplicator: Extends $\alpha_i := \alpha' \cup \{\ell = 1\}$ for some literal $\ell \in C$

- Game ends when Duplicator cannot extend such that
  - α_i satisfies C and
  - does not falsify any other clause in $\mathrm{ISO}(G, H)$

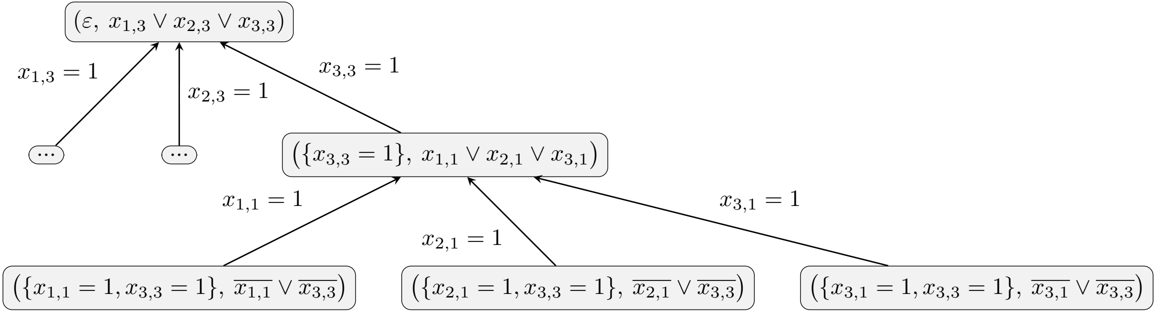
Proof: $G \not\equiv_{\mathscr{L}_k} H \implies \text{N-Width}\big(\text{ISO}(G,H) \vdash \bot\big) \leq k-1$

Convert Strategy Graph of Spoiler into Narrow Width Refutation



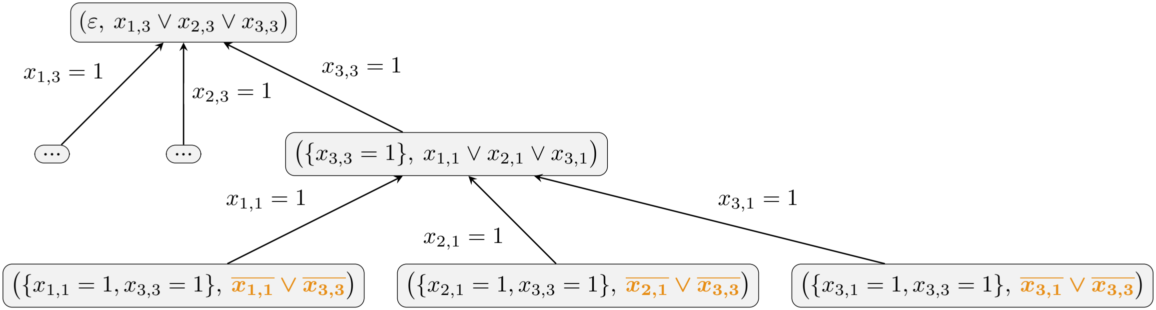
Proof: $G \not\equiv_{\mathscr{L}_k} H \implies \text{N-Width}\big(\text{ISO}(G,H) \vdash \bot\big) \leq k-1$

Convert Strategy Graph of Spoiler into Narrow Width Refutation



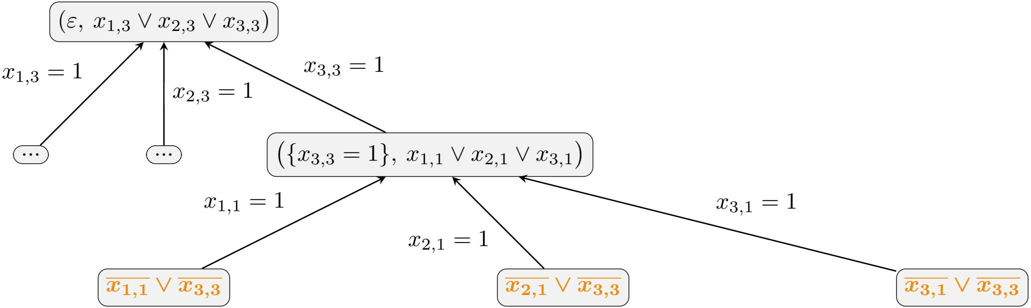
Proof: $G \not\equiv_{\mathscr{L}_k} H \implies \text{N-Width}\big(\text{ISO}(G,H) \vdash \bot\big) \le k-1$

Convert Strategy Graph of Spoiler into Narrow Width Refutation



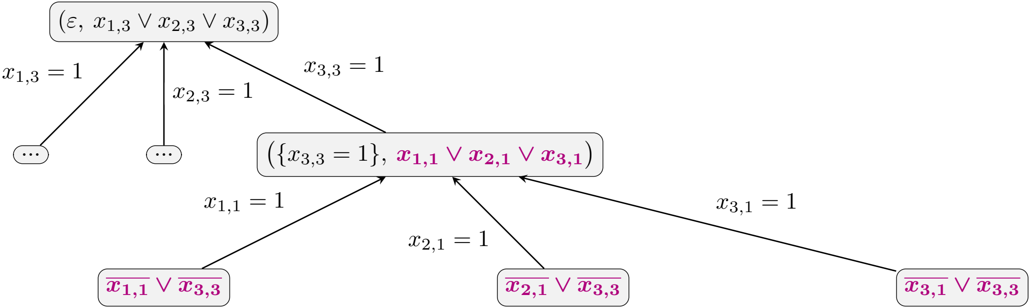
Proof: $G \not\equiv_{\mathscr{L}_k} H \implies \text{N-Width}\big(\text{ISO}(G,H) \vdash \bot\big) \leq k-1$

Convert Strategy Graph of Spoiler into Narrow Width Refutation



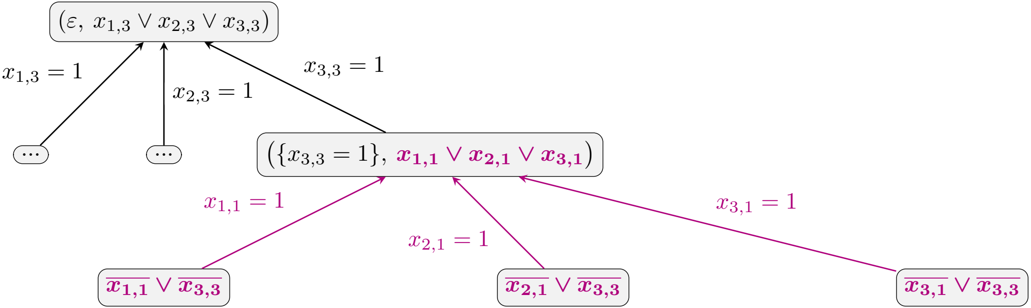
Proof: $G \not\equiv_{\mathscr{L}_k} H \implies \text{N-Width}\big(\text{ISO}(G,H) \vdash \bot\big) \le k-1$

Convert Strategy Graph of Spoiler into Narrow Width Refutation



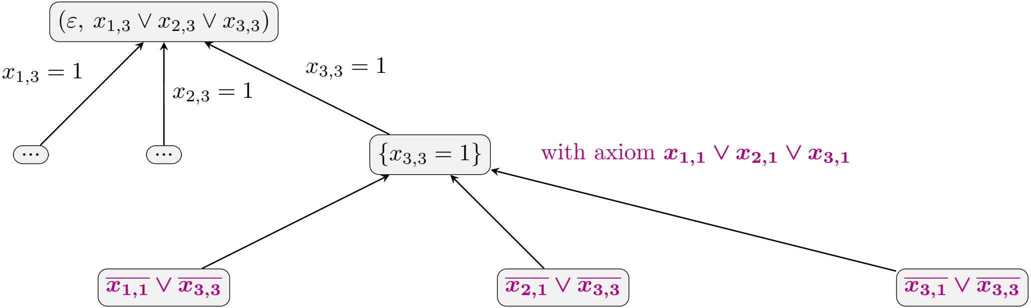
Proof: $G \not\equiv_{\mathscr{L}_k} H \implies \text{N-Width}\big(\text{ISO}(G,H) \vdash \bot\big) \leq k-1$

Convert Strategy Graph of Spoiler into Narrow Width Refutation



Proof: $G \not\equiv_{\mathscr{L}_k} H \Longrightarrow \text{N-Width}\big(\text{ISO}(G,H) \vdash \bot\big) \leq k-1$

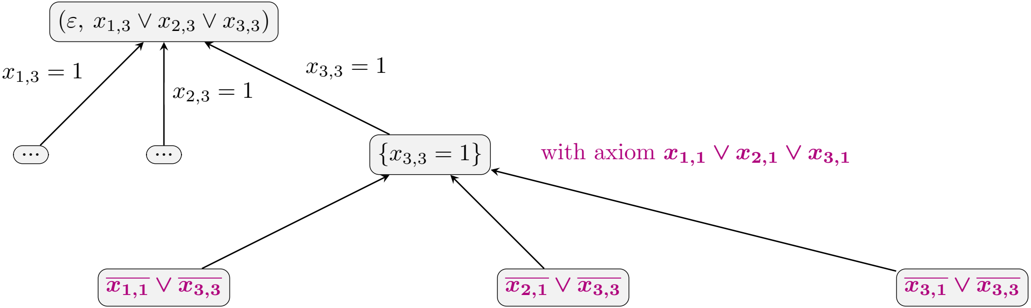
Convert Strategy Graph of Spoiler into Narrow Width Refutation



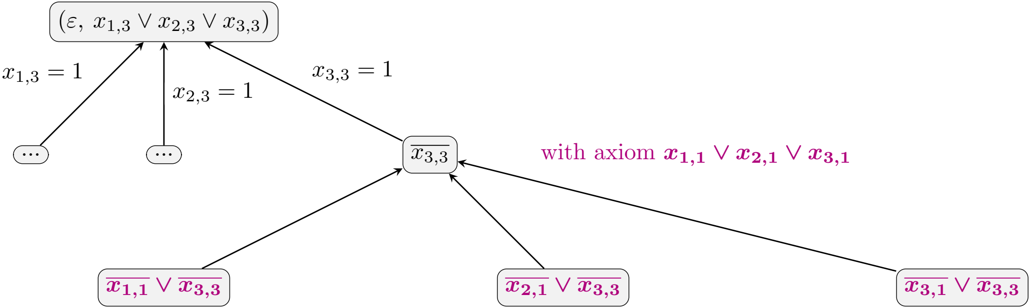
Proof: $G \not\equiv_{\mathscr{L}_k} H \implies \text{N-Width}\big(\text{ISO}(G,H) \vdash \bot\big) \leq k-1$

Convert Strategy Graph of Spoiler into Narrow Width Refutation:

$(\alpha, C) \rightsquigarrow C_\alpha$ (set of literals falsified by α)

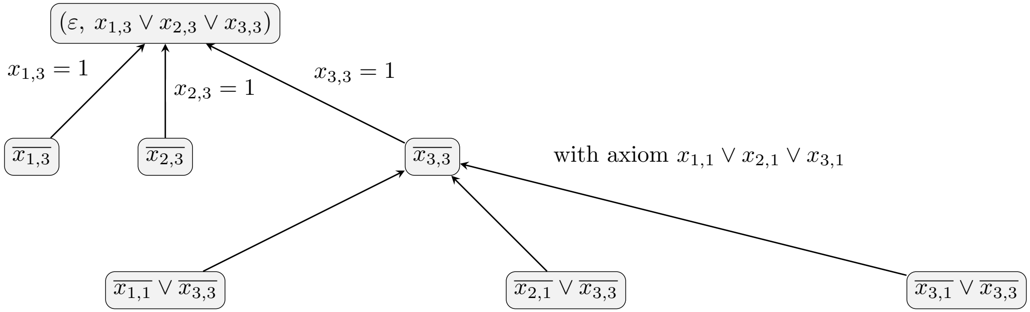


Convert Strategy Graph of Spoiler into Narrow Width Refutation:
$(\alpha, C) \rightsquigarrow C_\alpha$ (set of literals falsified by α)

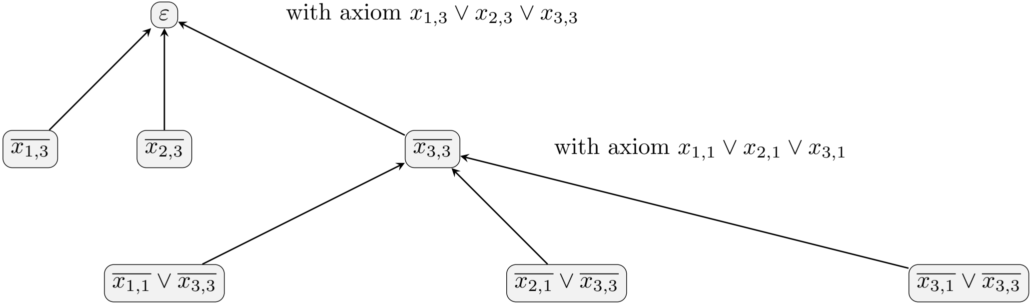


Proof: $G \not\equiv_{\mathscr{L}_k} H \implies \text{N-Width}\big(\text{ISO}(G,H) \vdash \bot\big) \leq k-1$

Convert Strategy Graph of Spoiler into Narrow Width Refutation:
$(\alpha, C) \rightsquigarrow C_\alpha$ (set of literals falsified by α)

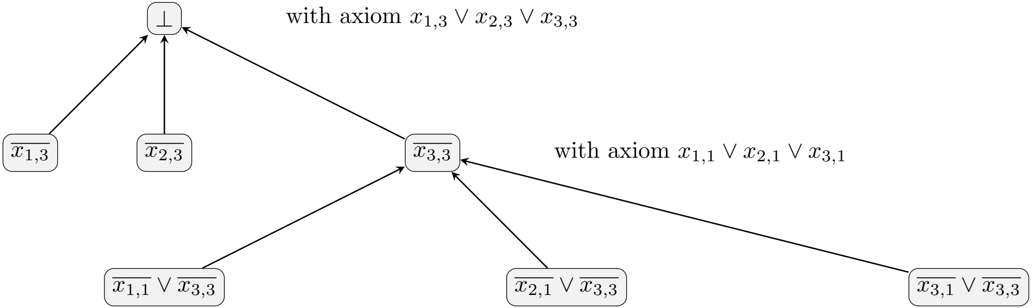


Convert Strategy Graph of Spoiler into Narrow Width Refutation:
$(\alpha, C) \rightsquigarrow C_\alpha$ (set of literals falsified by α)

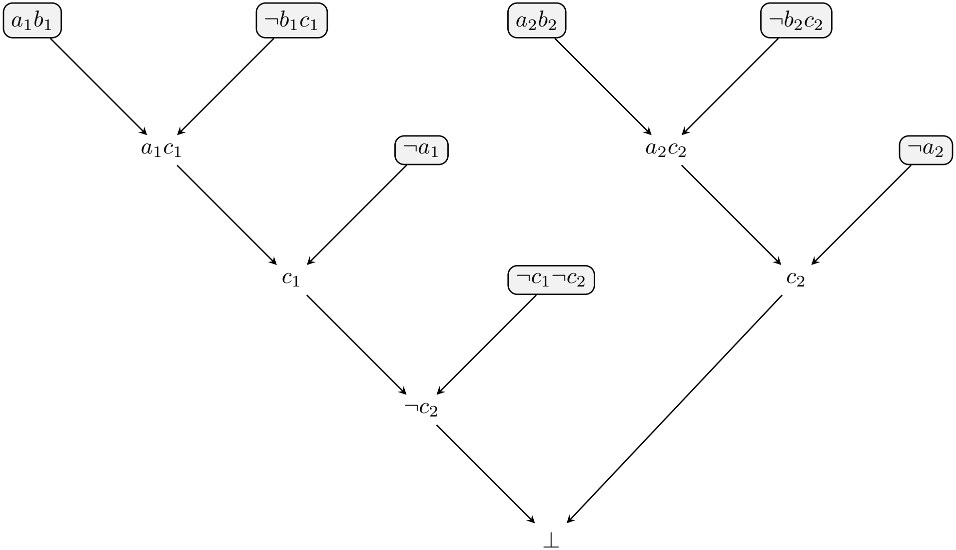


Proof: $G \not\equiv_{\mathscr{L}_k} H \implies \text{N-Width}\big(\text{ISO}(G,H) \vdash \bot\big) \leq k-1$

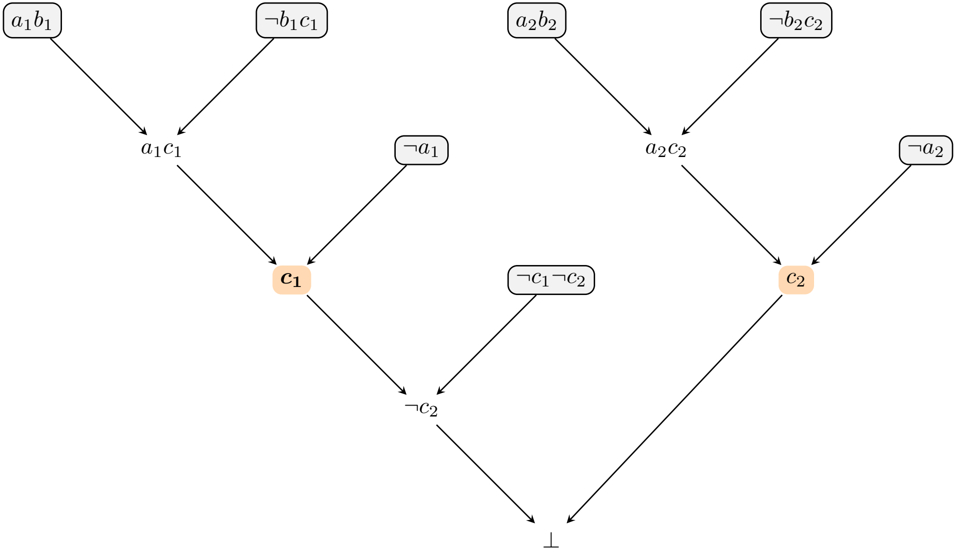
Convert Strategy Graph of Spoiler into Narrow Width Refutation:
$(\alpha, C) \rightsquigarrow C_\alpha$ (set of literals falsified by α)



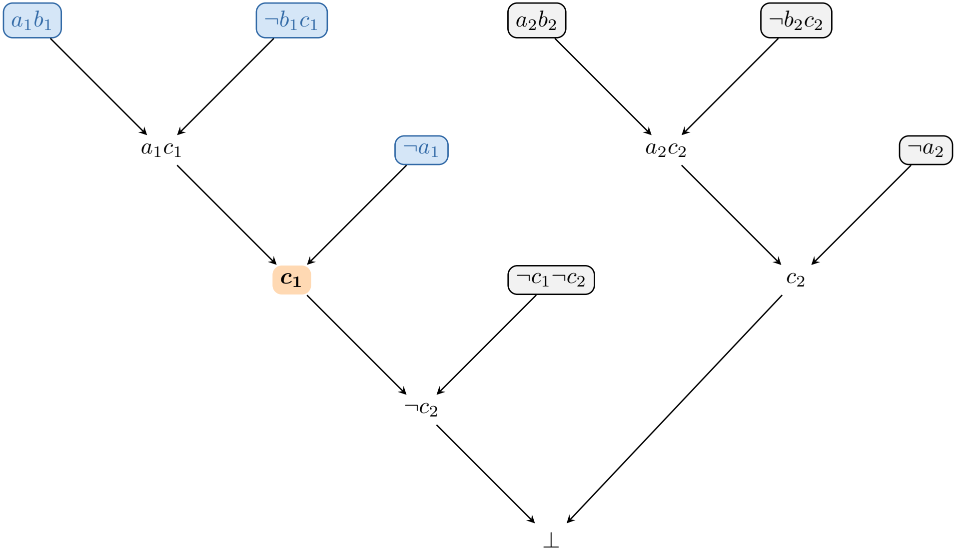
Extending Resolution: Krishnamurthy's Symmetry Rules



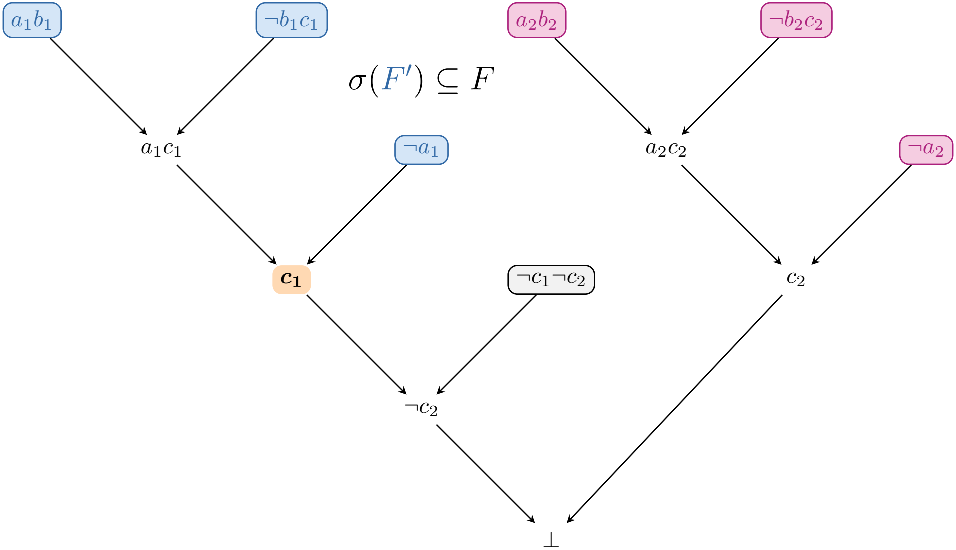
Extending Resolution: Krishnamurthy's Symmetry Rules



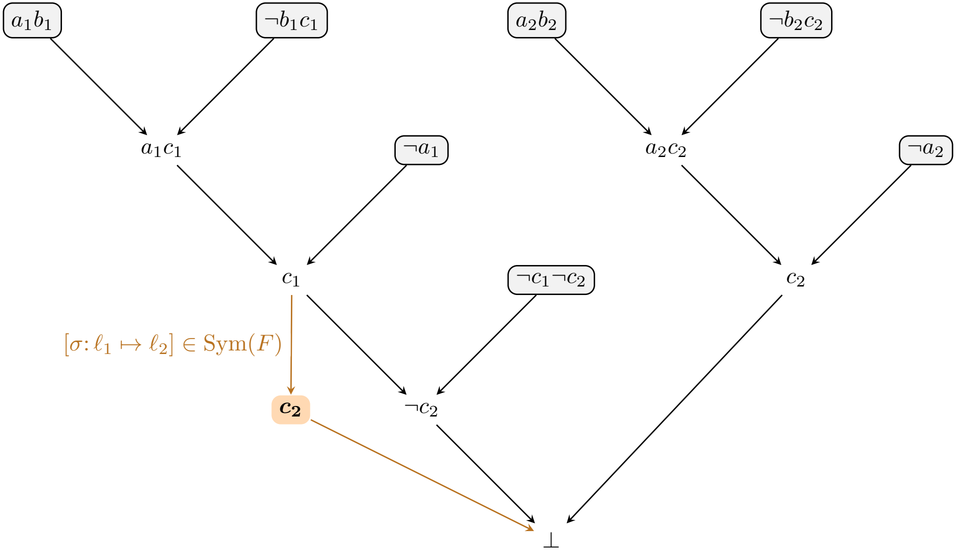
Extending Resolution: Krishnamurthy's Symmetry Rules



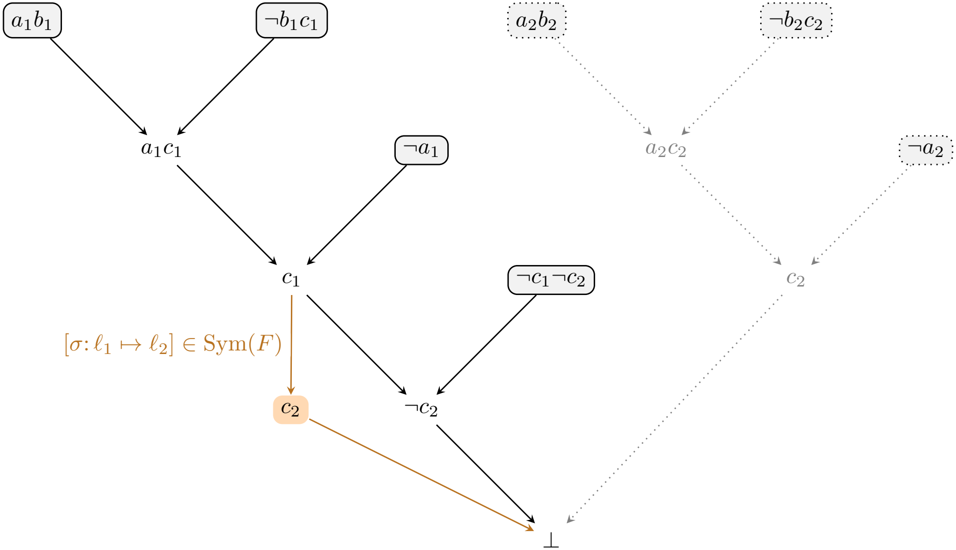
Extending Resolution: Krishnamurthy's Symmetry Rules



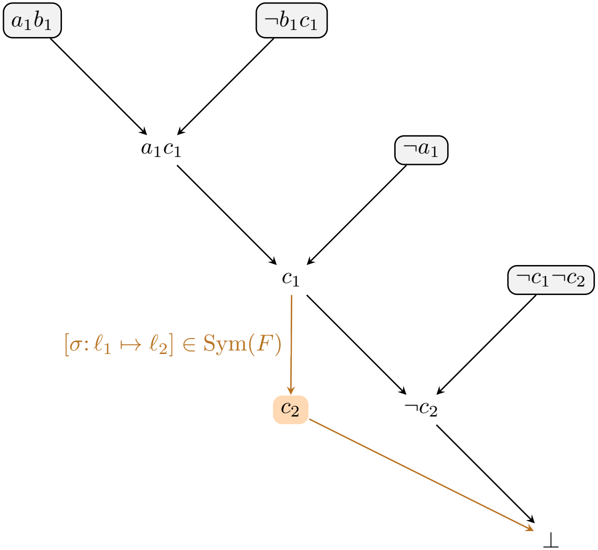
Extending Resolution: Krishnamurthy's Symmetry Rules



Extending Resolution: Krishnamurthy's Symmetry Rules



Extending Resolution: Krishnamurthy's Symmetry Rules



The SRC Proof Systems

Have a derivation $\pi : F' \vdash C$ from a subformula $F' \subseteq F$.
To derive $\sigma(C)$ from C in one step we need a renaming σ with

SRC-1 (Global Symmetries)

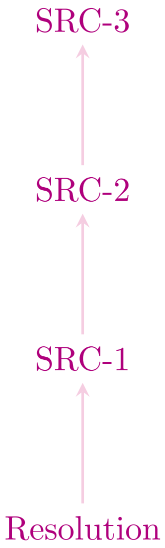
$\sigma(F) \subseteq F$

SRC-2 (Local Symmetries)

$\sigma(F') \subseteq F$

SRC-3 (Dynamic Symmetries)

also allow symmetries in resolvents



Battle SRC-1 With Asymmetric Graphs

Asymmetric Graph G: $\mathrm{Aut}(G) = \{\mathrm{id}\}$

Battle SRC-1 With Asymmetric Graphs

Asymmetric Graph G:	$\mathrm{Aut}(G) = \{\mathrm{id}\}$
Lemma:	Asymmetric graphs $\Longrightarrow$ Asymmetric ISO-formula

Battle SRC-1 With Asymmetric Graphs

Asymmetric Graph G:	$\mathrm{Aut}(G) = \{\mathrm{id}\}$
Lemma:	Asymmetric graphs $\Longrightarrow$ Asymmetric ISO-formula
Lemma:	Asymmetric formula $\Longrightarrow$ Res-Size $=$ SRC-1-Size [Szeider]

Asymmetric Graphs With Large Weisfeiler–Leman-Dimension

[Dawar and Khan] showed: There are pairs of non-isomorphic graphs that are

- asymmetric (unlike CFI-graphs)
- have small size $\mathrm{O}(k)$
- with large WL-dim k
- and color classes of size 4

Without looking at ISO-formula:

$$(G,\lambda) \equiv_{\mathscr{L}_k} (H,\mu) \implies \mathrm{Size}\big(\mathrm{ISO}(G,H) \vdash \bot\big) \geq \exp\left(\Omega\left(\frac{k^2}{\text{sum of color class sizes}}\right)\right)$$

Result: An Exponential GI Lower Bound for SRC-1

Our Result:

There is a family of non-isomorphic graph pairs (G_n, H_n)

- with $\mathrm{O}(n)$ vertices each,
- such that any SRC-1 refutation of $\mathrm{ISO}(G_n, H_n)$ requires

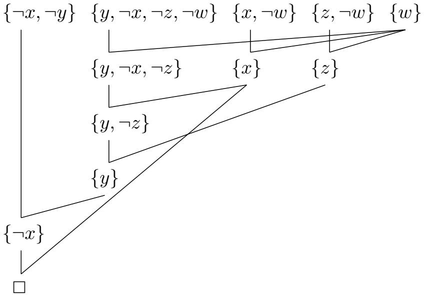
size $\exp\big(\Omega(n)\big)$.

Reversible Pebbling and Resolution Space

STACS 2020, Montpellier
Computational Complexity 30(7), 2021

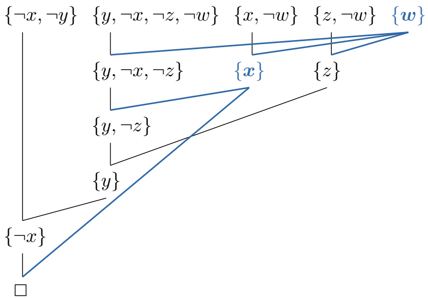
General vs. Tree-like Resolution

General refutation DAG G_π



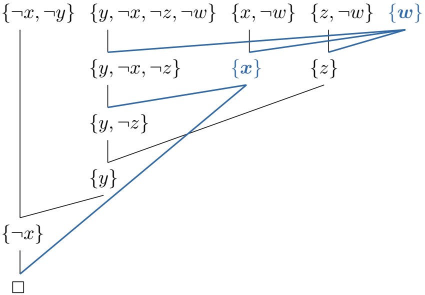
General vs. Tree-like Resolution

General refutation DAG G_π

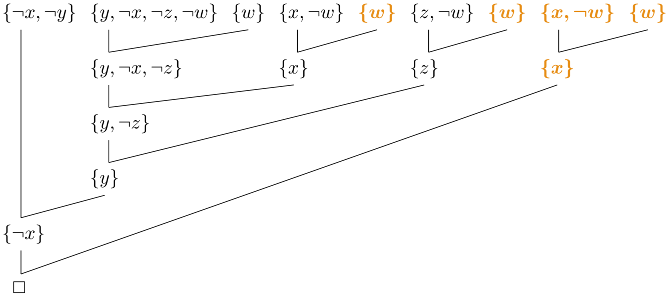


General vs. Tree-like Resolution

General refutation DAG G_π

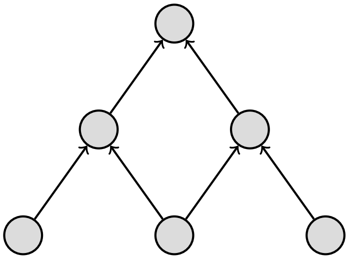


Tree-like refutation DAG G_π



The Black Pebble Game

Goal: Get a single black pebble on the sink of the graph.

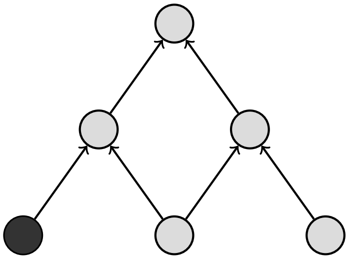


space($\mathcal{P}$)

$\max$ # of pebbles used at any point during the pebbling $\mathcal{P}$:

The Black Pebble Game

Goal: Get a single black pebble on the sink of the graph.



space($\mathcal{P}$)

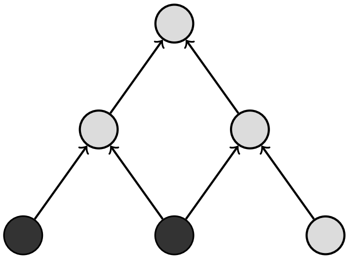
max # of pebbles used at any point during the pebbling $\mathcal{P}$:

|

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble (in particular: can always pebble sources)

The Black Pebble Game

Goal: Get a single black pebble on the sink of the graph.



space($\mathcal{P}$)

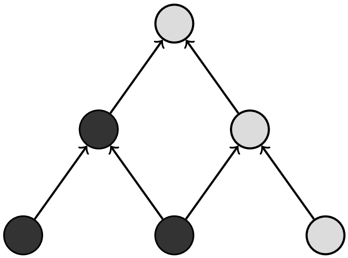
$\max$ # of pebbles used at any point during the pebbling $\mathcal{P}$:

$||$

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble (in particular: can always pebble sources)

The Black Pebble Game

Goal: Get a single black pebble on the sink of the graph.



space($\mathcal{P}$)

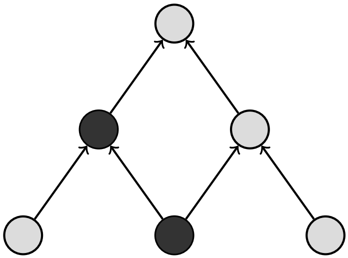
max # of pebbles used at any point during the pebbling $\mathcal{P}$:

|||

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble (in particular: can always pebble sources)

The Black Pebble Game

Goal: Get a single black pebble on the sink of the graph.



space($\mathcal{P}$)

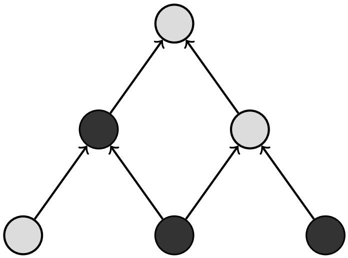
max # of pebbles used at any point during the pebbling $\mathcal{P}$:

|||

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble (in particular: can always pebble sources)
- **Pebble Removal:** At any time

The Black Pebble Game

Goal: Get a single black pebble on the sink of the graph.



space($\mathcal{P}$)

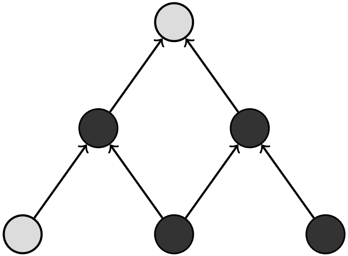
max # of pebbles used at any point during the pebbling $\mathcal{P}$:

|||

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble (in particular: can always pebble sources)
- **Pebble Removal:** At any time

The Black Pebble Game

Goal: Get a single black pebble on the sink of the graph.



space($\mathcal{P}$)

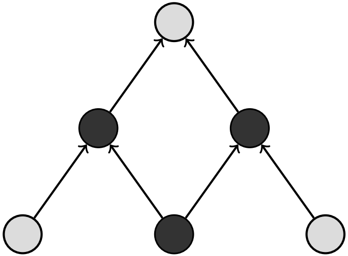
$\max$ # of pebbles used at any point during the pebbling $\mathcal{P}$:

||||

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble (in particular: can always pebble sources)
- **Pebble Removal:** At any time

The Black Pebble Game

Goal: Get a single black pebble on the sink of the graph.



space($\mathcal{P}$)

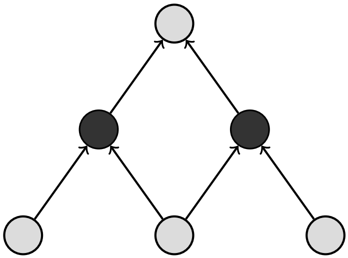
$\max$ # of pebbles used at any point during the pebbling $\mathcal{P}$:

||||

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble (in particular: can always pebble sources)
- **Pebble Removal:** At any time

The Black Pebble Game

Goal: Get a single black pebble on the sink of the graph.



space($\mathcal{P}$)

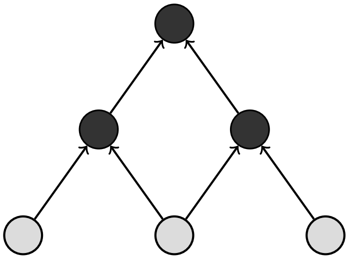
$\max$ # of pebbles used at any point during the pebbling $\mathcal{P}$:

||||

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble (in particular: can always pebble sources)
- **Pebble Removal:** At any time

The Black Pebble Game

Goal: Get a single black pebble on the sink of the graph.



space($\mathcal{P}$)

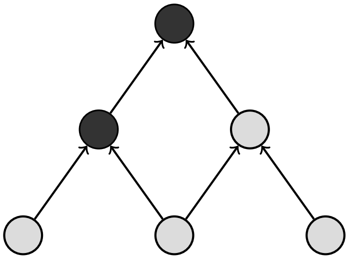
$\max$ # of pebbles used at any point during the pebbling $\mathcal{P}$:

||||

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble (in particular: can always pebble sources)
- **Pebble Removal:** At any time

The Black Pebble Game

Goal: Get a single black pebble on the sink of the graph.



space($\mathcal{P}$)

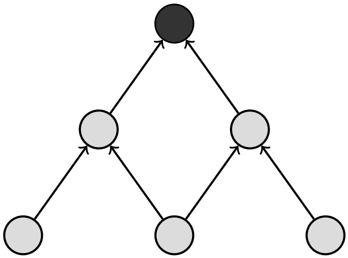
$\max$ # of pebbles used at any point during the pebbling $\mathcal{P}$:

||||

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble (in particular: can always pebble sources)
- **Pebble Removal:** At any time

The Black Pebble Game

Goal: Get a single black pebble on the sink of the graph.



space($\mathcal{P}$)

$\max$ # of pebbles used at any point during the pebbling $\mathcal{P}$:

||||

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble (in particular: can always pebble sources)
- **Pebble Removal:** At any time

Complexity Measure for the Black Pebble Game

$$\mathsf{Black}(G) := \min_{\text{black pebblings } \mathcal{P}} \Big(\underbrace{\max \ \# \text{ of pebbles used at any point in } \mathcal{P}}_{=:\mathsf{space}(\mathcal{P})} \Big)$$

Complexity Measure for the Black Pebble Game

$$\mathsf{Black}(G) := \min_{\text{black pebblings } \mathcal{P}} \Big(\underbrace{\max \ \# \text{ of pebbles used at any point in } \mathcal{P}}_{=:\mathsf{space}(\mathcal{P})} \Big)$$

Why do we care about the pebbling price?

Plethora of connections to resolution, e. g.,

$$\mathrm{CS}(F \vdash \bot) = \min_{\pi : F \vdash \bot} \mathsf{Black}(G_\pi)$$

[Esteban, Torán '01: Space bounds for res.]

Complexity Measure for the Black Pebble Game

$$\mathsf{Black}(G) := \min_{\text{black pebblings } \mathcal{P}} \Bigl(\underbrace{\max \ \# \text{ of pebbles used at any point in } \mathcal{P}}_{=:\mathsf{space}(\mathcal{P})} \Bigr)$$

Why do we care about the pebbling price?

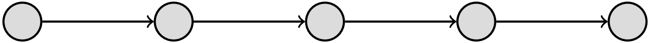
What about Tree-CS?

Plethora of connections to resolution, e. g.,

$$\mathrm{CS}(F \vdash \bot) = \min_{\pi : F \vdash \bot} \mathsf{Black}(G_\pi)$$

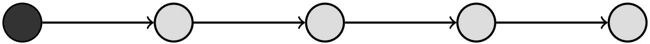
[Esteban, Torán '01: Space bounds for res.]

The Reversible Pebble Game



Different rules:

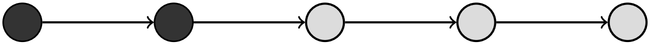
The Reversible Pebble Game



Different rules:

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble

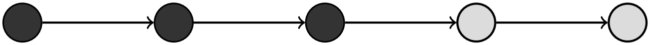
The Reversible Pebble Game



Different rules:

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble

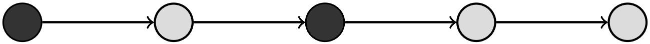
The Reversible Pebble Game



Different rules:

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble

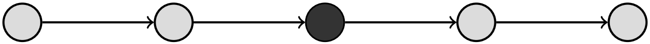
The Reversible Pebble Game



Different rules:

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble
- **Pebble Removal:** Only if all direct predecessors have a pebble

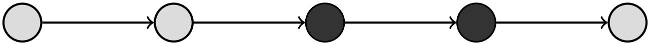
The Reversible Pebble Game



Different rules:

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble
- **Pebble Removal:** Only if all direct predecessors have a pebble

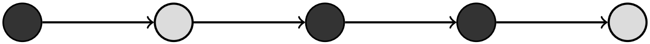
The Reversible Pebble Game



Different rules:

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble
- **Pebble Removal:** Only if all direct predecessors have a pebble

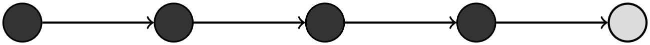
The Reversible Pebble Game



Different rules:

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble
- **Pebble Removal:** Only if all direct predecessors have a pebble

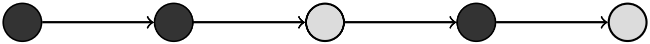
The Reversible Pebble Game



Different rules:

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble
- **Pebble Removal:** Only if all direct predecessors have a pebble

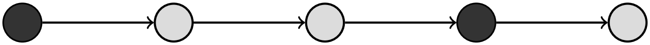
The Reversible Pebble Game



Different rules:

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble
- **Pebble Removal:** Only if all direct predecessors have a pebble

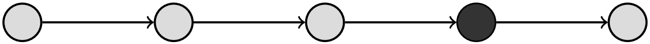
The Reversible Pebble Game



Different rules:

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble
- **Pebble Removal:** Only if all direct predecessors have a pebble

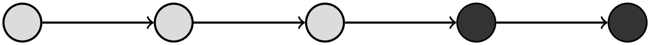
The Reversible Pebble Game



Different rules:

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble
- **Pebble Removal:** Only if all direct predecessors have a pebble

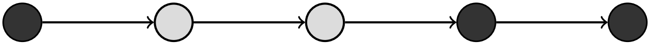
The Reversible Pebble Game



Different rules:

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble
- **Pebble Removal:** Only if all direct predecessors have a pebble

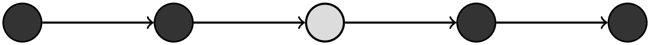
The Reversible Pebble Game



Different rules:

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble
- **Pebble Removal:** Only if all direct predecessors have a pebble

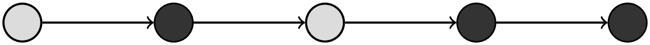
The Reversible Pebble Game



Different rules:

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble
- **Pebble Removal:** Only if all direct predecessors have a pebble

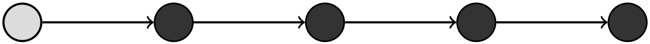
The Reversible Pebble Game



Different rules:

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble
- **Pebble Removal:** Only if all direct predecessors have a pebble

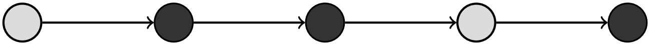
The Reversible Pebble Game



Different rules:

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble
- **Pebble Removal:** Only if all direct predecessors have a pebble

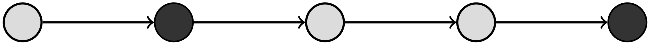
The Reversible Pebble Game



Different rules:

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble
- **Pebble Removal:** Only if all direct predecessors have a pebble

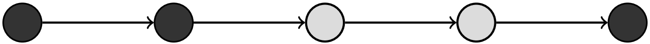
The Reversible Pebble Game



Different rules:

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble
- **Pebble Removal:** Only if all direct predecessors have a pebble

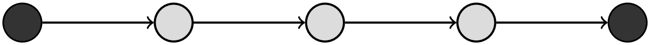
The Reversible Pebble Game



Different rules:

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble
- **Pebble Removal:** Only if all direct predecessors have a pebble

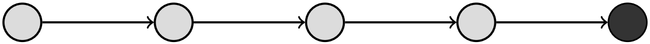
The Reversible Pebble Game



Different rules:

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble
- **Pebble Removal:** Only if all direct predecessors have a pebble

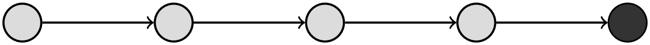
The Reversible Pebble Game



Different rules:

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble
- **Pebble Removal:** Only if all direct predecessors have a pebble

The Reversible Pebble Game



Different rules:

- **Pebble Placement:** On empty vertex if all direct predecessors have a pebble
- **Pebble Removal:** Only if all direct predecessors have a pebble

Complexity measure: **Rev**(G)

Result 1: New Connection Of Tree-CS and Rev

For formulas stating the rules of the pebbling game:

$$\mathsf{Rev}(G) \leq \mathrm{Tree\text{-}CS}\big(\mathrm{Peb}_G[\oplus_2] \vdash \bot\big) \lesssim \mathsf{Rev}(G).$$

For any UNSAT formula in n variables:

$$\mathrm{Tree\text{-}CS}(F \vdash \bot) \lesssim \min_{\pi : F \vdash \bot} \mathsf{Rev}(G_\pi) \lesssim \mathrm{Tree\text{-}CS}(F \vdash \bot) \cdot \log n.$$

$\mathrm{CS}(F \vdash \bot) = \min_{\pi : F \vdash \bot} \mathsf{Black}(G_\pi)$

Result 2: Separations between Tree-CS and CS

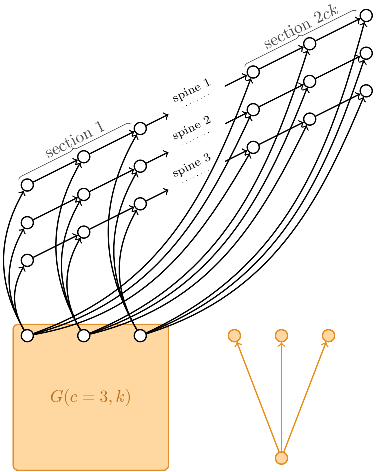
Idea:

- $\mathrm{CS}(\mathrm{Peb}_{G_n}[\oplus_2] \vdash \bot) = \mathrm{O}\big(\mathsf{Black}(G_n)\big)$
- $\text{Tree-CS}\big(\mathrm{Peb}_{G_n}[\oplus_2] \vdash \bot\big) = \Omega\big(\mathsf{Rev}(G_n)\big)$

Result 2: Separations between Tree-CS and CS

Idea:

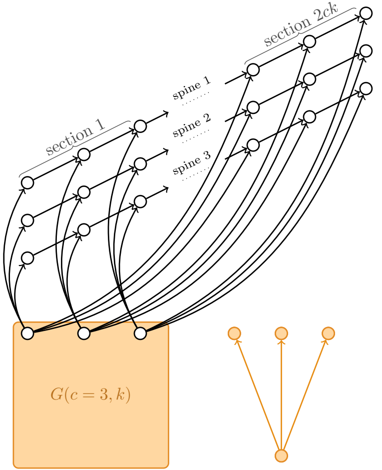
- $\mathrm{CS}\left(\mathrm{Peb}_{G_n}[\oplus_2] \vdash \bot\right) = \mathrm{O}\left(\mathsf{Black}(G_n)\right)$
- $\text{Tree-CS}\left(\mathrm{Peb}_{G_n}[\oplus_2] \vdash \bot\right) = \Omega\left(\mathsf{Rev}(G_n)\right)$



Result 2: Separations between Tree-CS and CS

Idea:

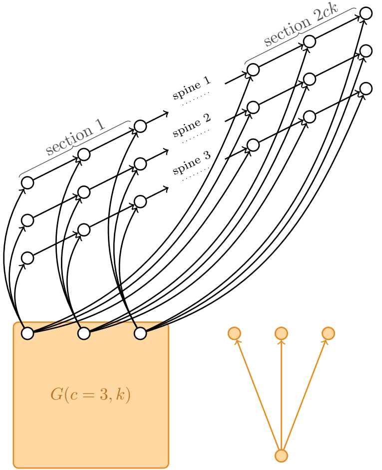
- $\mathrm{CS}(\mathrm{Peb}_{G_n}[\oplus_2] \vdash \bot) = \mathrm{O}\big(\overbrace{\mathsf{Black}(G_n)}^{s(n)}\big)$
- $\text{Tree-CS}\big(\mathrm{Peb}_{G_n}[\oplus_2] \vdash \bot\big) = \Omega\big(\underbrace{\mathsf{Rev}(G_n)}_{s(n)\log n}\big)$



Result 2: Separations between Tree-CS and CS

Idea:

- $\mathrm{CS}\big(\mathrm{Peb}_{G_n}[\oplus_2] \vdash \bot\big) = \mathrm{O}\Big(\overbrace{\text{Black}(G_n)}^{s(n)}\Big)$
- $\text{Tree-CS}\big(\mathrm{Peb}_{G_n}[\oplus_2] \vdash \bot\big) = \Omega\Big(\underbrace{\text{Rev}(G_n)}_{s(n)\log n}\Big)$



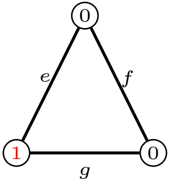
Only room for improvement:
best pebbling strategy needs to revisit nodes

Result 3: Upper Bounds & Optimal Separations

How large can the gap grow?

Razborov's amortized measures
$\mathrm{CS}^*(F \vdash \bot) := \min_{\pi : F \vdash \bot} (\mathrm{CS}(\pi) \cdot \log \mathrm{Size}(\pi))$

$$\text{Tree-CS}(F \vdash \bot) \lesssim \mathrm{CS}^*(F \vdash \bot)$$



$\mathrm{Ts} :=$
$e + g \equiv 1 \pmod 2$
$e + f \equiv 0 \pmod 2$
$f + g \equiv 0 \pmod 2$

- For **Tseitin formulas** (encoding the *degree sum principle*) over n vertices:

$$\text{Tree-CS}(\mathrm{Ts} \vdash \bot) \lesssim \mathrm{CS}(\mathrm{Ts} \vdash \bot) \cdot \log n$$

- $\exists$ a Tseitin family:

$$\text{Tree-CS}(\mathrm{Ts} \vdash \bot) = \Omega\Big(\mathrm{CS}(\mathrm{Ts} \vdash \bot) \cdot \log n\Big)$$

Interesting Open Research Problems

Interesting Open Research Problems

- Can the bound $\mathrm{Tree\text{-}CS}(F \vdash \bot) \lesssim \mathrm{CS}^*(F \vdash \bot)$ be brought down to a $\log n$ factor?
- Is there a (interactive) game for CS?
- Classical complexity:
  $\mathrm{RCS} := \{(F,k) \mid \mathrm{CS}(F \vdash \bot) \leq k\} \in$ coNP-hard, PSPACE.
  Is $\mathrm{RCS} \in$ coNP? Is $\mathrm{RCS} \in$ PSPACE-complete?
- How does one show "true" exponential lower bounds (for a symmetric formula) in the SRC systems?

List of publications

→ **Number of Variables for Graph Identification and the Resolution of GI Formulas.**
J. Torán and F. Wörz. Accepted at *CSL 2022*.

Evidence for Long-Tails in SLS Algorithms.
F. Wörz and J.-H. Lorenz. *ESA 2021*. Best Student Paper Award.

On the Effect of Learned Clauses on Stochastic Local Search.
J.-H. Lorenz and F. Wörz. *SAT 2020*.

→ **Reversible Pebble Games and the Relation Between Tree-Like and General Resolution Space.**
J. Torán and F. Wörz. *Computational Complexity* 2021 and *STACS 2020*.