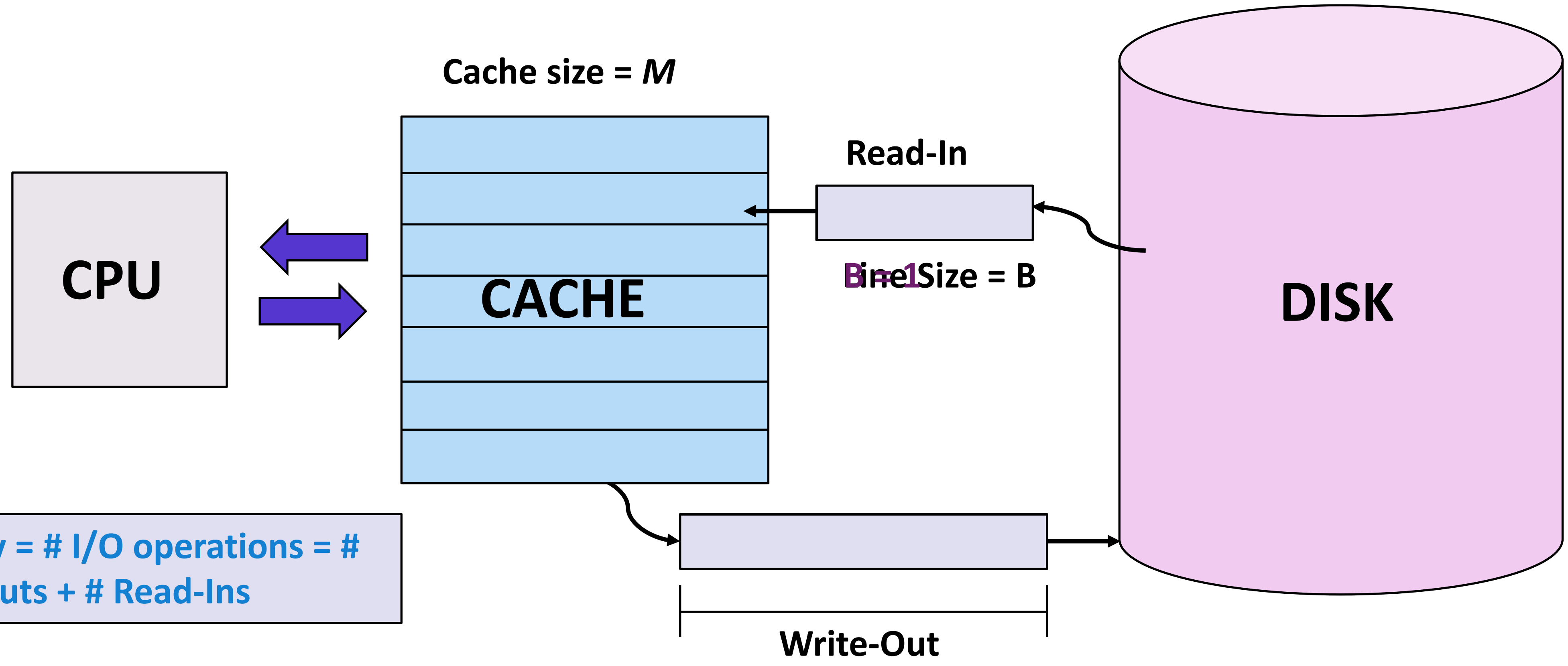
Red-Blue Pebble Game: Complexity of Computing the Trade-Off between Cache Size and Memory Transfers

Erik D. Demaine and Quanquan C. Liu

I/O-Model

- Two-level memory hierarchy: fast cache and slow memory [HK81, AV88]



I/O-Complexity = # I/O operations = # Write-Outs + # Read-Ins

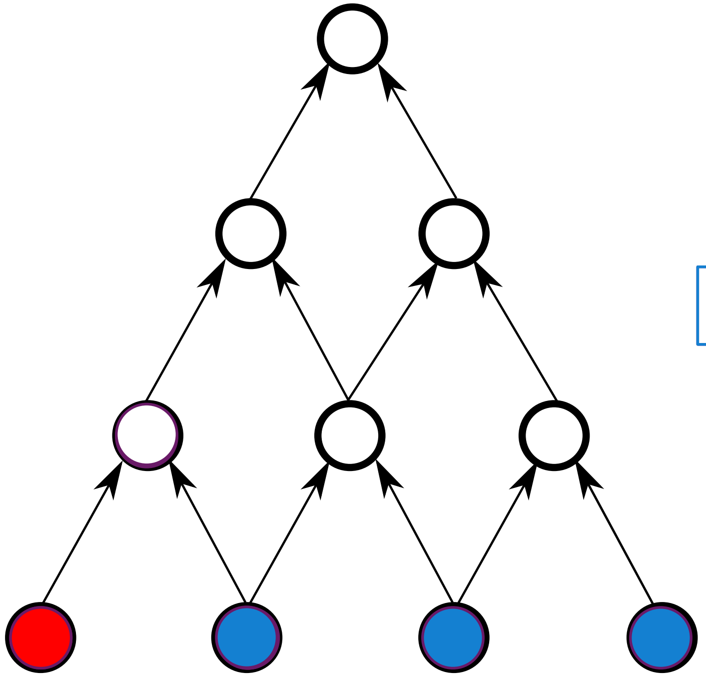
Red-Blue Pebble Game

- Used to **model I/O complexity** of I/O-model [HK81]
- Sequentially **add, remove, and recolor** ”red” and “blue” pebbles on a DAG
- Dependency DAG represents **data dependency** in computation

Red-Blue Pebble Game

Deleting pebble: remove data from cache or disk

in



3 Transitions

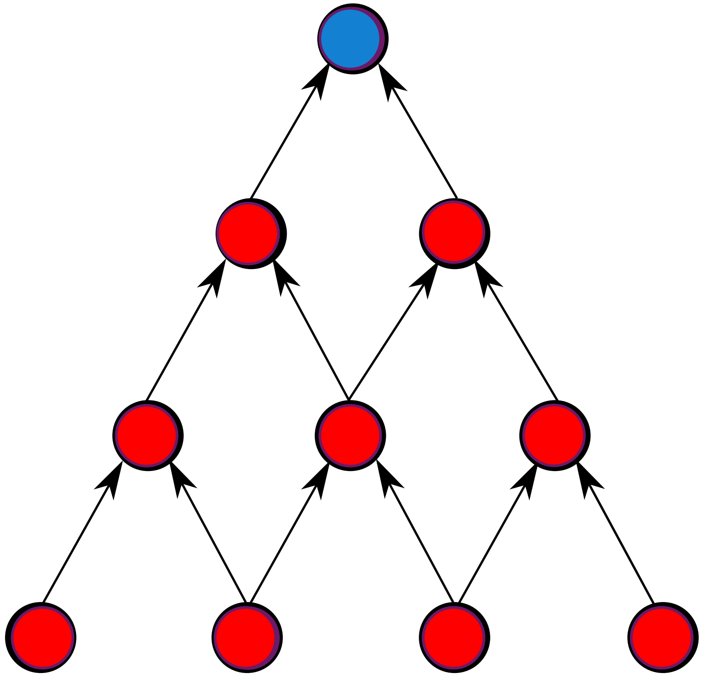
Represents I/Os

Goal: Pebble sink nodes with blue pebbles.

= data in disk

Red-Blue Pebble Game

Minimizing red pebbles:
minimizing cache size = 5



Minimizing transitions: minimizing I/O-complexity (number of read-ins/write-outs) = 5

Goal: Pebble sink nodes with blue pebbles.

Pebble Games and Hardness

- Used to model computation and space constraints in many different models of computation
- Standard (black) pebble game: PSPACE-complete [GLT80]
- Black-white pebble game: PSPACE-complete [HP10]
- Reversible pebble game: PSPACE-complete [CLNV15]

Other Applications

- Protection against large-scale attacks on secure systems
- **Proofs of work** (via pebbling) use large computation time [DNW05]
  - Adversaries build specialized circuits
- **Memory-hard functions** [AS15] — use lots of memory to perform computation
  - Doesn’t account for different access times
- **Bandwidth-hard functions** [BRZ18] — use many I/Os to perform computation

Our Results

Extension of [GLT80]

Thm 1. Computing the number of red pebbles and number of transitions in the Red-Blue Pebble Game is PSPACE-Complete even given constant number of transitions.

Thm 2. Computing the number of red pebbles and number of transitions (even constant) in the Red-Blue Pebble Game with No Deletion is NP-Complete.

Thm 3. Computing the number of red pebbles and number of transitions in the Red-Blue Pebble Game is W[1]-hard when parameterized by the number of transitions, even for layered graphs.

Red-Blue Pebble Game with No Deletions

- No deletion move allowed
- Studies a simpler problem—what does deletion afford in the I/O-model?
- Applications for when computed data need to be maintained
- Can be used to model cases where computation time in cache similar to I/O cost
- Provides an additional proof of NP-completeness for model in [BRZ18] when computation time cost in cache is equal to I/O cost

NP-Completeness Proof

- Similar in spirit to [GLT80] proof framework
- Reduction from Positive 1-in-3 SAT [GJ90]

Positive 1-in-3 SAT [GJ90]: Set $\mathcal{U}$ of variables and $\mathcal{C}$ of clauses over $\mathcal{U}$ where each clause $c \in \mathcal{C}$ has size $|c| = 3$ and all literals in c are positive. Does there exist a truth assignment for $\mathcal{U}$ such that each clause has exactly one true literal?

$$\mathcal{U} = \{x_1, x_2, x_3, x_4, x_5, x_6\}$$

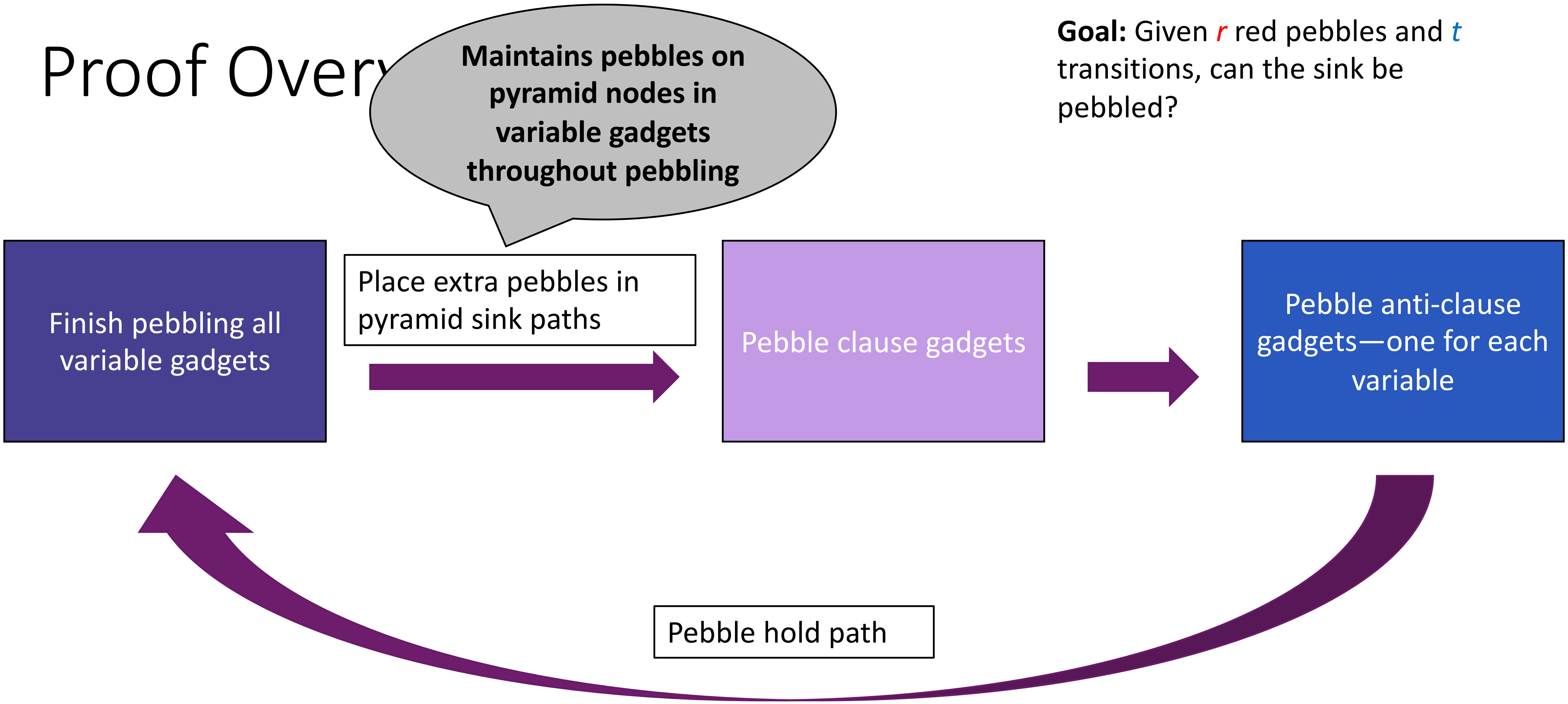
T T F T F F

$$\mathcal{C} = (x_1 \vee x_3 \vee x_6) \wedge (x_2 \vee x_5 \vee x_6) \wedge (x_3 \vee x_4 \vee x_5)$$

$$\mathcal{C} = (x_1 \vee x_3 \vee x_6) \wedge (x_2 \vee x_5 \vee x_6) \wedge (x_3 \vee x_4 \vee x_5)$$

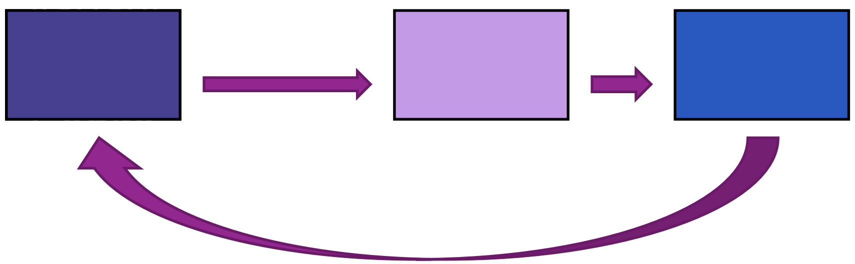
Proof Overv

Goal: Given r red pebbles and t transitions, can the sink be pebbled?



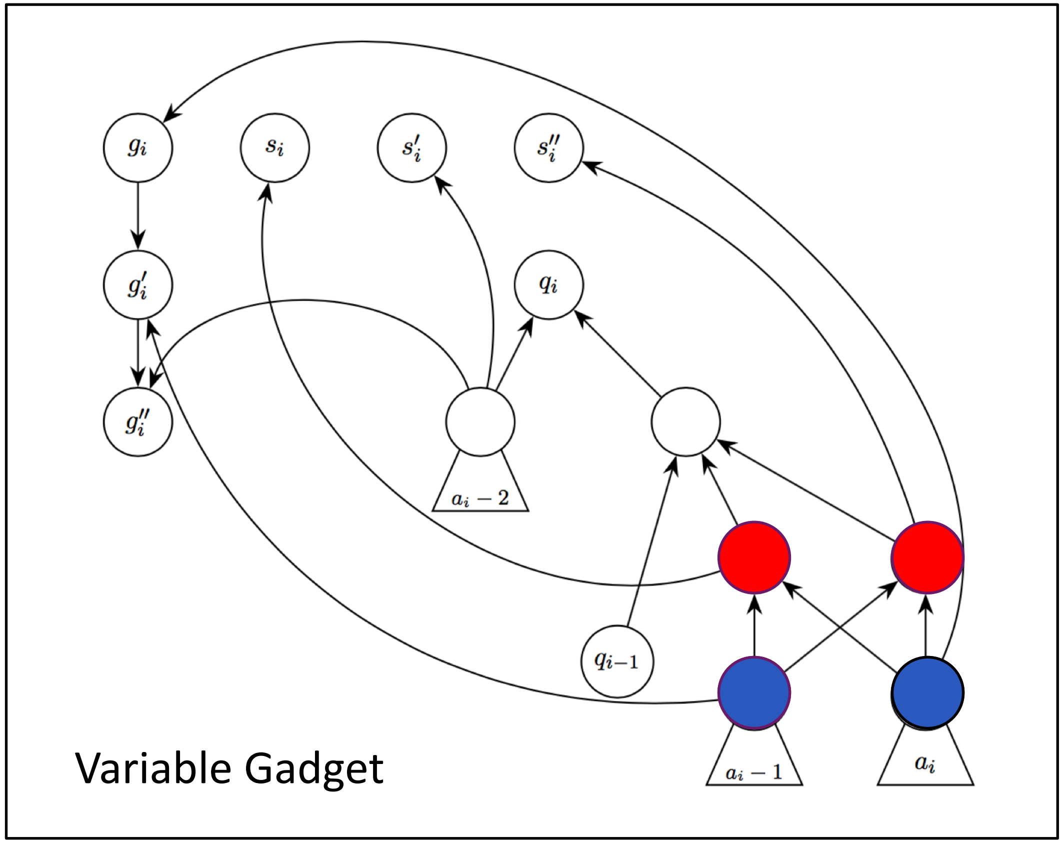
Reduction: The sink can be pebbled using r red pebbles and t transitions if and only if the Positive 1-in-3 SAT instance can be solved for some setting of variables.

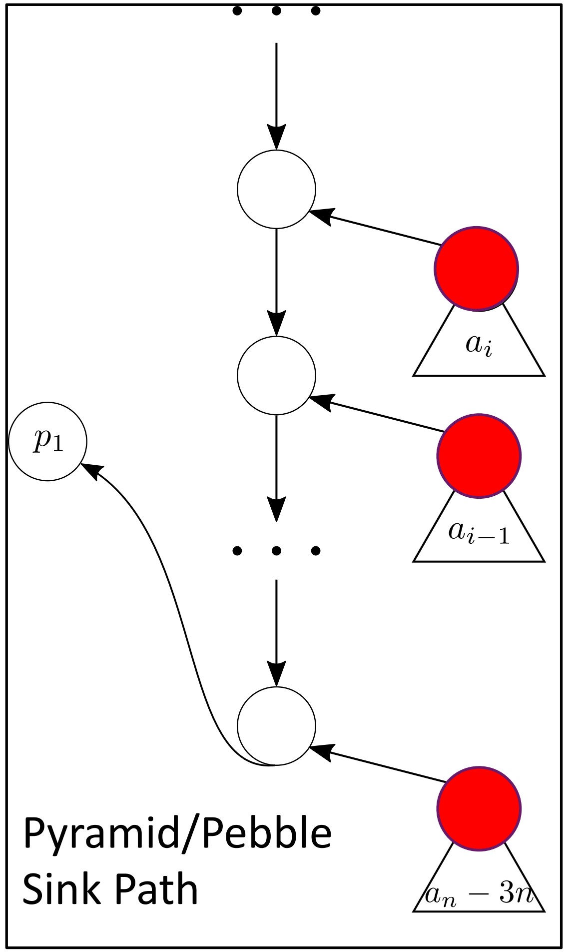
Gadgets



False Configuration

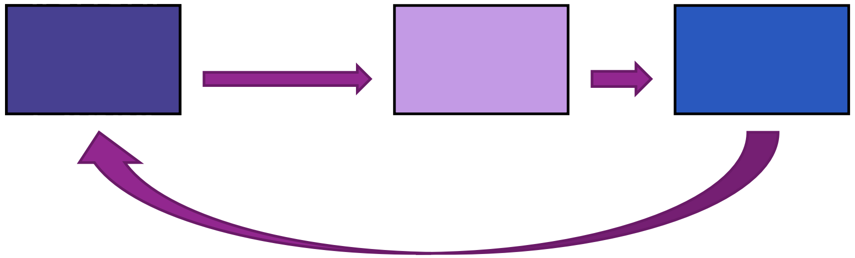
True Configuration





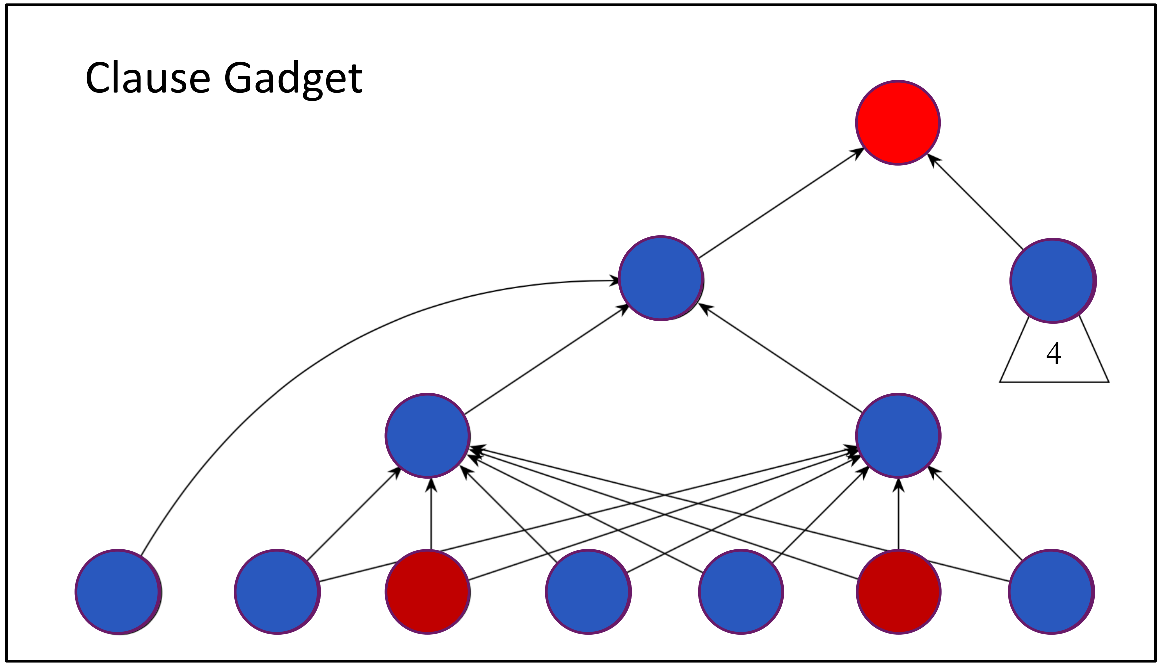
Must keep a pebble on every pyramid in the path.

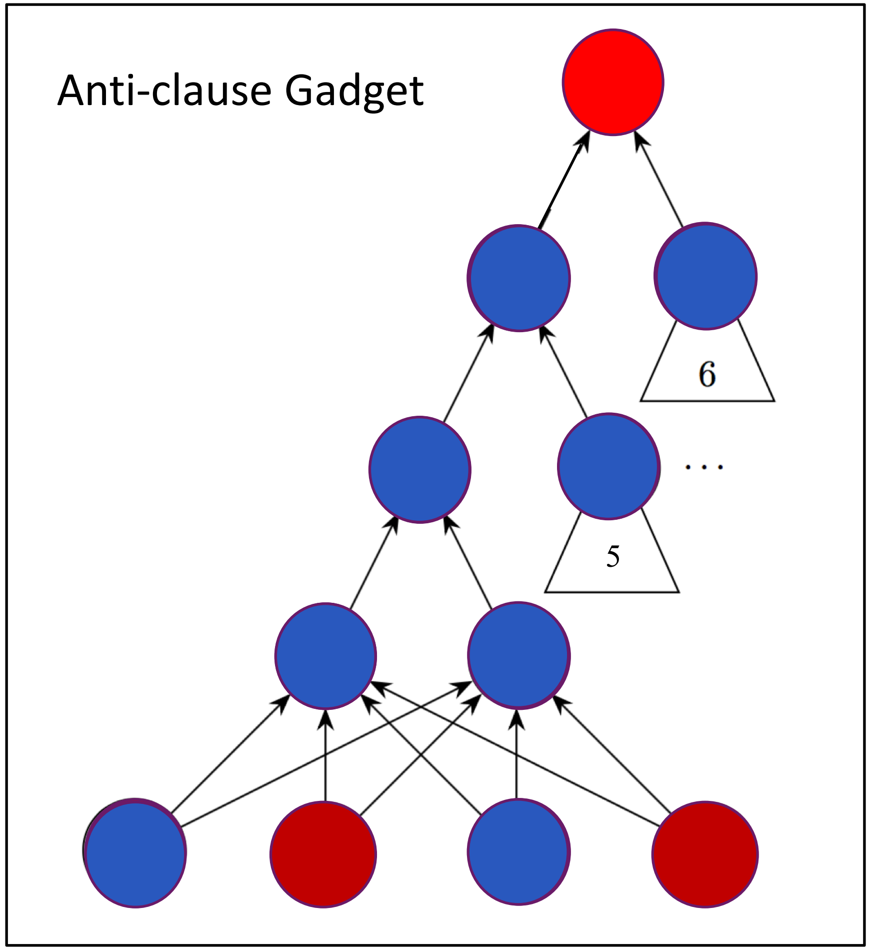
Gadgets



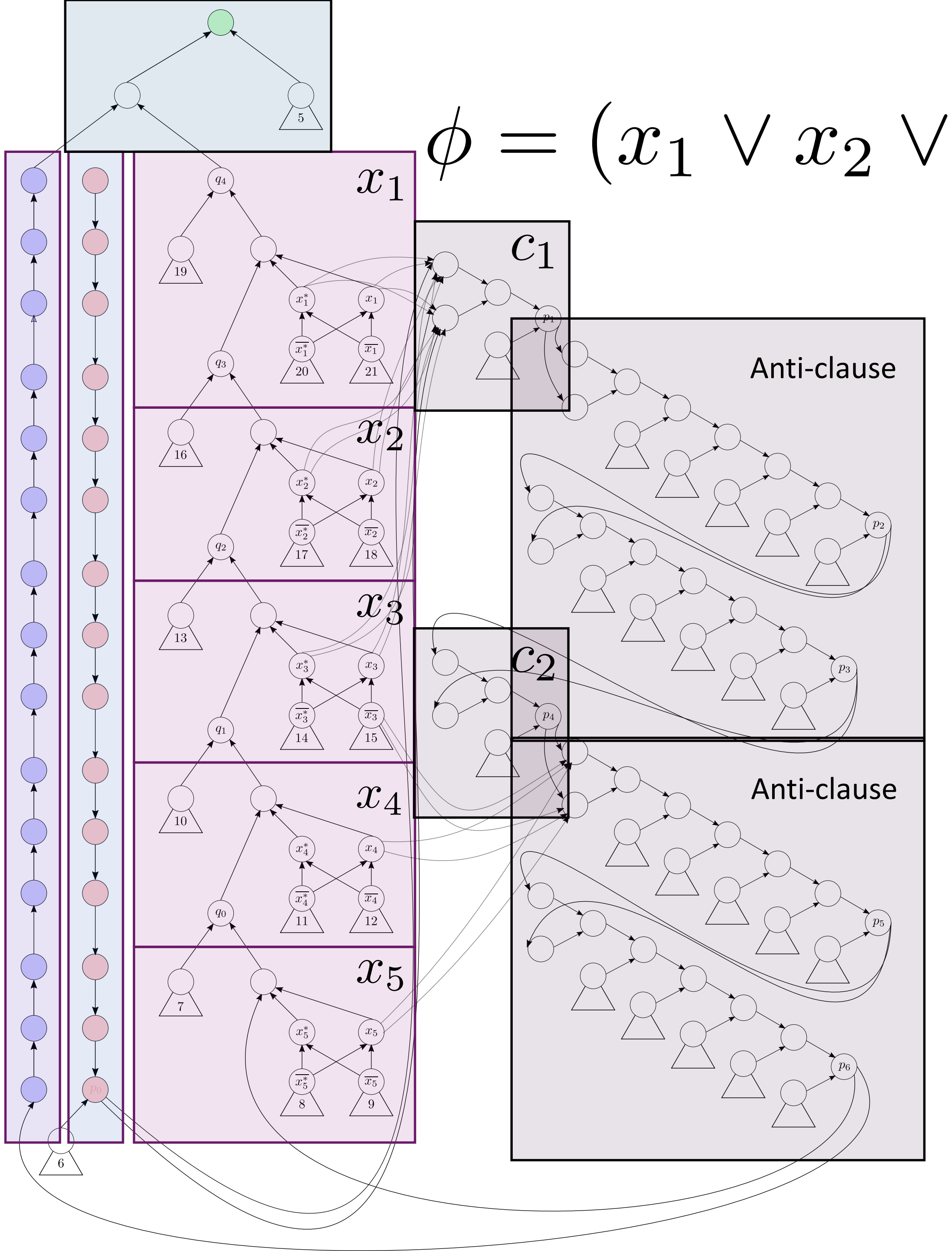
Cannot reduce from 3SAT because too much slack in bound for # of transitions.

Need an exact bound on the number of transitions.





$$\phi = (x_1 \vee x_2 \vee x_3) \wedge (x_3 \vee x_5 \vee x_4)$$



1. Set Variable Gadgets

1) a. Pebble Sink Path

2. Clauses/Anti-clause Gadgets

4. Pebble Hold Path

5. Target Node

Parameterized Complexity

- **Fixed-parameter tractable:** problem parameterized by k can be solved in $f(k)n^{O(1)}$ time
- **W[1]-hardness:** assuming ETH (Exponential Time Hypothesis) no FPT algorithm for problem parameterized k (e.g. FPT $\neq$ W[1])

Exponential Time Hypothesis [IPZ01]: There exists a positive real s such that 3-CNF-SAT with parameter n cannot be solved in time $2^{sn}(n+m)^{O(1)}$.

Our Results

Extension of [GLT80]

Thm 1. Computing the number of red pebbles and number of transitions in the Red-Blue Pebble Game is PSPACE-Complete even given constant number of transitions.

Thm 2. Computing the number of red pebbles and number of transitions (even constant) in the Red-Blue Pebble Game with No Deletion is NP-Complete.

Thm 3. Computing the number of red pebbles and number of transitions in the Red-Blue Pebble Game is W[1]-hard when parameterized by the number of transitions, even for layered graphs.

W[1]-hardness Proof

- Red-blue pebble game parameterized by number of transitions *t* is W[1]-hard
- Reduction from **Weighted 3-CNF SAT**

Weighted 3-CNF SAT(k): Set $\mathcal{U}$ of variables and $\mathcal{C}$ of clauses over $\mathcal{U}$ where each clause $c \in \mathcal{C}$ has size $|c| = 3$ and all literals in *c* are positive. Does there exist a truth assignment for $\mathcal{U}$ such that exactly *k* variables are true in $\mathcal{U}$?

k = 3

$$\mathcal{U} = \{x_1, x_2, x_3, x_4, x_5, x_6\}$$

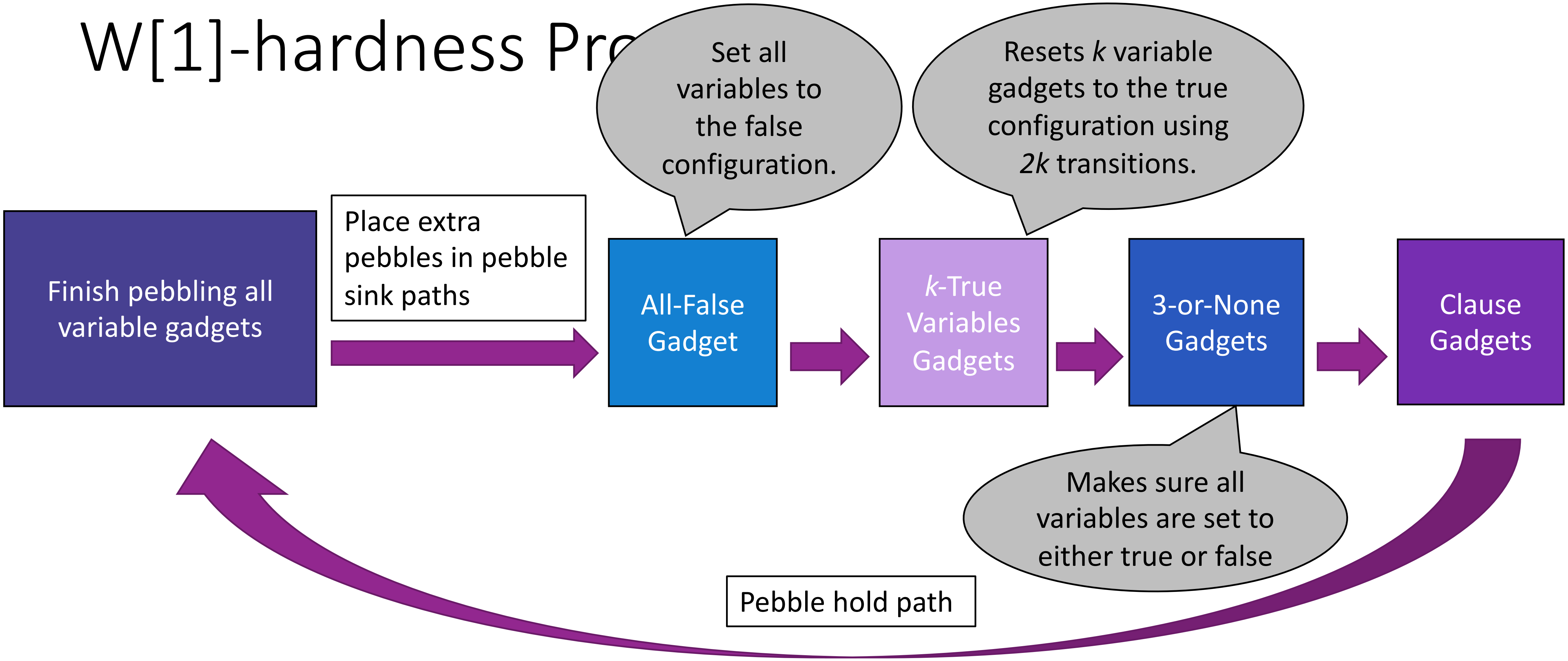
T T F T F F

$$\mathcal{C} = (x_1 \vee x_3 \vee x_6) \wedge (x_2 \vee x_5 \vee x_6) \wedge (x_3 \vee x_4 \vee x_5)$$

$$\mathcal{C} = (x_1 \vee x_3 \vee x_6) \wedge (x_2 \vee x_5 \vee x_6) \wedge (x_3 \vee x_4 \vee x_5)$$

Transitions are limited!

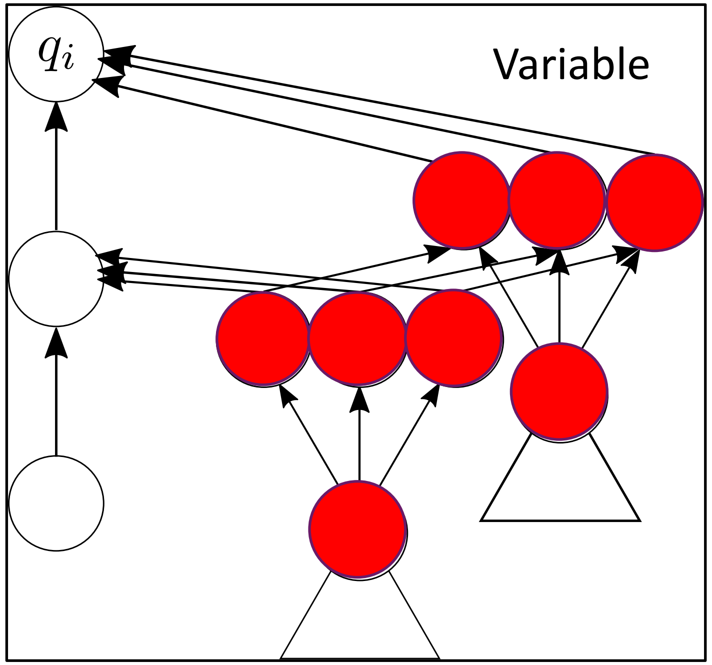
W[1]-hardness Pr

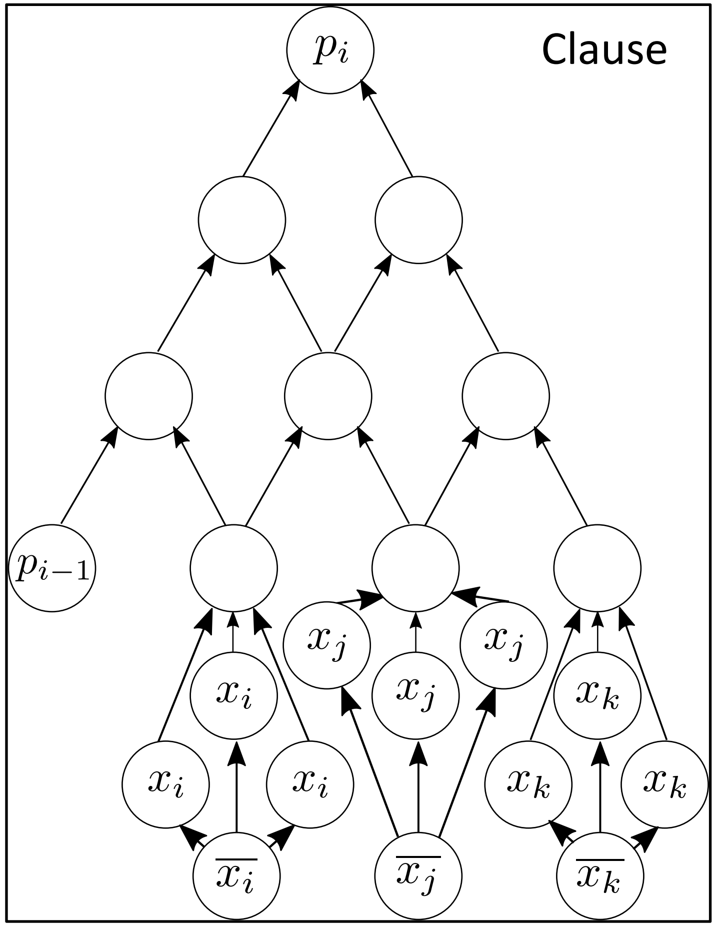


W[1]-hardness Proof Gadgets

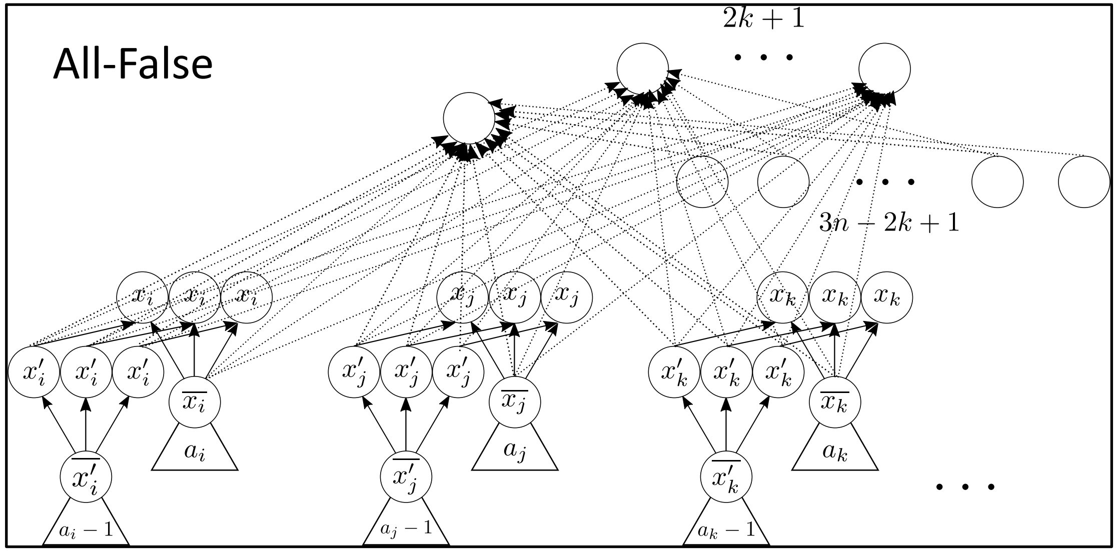
True Configuration

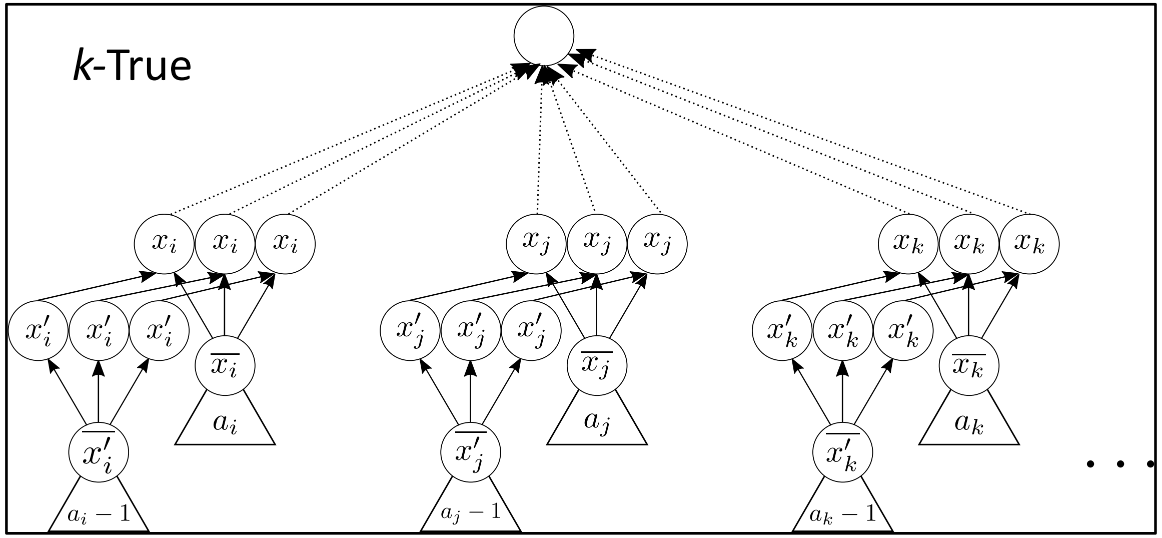
False Configuration

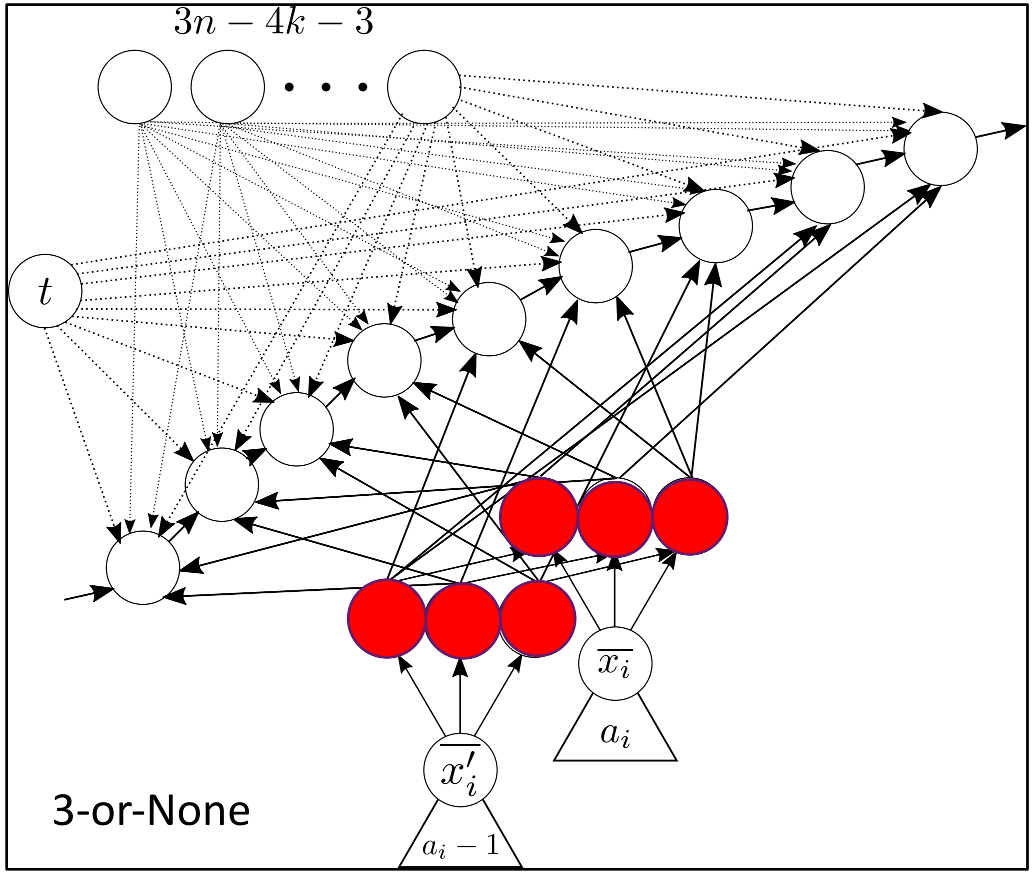


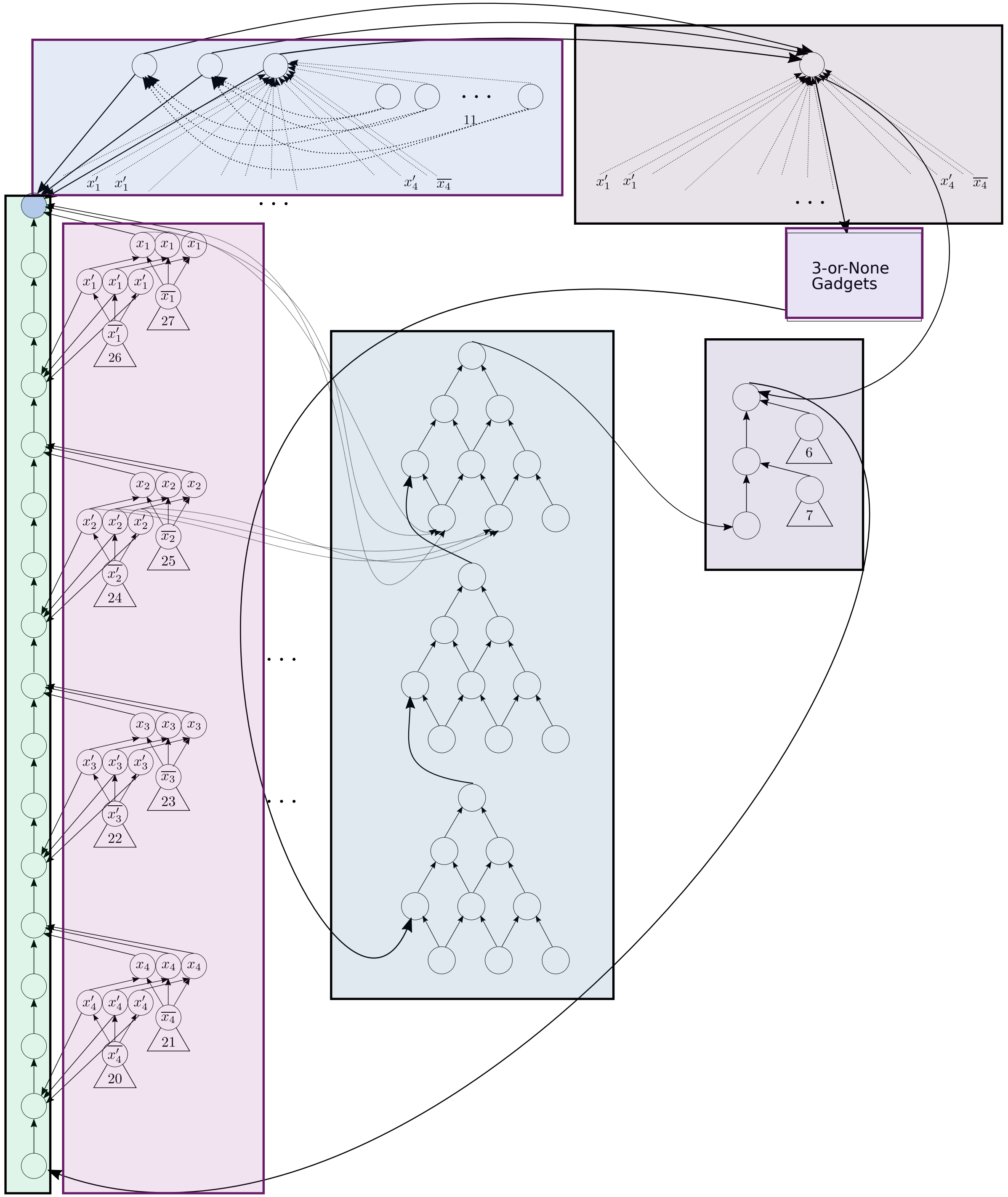


W[1]-hardness Proof Gadgets









$$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_2 \vee x_1 \vee \overline{x_4}) \wedge (\overline{x_2} \vee \overline{x_3} \vee \overline{x_4})$$

1. Set Variable Gadgets
2. All-False Gadget
3. *k*-True Gadget
4. 3-or-None Gadgets
5. Clauses
6. Pebble Sink Path
6. Pebble Hold Path + Target

Open questions

- Hardness of approximation—we don't even have constant factor inapproximation!
- FPT algorithms for restricted classes of graphs
  - Our results can be easily expanded to layered graphs
  - Bounded width graphs?
  - Planar and series-parallel?
- W[1]-hardness when parameterized by the number of red pebbles