Any Monotone Function Is Realized by Interlocked Polygons

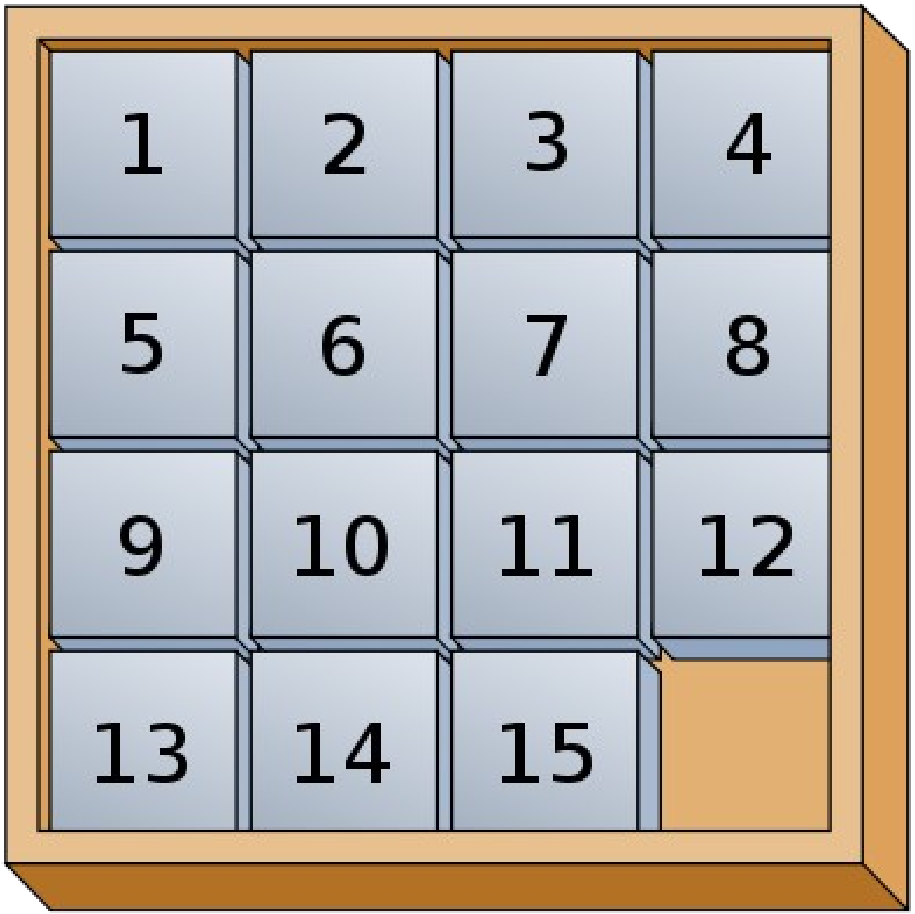
Authors: Erik Demaine, Martin Demaine, and Ryuhei Uehara

Algorithms 2012

Outline

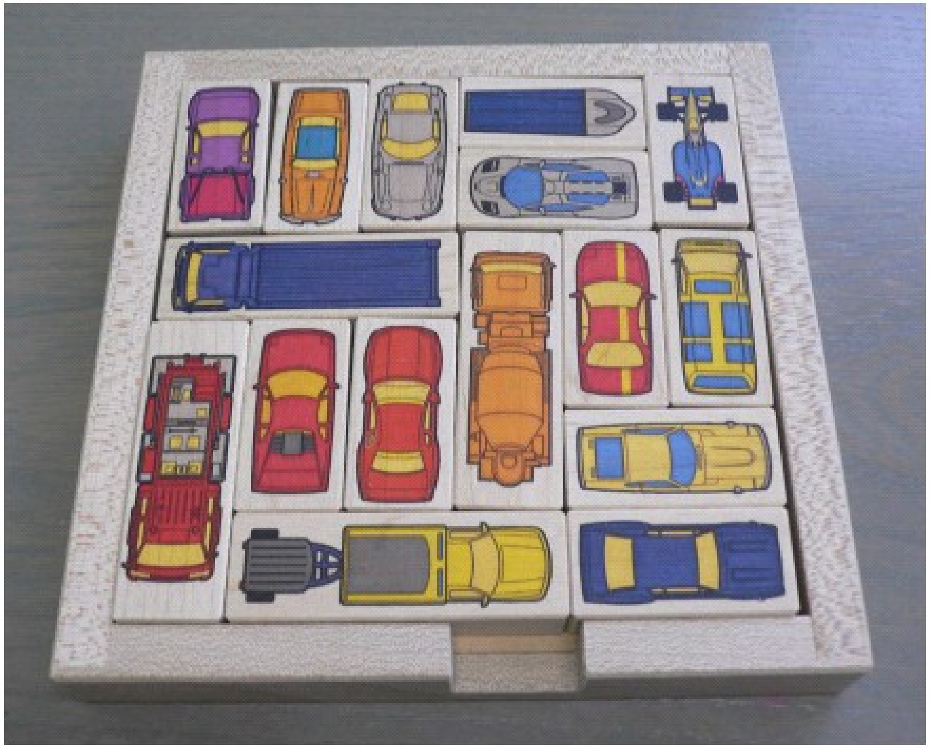
- Introduction:
  - Sliding Block Puzzles
  - Interlocked Polygons
  - Monotone Boolean Functions
- PSPACE-completeness
- Nondeterministic Constraint Logic
  - Introduction
    - True Quantified Boolean Formulas (TQBF)
  - Proof Idea

Sliding Block Puzzles



- There are many variations of sliding block puzzles.
- The idea is to go from an initial state to a goal state through a series of valid moves.
- 15 puzzle is one of the first such puzzles studied.
- Left is the goal state of the puzzle.

Sliding Block Puzzles



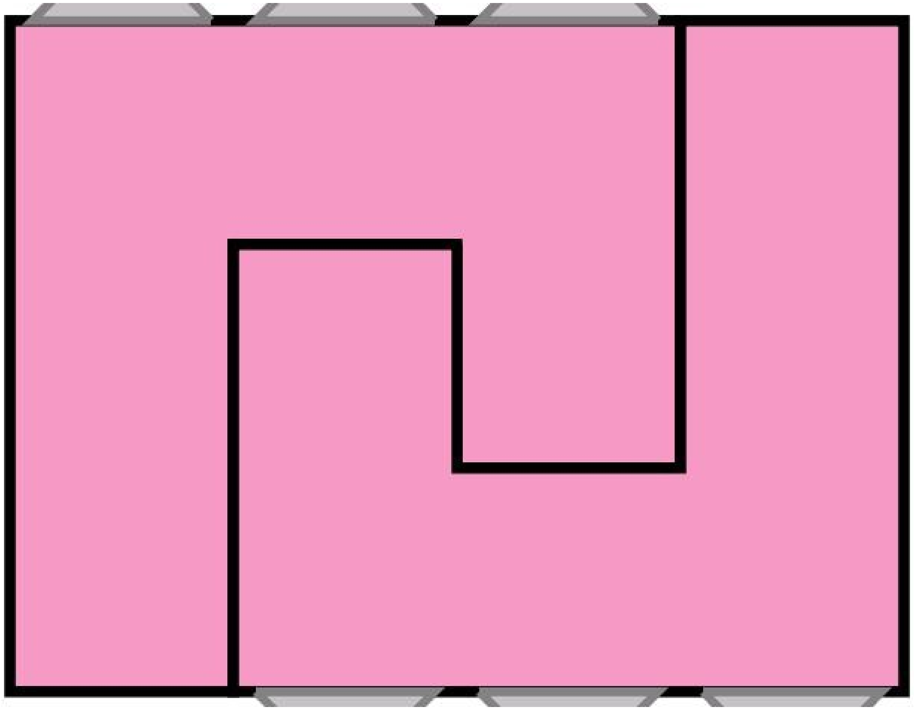
- Rush Hour is another sliding block puzzle variation.
- The goal is to help a specified car escape a traffic jam.
- Note that in both of these problems the less objects (i.e. tiles/cars) the easier the puzzle is to solve.

Sliding Block Puzzles

- 3d variants are possible too naturally but we will see that 2d is already hard.
- The authors introduce the interlocked polygons problem as a generalization of such puzzles.

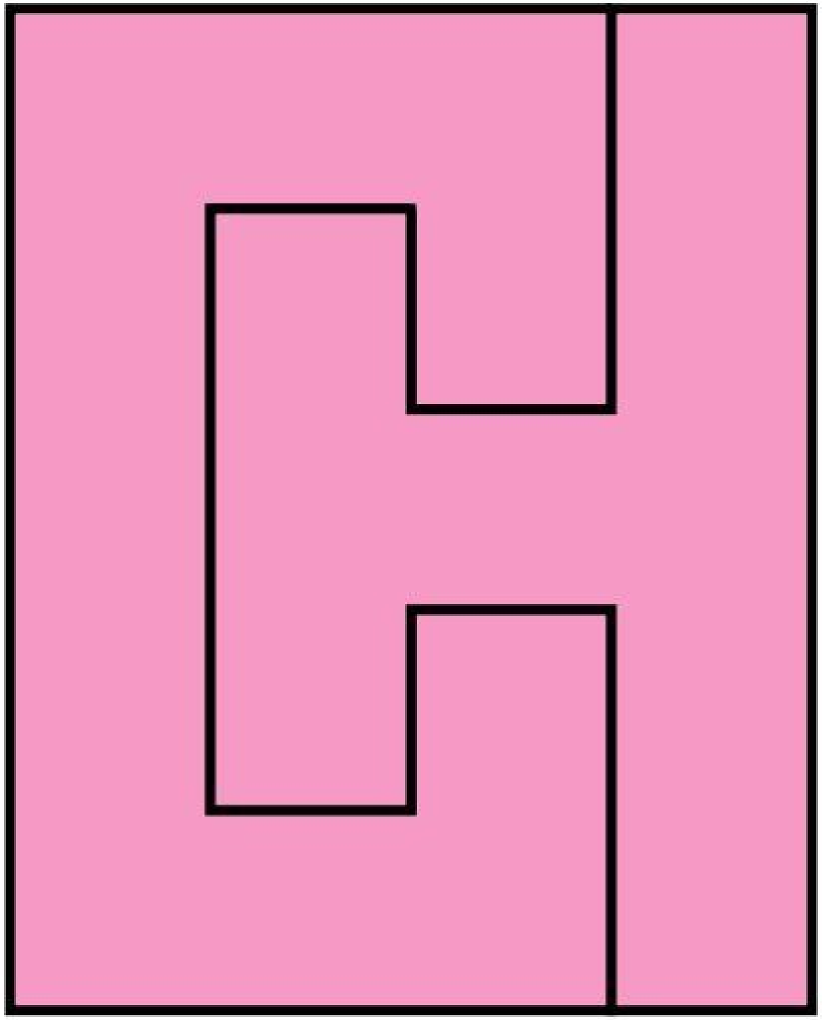
Interlocked Polygons

- Suppose we have a set of n non-overlapping simple polygons.
- The polygons are *interlocked* if no subset can be separated arbitrarily far from the rest.
  - (i.e. separated using translations/rotations which do not cause polygons to overlap)
- Example here.



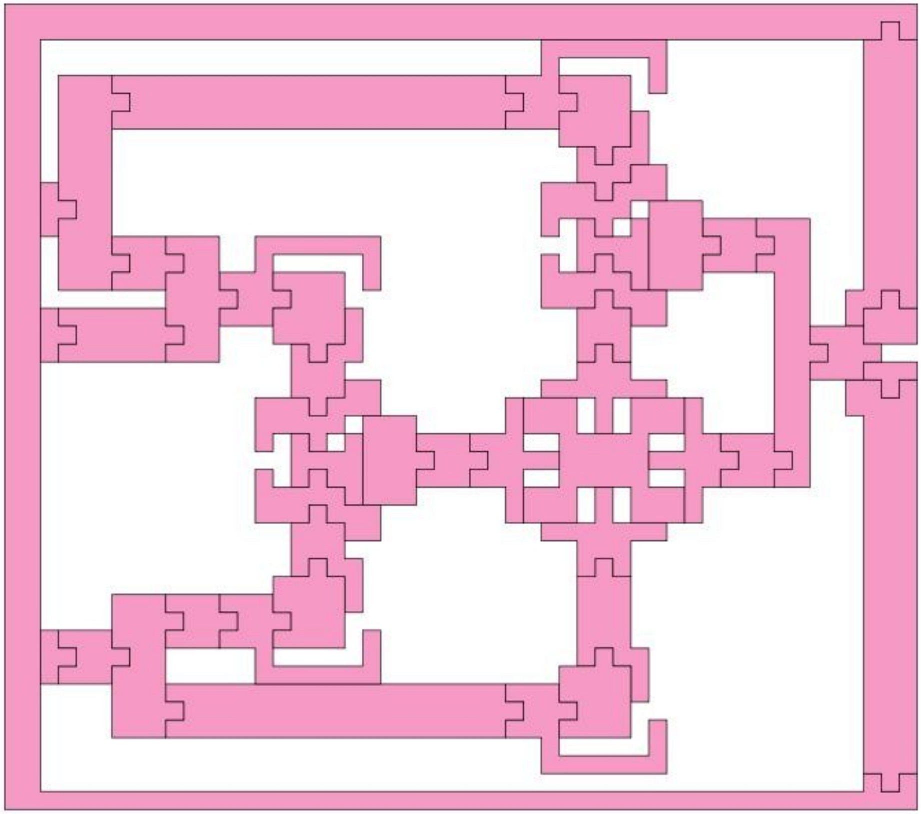
Interlocked Polygons

- Suppose we have a set of n non-overlapping simple polygons.
- The polygons are *interlocked* if no subset can be separated arbitrarily far from the rest.
  - (i.e. separated using translations/rotations which do not cause polygons to overlap)
- Example here.



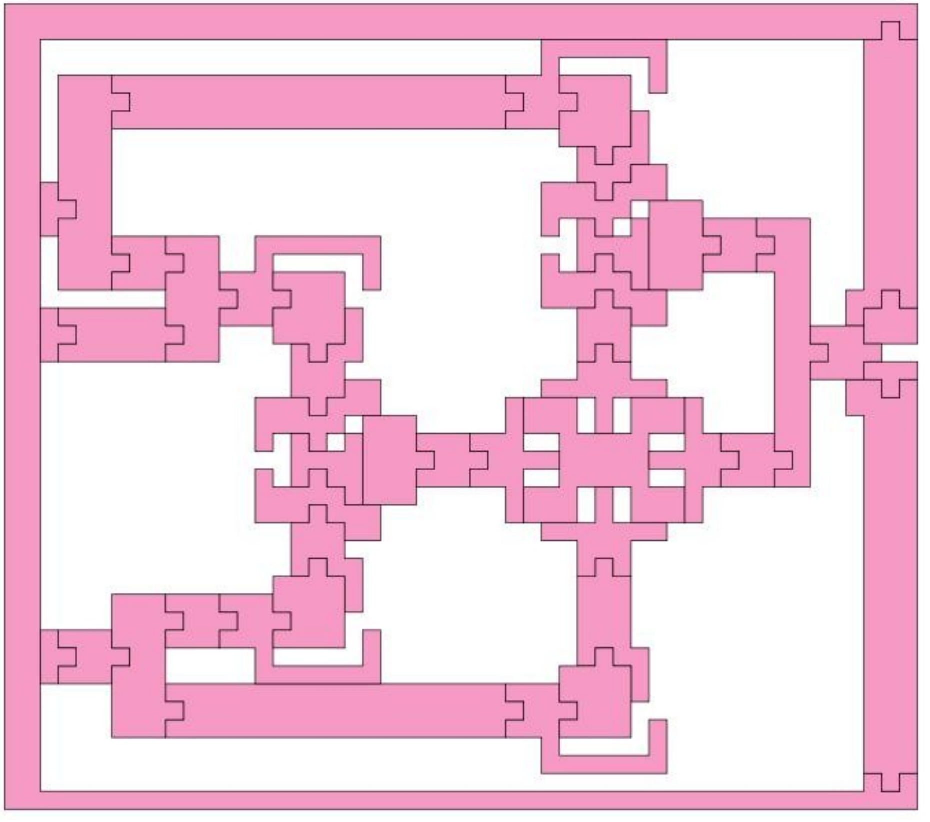
Interlocked Polygons

- Suppose we have a set of n non-overlapping simple polygons.
- The polygons are *interlocked* if no subset can be separated arbitrarily far from the rest.
  - (i.e. separated using translations/rotations which do not cause polygons to overlap)
- Example here.



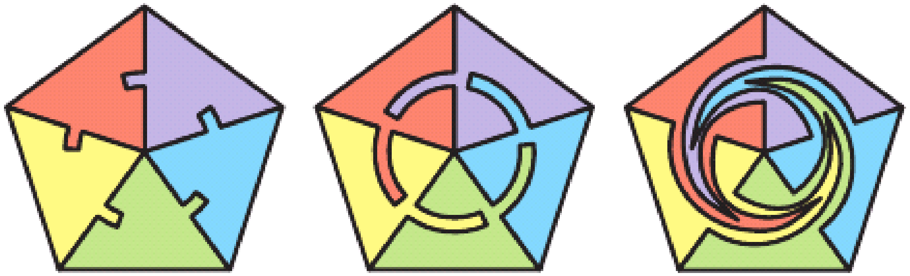
Interlocked Polygons

- The new puzzle they introduce is the *exploding sliding block puzzle*.
- Such a puzzle asks if all polygons of a given collection of polygons can be free.

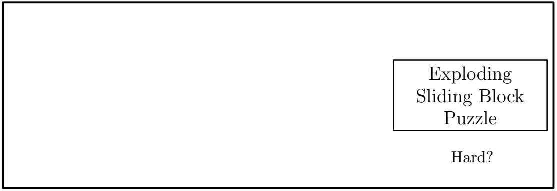


Interlocked Polygons

- If one allows removing polygons from the set, an interlocked set of polygons can become free.
- Removing polygons from the set cannot cause a free set to become interlocked.
- They use these properties to reduce solving a monotone boolean function to the interlocked polygon problem.

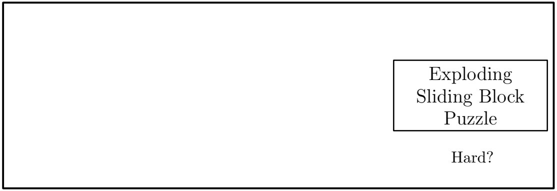


- The authors want to say something about the hardness of this new problem.



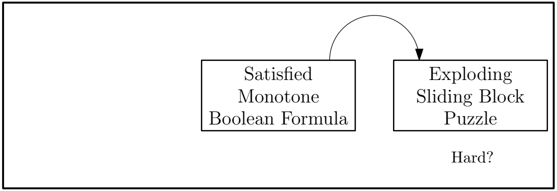
Hardness Reduction

- The authors want to say something about the hardness of this new problem.
- Similar to the lowerbound proofs they are going to reduce solving a known hard problem to solving this problem.



Hardness Reduction

- The authors want to say something about the hardness of this new problem.
- Similar to the lowerbound proofs they are going to reduce solving a known hard problem to solving this problem.
- We begin by considering reducing from an easy problem which is related to the problem they will eventually reduce to solving the Exploding Sliding Block Problem.



Satisfied Monotone Boolean Formula

- You are given a Monotone Boolean Formula and a set of assignments for the variables.
- (Monotone indicates that variables only appear as positive literals in the formula)
- This is a Satisfied Monotone Boolean Formula if the formula evaluates to TRUE for the given assignments of the variables.

$$((x_1 \wedge x_2) \vee x_3) \wedge (x_1 \vee x_3)$$

Satisfied Monotone Boolean Formula

- You are given a Monotone Boolean Formula and a set of assignments for the variables.
- (Monotone indicates that variables only appear as positive literals in the formula)
- This is a Satisfied Monotone Boolean Formula if the formula evaluates to TRUE for the given assignments of the variables.

x_1	x_2	x_3		$((x_1 \wedge x_2) \vee x_3) \wedge (x_1 \vee x_3)$

Satisfied Monotone Boolean Formula

- You are given a Monotone Boolean Formula and a set of assignments for the variables.
- (Monotone indicates that variables only appear as positive literals in the formula)
- This is a Satisfied Monotone Boolean Formula if the formula evaluates to TRUE for the given assignments of the variables.

x_1	x_2	x_3		$((x_1 \wedge x_2) \vee x_3) \wedge (x_1 \vee x_3)$
T	T	T		

Satisfied Monotone Boolean Formula

- You are given a Monotone Boolean Formula and a set of assignments for the variables.
- (Monotone indicates that variables only appear as positive literals in the formula)
- This is a Satisfied Monotone Boolean Formula if the formula evaluates to TRUE for the given assignments of the variables.

x_1	x_2	x_3	$((x_1 \wedge x_2) \vee x_3) \wedge (x_1 \vee x_3)$
T	T	T	T

Satisfied Monotone Boolean Formula

- You are given a Monotone Boolean Formula and a set of assignments for the variables.
- (Monotone indicates that variables only appear as positive literals in the formula)
- This is a Satisfied Monotone Boolean Formula if the formula evaluates to TRUE for the given assignments of the variables.

x_1	x_2	x_3	$((x_1 \wedge x_2) \vee x_3) \wedge (x_1 \vee x_3)$
T	T	T	T
T	T	F	

Satisfied Monotone Boolean Formula

- You are given a Monotone Boolean Formula and a set of assignments for the variables.
- (Monotone indicates that variables only appear as positive literals in the formula)
- This is a Satisfied Monotone Boolean Formula if the formula evaluates to TRUE for the given assignments of the variables.

x_1	x_2	x_3	$((x_1 \land x_2) \lor x_3) \land (x_1 \lor x_3)$
T	T	T	T
T	T	F	T

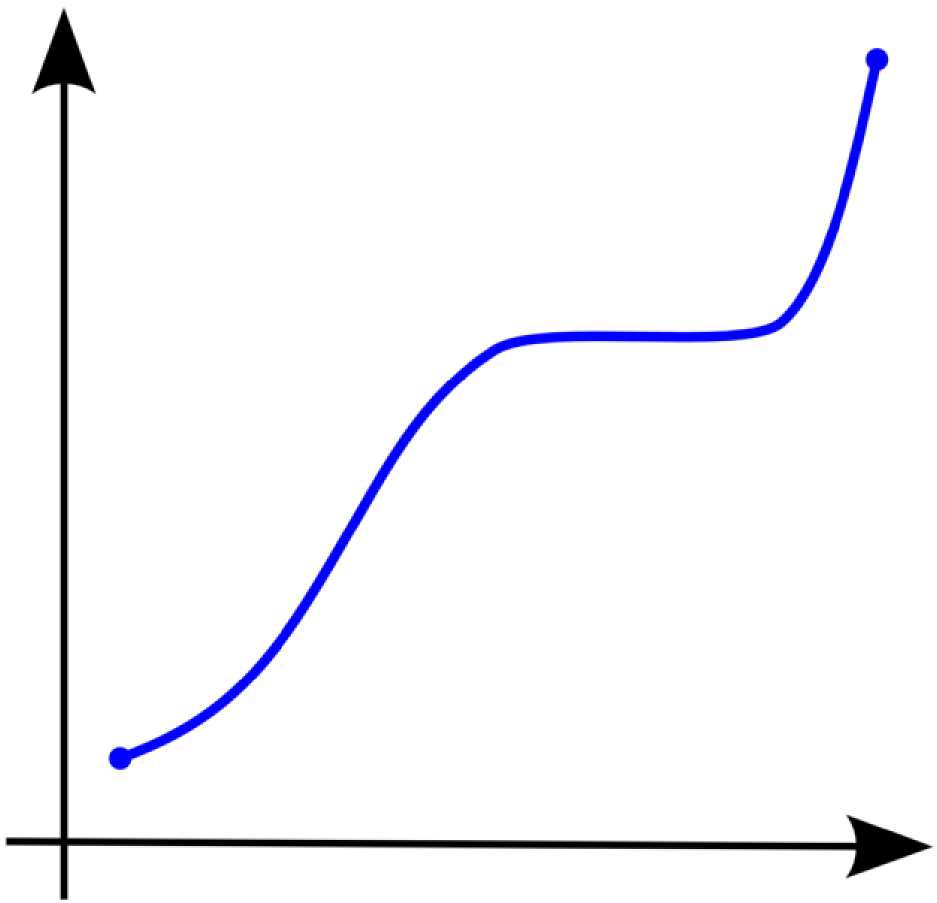
Satisfied Monotone Boolean Formula

- You are given a Monotone Boolean Formula and a set of assignments for the variables.
- (Monotone indicates that variables only appear as positive literals in the formula)
- This is a Satisfied Monotone Boolean Formula if the formula evaluates to TRUE for the given assignments of the variables.

x_1	x_2	x_3	$((x_1 \wedge x_2) \vee x_3) \wedge (x_1 \vee x_3)$
T	T	T	T
T	T	F	T
T	F	T	T
T	F	F	F
F	T	T	T
F	T	F	F
F	F	T	T
F	F	F	F

Monotone Boolean Functions

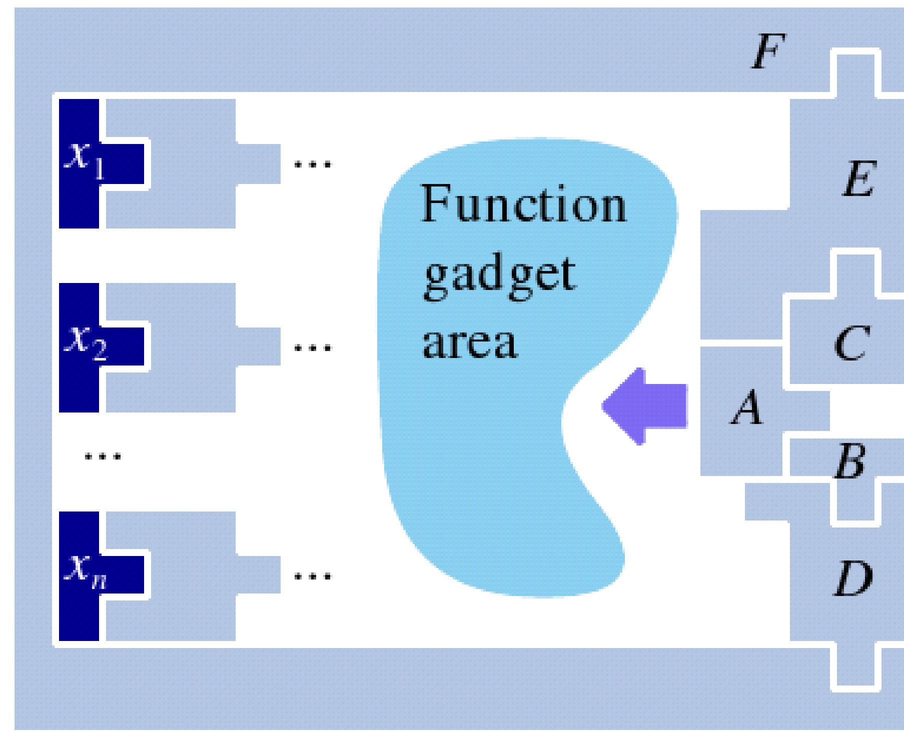
- What is a Monotone Boolean Function?
  - All variables appear as positive literals.
  - (Only ANDs and ORs allowed.)
- Thus, a variable being assigned as true cannot cause the function to become false.



$$f(x_1, x_2, x_3) = ((x_1 \wedge x_2) \vee x_3) \wedge (x_1 \vee x_3)$$

Reduction Gadgets: Frame

- All of the gadgets are constructed in a frame.
- This frame ensures that the set of polygons can be separated iff the polygon A can be moved left.
- The variables of f appear as polygons on the left hand side of the frame.
- Removing them corresponds to setting them to true.

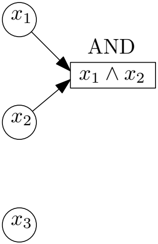


Reduction Graph

$$((x_1 \wedge x_2) \vee x_3) \wedge (x_1 \vee x_3)$$

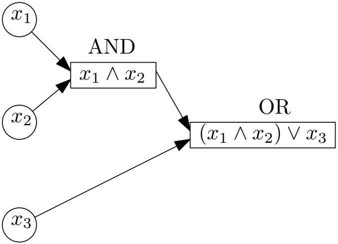
Reduction Graph

$$((x_1 \wedge x_2) \vee x_3) \wedge (x_1 \vee x_3)$$



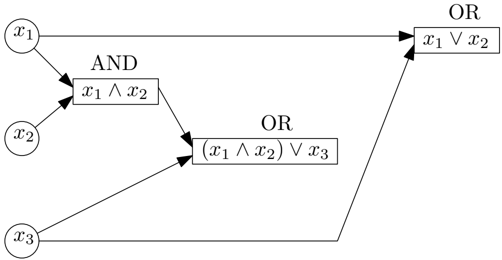
Reduction Graph

$$((x_1 \wedge x_2) \vee x_3) \wedge (x_1 \vee x_3)$$



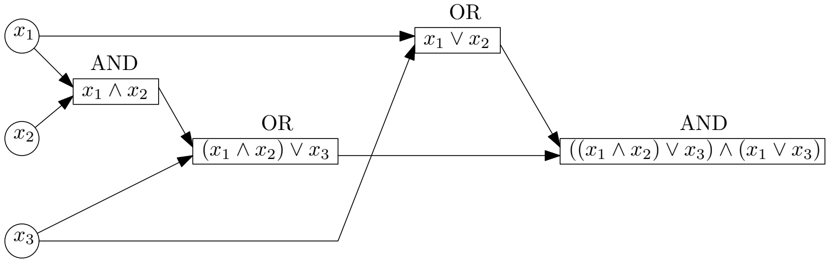
Reduction Graph

$$((x_1 \wedge x_2) \vee x_3) \wedge (x_1 \vee x_3)$$



Reduction Graph

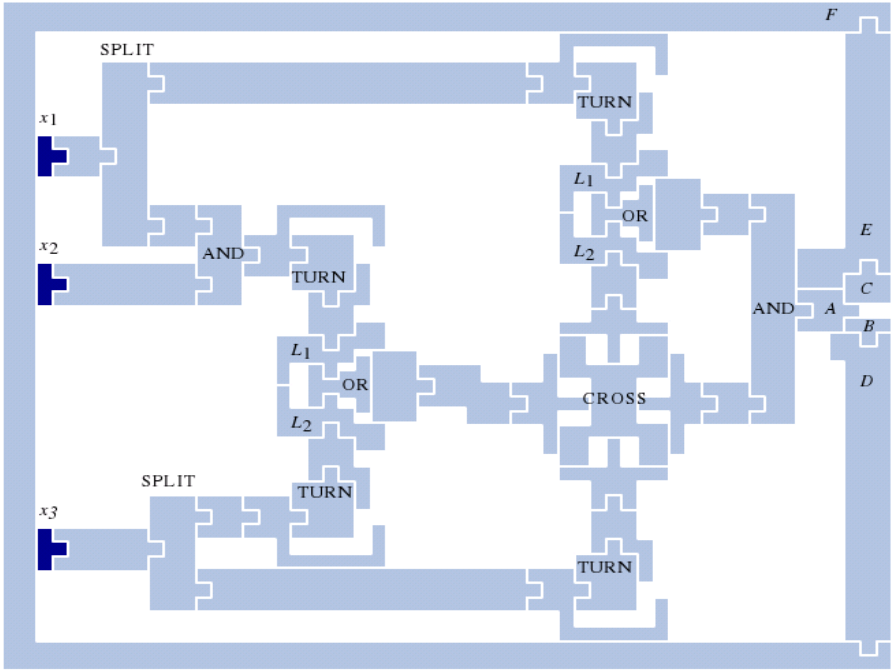
$$((x_1 \wedge x_2) \vee x_3) \wedge (x_1 \vee x_3)$$



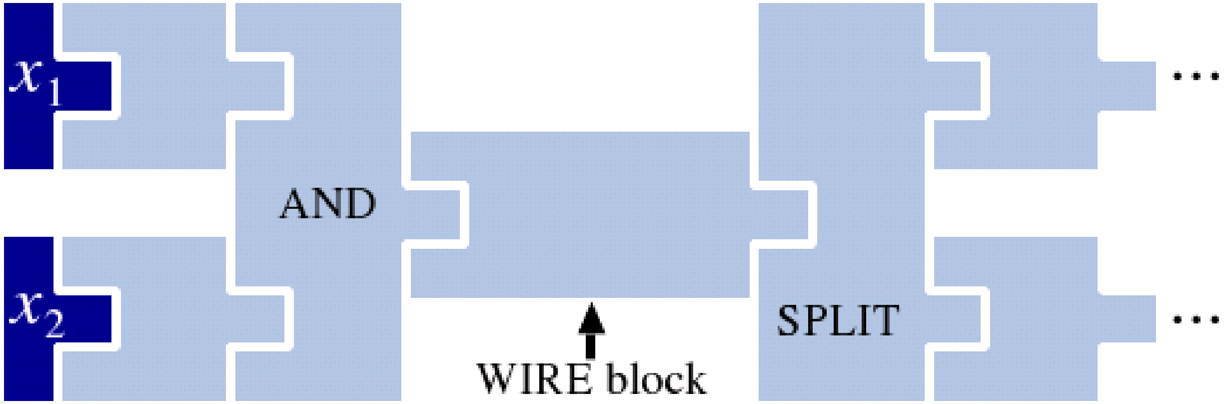
Reduction Gadgets: Frame

- All of the gadgets are constructed in a frame.
- This frame ensures that the set of polygons can be separated iff the polygon A can be moved left.
- The variables of f appear as polygons on the left hand side of the frame.
- Removing them corresponds to setting them to true.

Figure 8. A construction for $f(x_1, x_2, x_3) = ((x_1 \wedge x_2) \vee x_3) \wedge (x_1 \vee x_3)$.

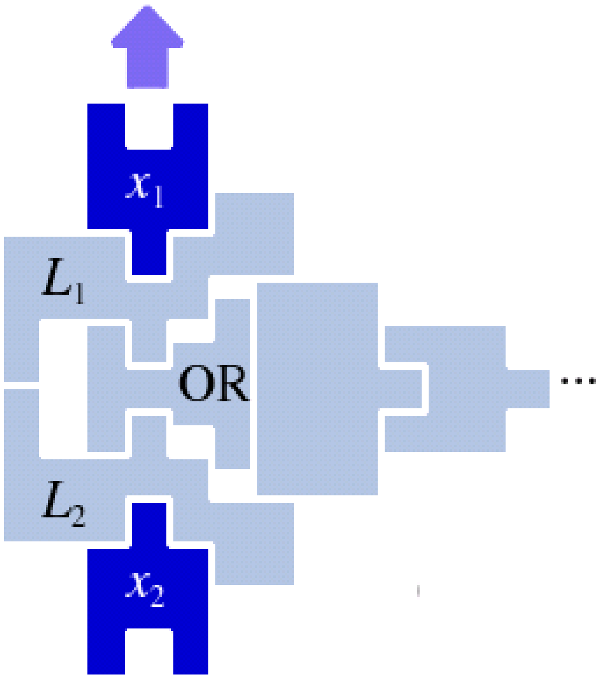


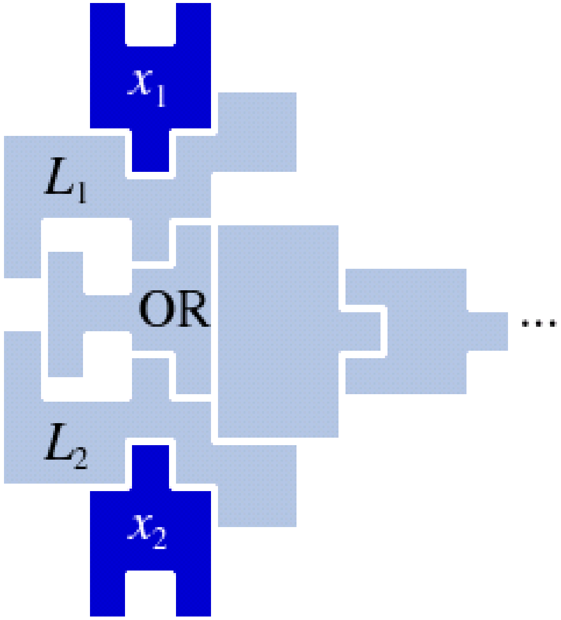
Reduction Gadgets: And/Split



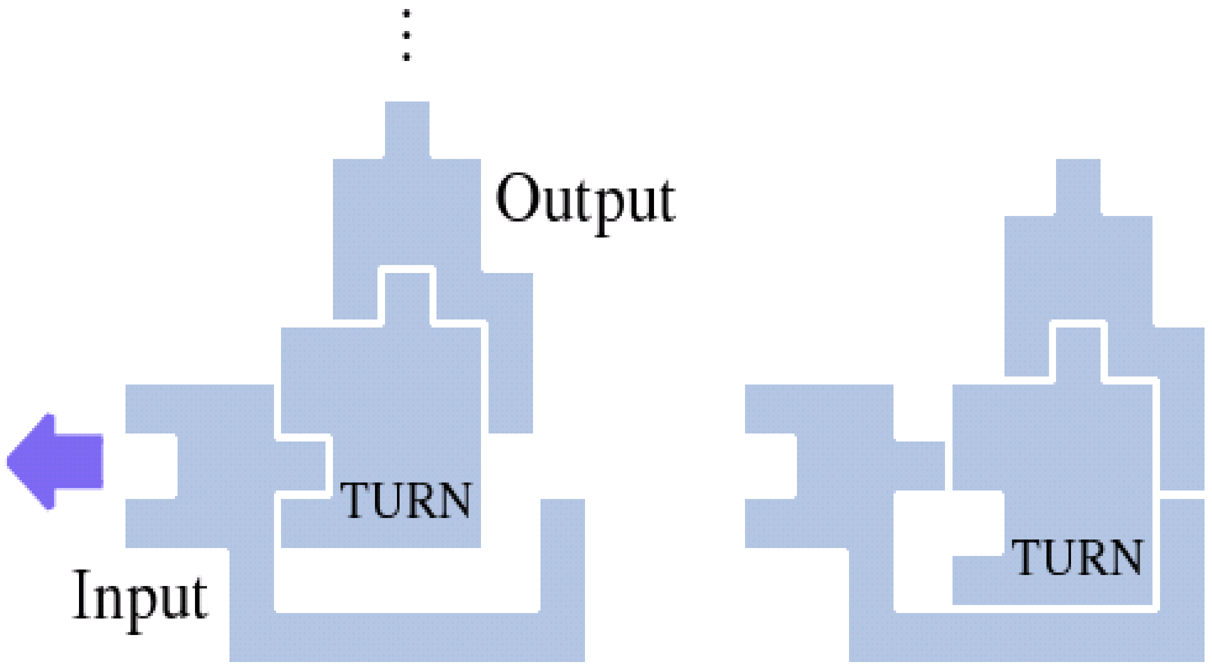
- The And and Split gadgets are mirrors of each other.

Reduction Gadgets: Or

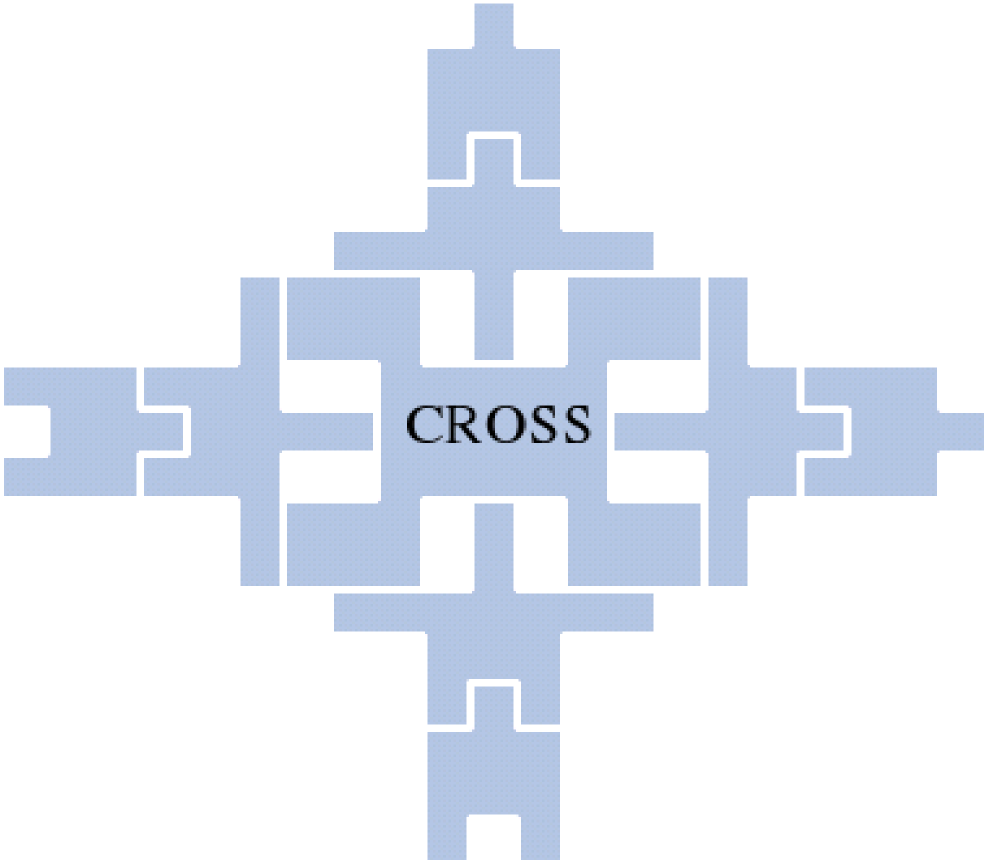




Reduction Gadgets: Turn



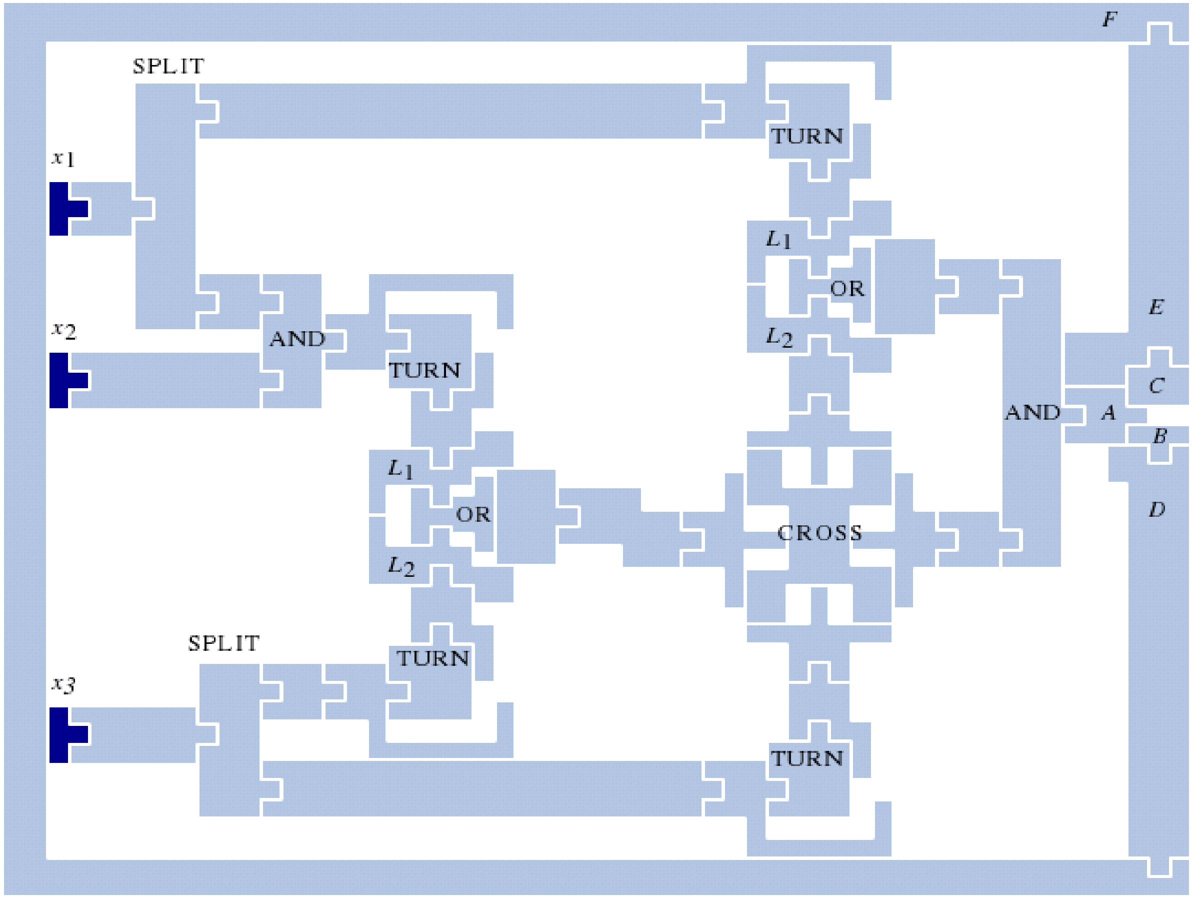
Reduction Gadgets: Crossover



- Note that for all of these gadgets the operations are reversible. (i.e. can be undone)

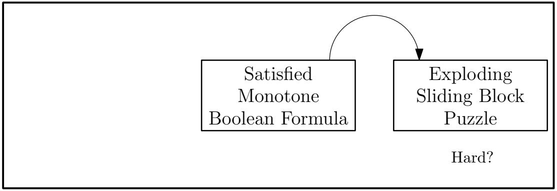
Reduction Gadgets

Figure 8. A construction for $f(x_1, x_2, x_3) = ((x_1 \wedge x_2) \vee x_3) \wedge (x_1 \vee x_3)$.

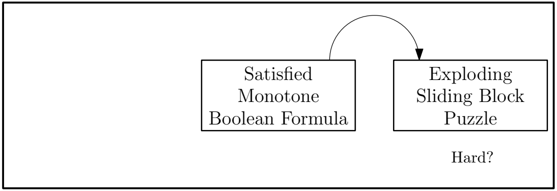


Hardness Reduction

- Of course, Monotone Boolean Formula is an easy problem to solve so this doesn't say much about the difficulty of the Exploding Sliding Block Problem to reduce from it.

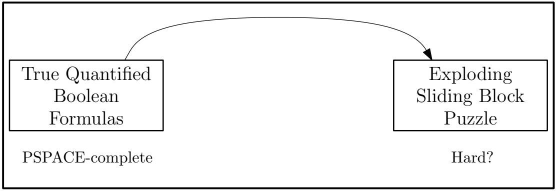


- Of course, Monotone Boolean Formula is an easy problem to solve so this doesn't say much about the difficulty of the Exploding Sliding Block Problem to reduce from it.
- We now consider a more difficult problem: True Quantified Boolean Formulas



Hardness Reduction

- Of course, Monotone Boolean Formula is an easy problem to solve so this doesn't say much about the difficulty of the Exploding Sliding Block Problem to reduce from it.
- We now consider a more difficult problem: True Quantified Boolean Formulas



True Quantified Boolean Formulas

- True Quantified Boolean Formulas contain universal and existential quantification instead of having a fixed assignment of the variables.

$$\forall x_1 \exists x_2 \forall x_3 : ((x_1 \wedge x_2) \vee x_3) \wedge (x_1 \vee x_3)$$

True Quantified Boolean Formulas

- True Quantified Boolean Formulas contain universal and existential quantification instead of having a fixed assignment of the variables.

$\forall x_1 \exists x_2 \forall x_3 : ((x_1 \wedge x_2) \vee x_3) \wedge (x_1 \vee x_3)$

x_1	x_2	x_3	$((x_1 \wedge x_2) \vee x_3) \wedge (x_1 \vee x_3)$
T	T	T	T
T	T	F	T
T	F	T	T
T	F	F	F
F	T	T	T
F	T	F	F
F	F	T	T
F	F	F	F

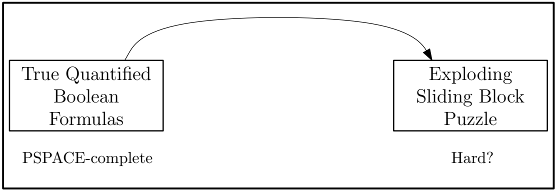
True Quantified Boolean Formulas

- True Quantified Boolean Formulas contain universal and existential quantification instead of having a fixed assignment of the variables.
- $\forall x_1 \exists x_2 \forall x_3 : ((x_1 \land x_2) \lor x_3) \land (x_1 \lor x_3)$ evaluates to FALSE.

$\forall x_1 \exists x_2 \forall x_3 : ((x_1 \land x_2) \lor x_3) \land (x_1 \lor x_3)$

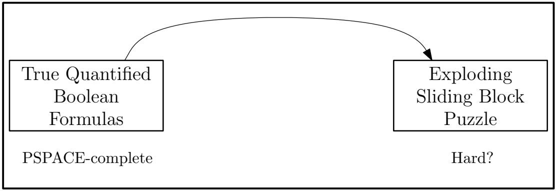
x_1	x_2	x_3	$((x_1 \land x_2) \lor x_3) \land (x_1 \lor x_3)$
T	T	T	T
T	T	F	T
T	F	T	T
T	F	F	F
F	T	T	T
F	T	F	F
F	F	T	T
F	F	F	F

- On last wrinkle.



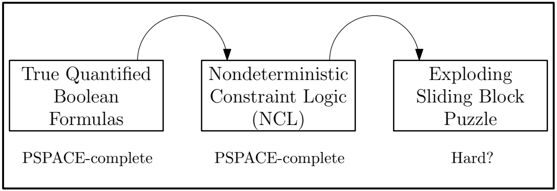
Hardness Reduction

- On last wrinkle.
- They reduce from a True Quantified Boolean Formulas to a problem, NCL, which they then reduce to the Exploding Sliding Block Problem.



Hardness Reduction

- On last wrinkle.
- They reduce from a True Quantified Boolean Formulas to a problem, NCL, which they then reduce to the Exploding Sliding Block Problem.



Monotone Boolean Functions

- This completes the reduction.
  - Interesting gadgets but is this problem at all hard?
  - This reduction may suggest that the exploding sliding block puzzle is easy.

Monotone Boolean Functions

- This completes the reduction.
  - Interesting gadgets but is this problem at all hard?
  - This reduction may suggest that the exploding sliding block puzzle is easy.
- The authors go on to show that the exploding sliding block puzzle is PSPACE-complete.

Monotone Boolean Functions

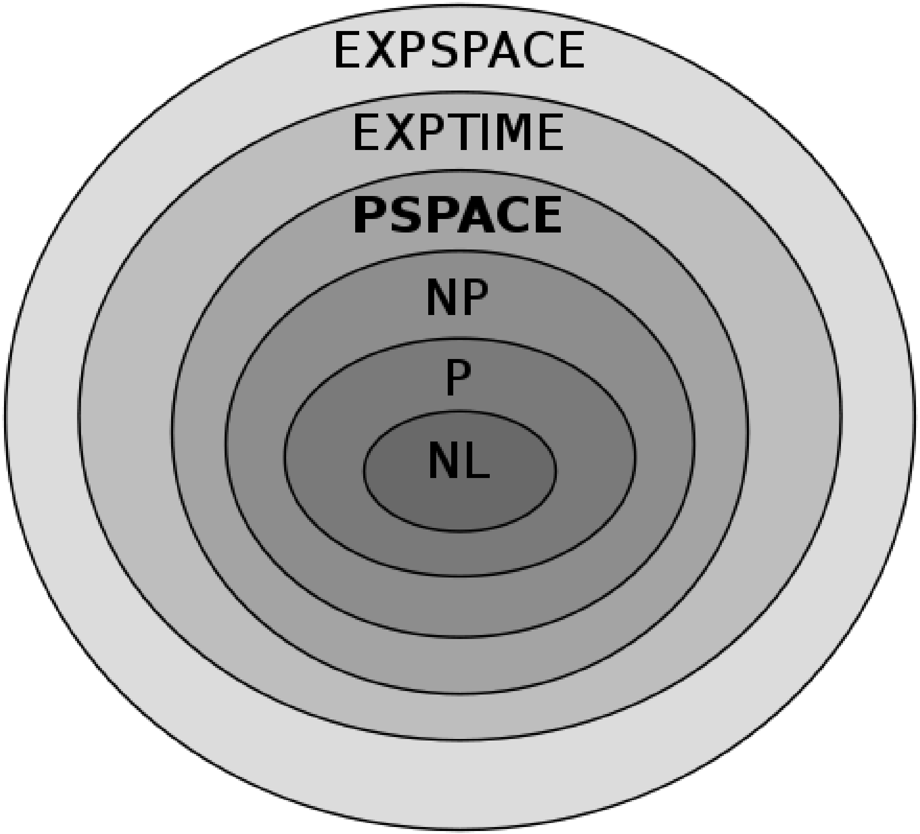
- This completes the reduction.
  - Interesting gadgets but is this problem at all hard?
  - This reduction may suggest that the exploding sliding block puzzle is easy.
- The authors go on to show that the exploding sliding block puzzle is PSPACE-complete.
  - Really?

Outline

- Introduction:
  - Sliding Block Puzzles
  - Interlocked Polygons
  - Monotone Boolean Functions
- PSPACE-completeness
- Nondeterministic Constraint Logic
  - Introduction
    - True Quantified Boolean Formulas (TQBF)
  - Proof Idea

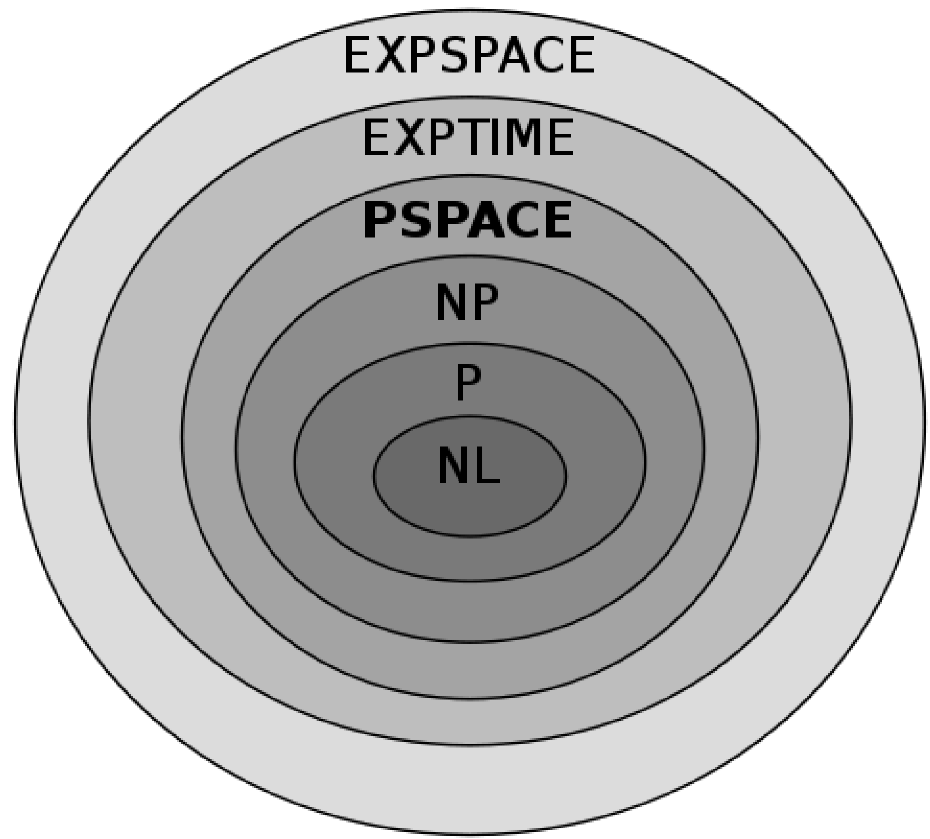
PSPACE-Complete?

- PSPACE is the class of problems which can be decided in polynomial space.
- NP is contained in it (may not be strict).
- Intuitively, PSPACE-complete problems are harder than NP-complete problems.



PSPACE-Complete?

- The details of the PSPACE-Completeness proof are largely absent from this paper.
- The authors reduce from a problem called Nondeterministic Constraint Logic (NCL).
- We give a quick overview of this problem next and examine why it is hard to solve.



Outline

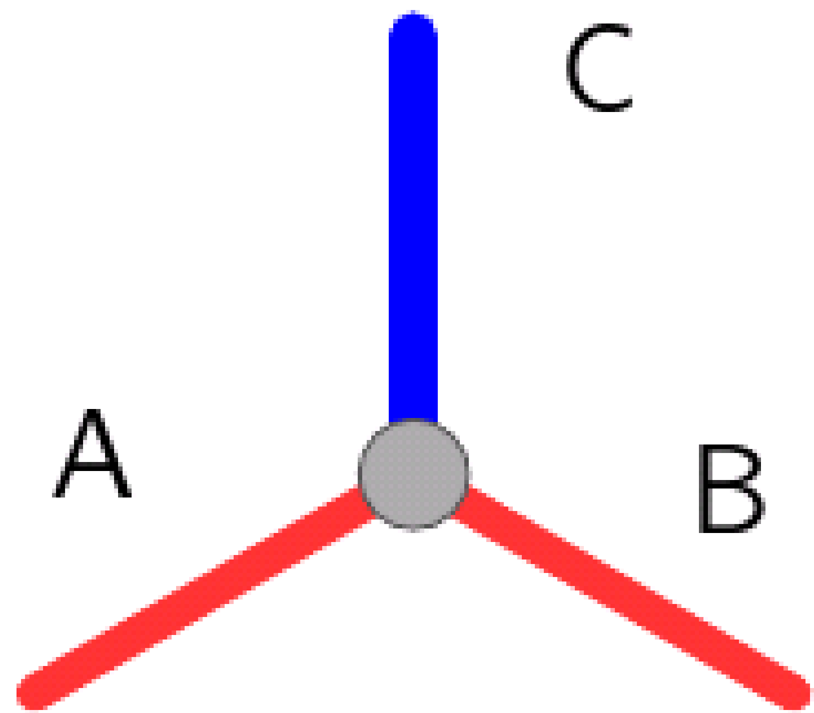
- Introduction:
  - Sliding Block Puzzles
  - Interlocked Polygons
  - Monotone Boolean Functions
- PSPACE-completeness
- Nondeterministic Constraint Logic
  - Introduction
    - True Quantified Boolean Formulas (TQBF)
  - Proof Idea

Nondeterministic Constraint Logic

- Idea:
  - Given a directed graph.
  - Each vertex has a weight requirement it must meet.
  - Each edge has a weight that it contributes to one of the two vertices it's adjacent to.
  - The direction of the edge determines which vertex gets the weight.
  - The direction of edges can be flipped if the weight requirement is still met on its vertices after the flip.
- Goal:
  - Decide if a given edge e in the graph can be flipped.
  - (through a sequence of valid flips of other edges)

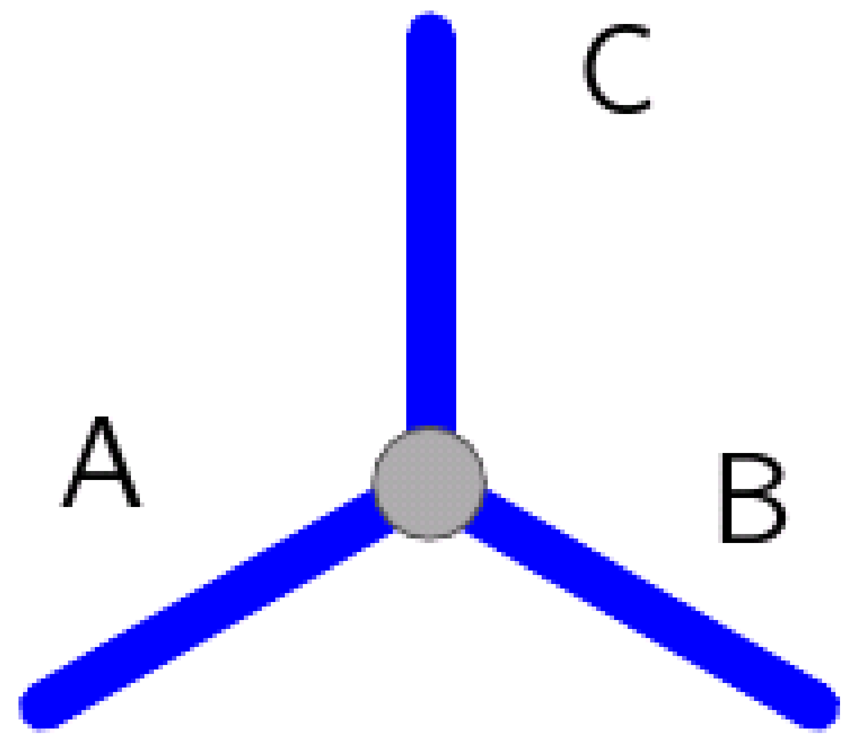
Nondeterministic Constraint Logic

- They further restrict this such that:
  - each vertex has a weight requirement of 2.
  - each edge has weight 1 or 2 (red or blue respectively).
- The structure to the right behaves similar to an AND.

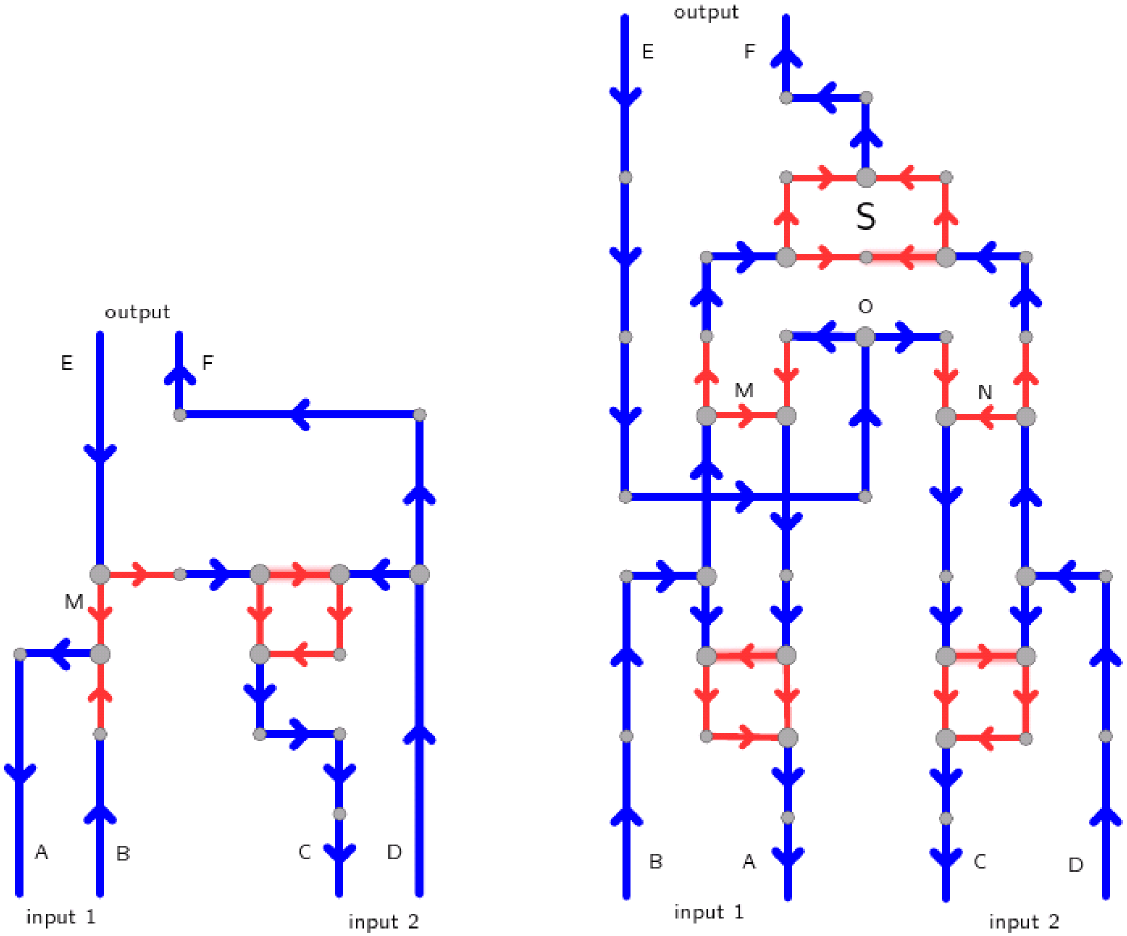


Nondeterministic Constraint Logic

- They further restrict this such that:
  - each vertex has a weight requirement of 2.
  - each edge has weight 1 or 2 (red or blue respectively).
- The structure to the right behaves similar to an OR.



Nondeterministic Constraint Logic

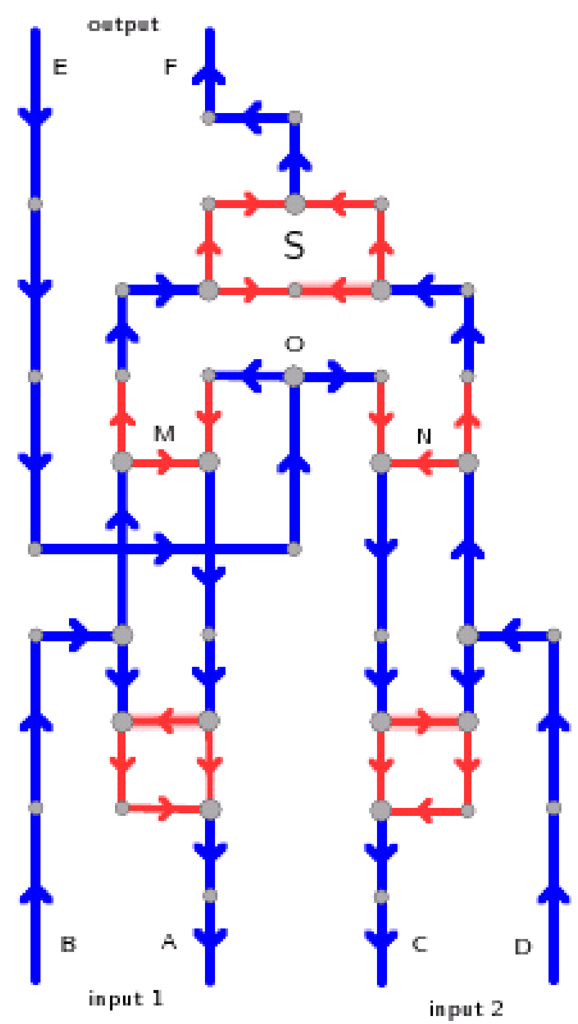


(a) AND′ gadget (b) OR′ gadget

- Naturally these can be more complex with many vertices.

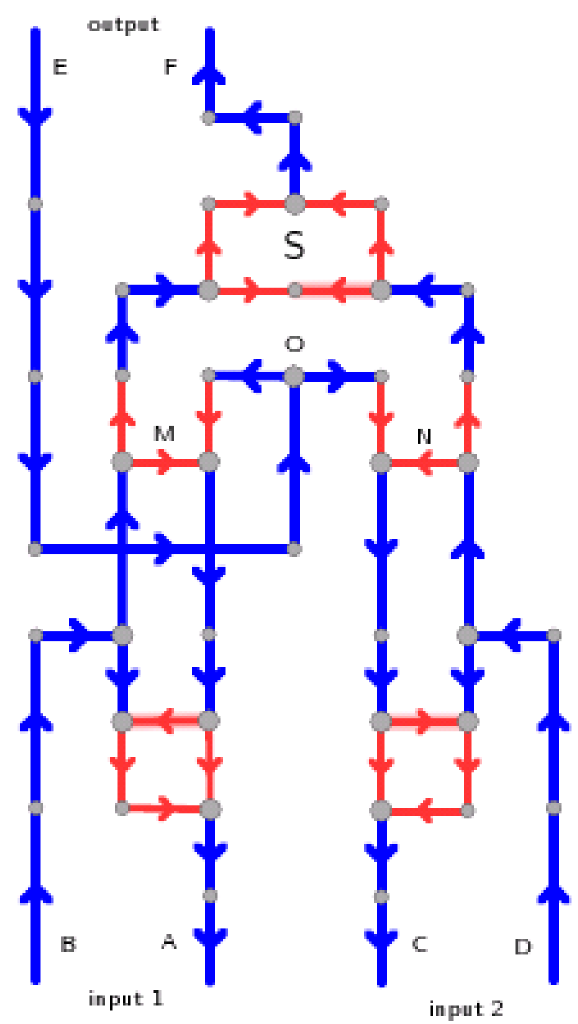
Nondeterministic Constraint Logic

- They want to show that NCL is PSPACE-complete
- To show this they reduce from True Quantified Boolean Formulas (TQBF).
- Next we introduce this problem.



Nondeterministic Constraint Logic

- They want to show that NCL is PSPACE-complete
- To show this they reduce from True Quantified Boolean Formulas (TQBF).
- Next we introduce this problem.
- (note: this only gives the hardness result but the other half of the completeness proof is trivial)



True Quantified Boolean Formulas (TQBF)

- Deciding if a fully quantified boolean formula is true is PSPACE-complete.
- This serves as the canonical complete problem for PSPACE.
- TQBF = { <F> : F is a true fully quantified boolean formula }

$$\forall x \exists y \forall w \cdots \exists z \left[(x \vee y) \wedge \cdots \wedge (\overline{z} \vee x \vee \overline{w}) \right]$$

TQBF to NCL Reduction

- To do this they create gadgets which emulate existential and universal quantification.
- This is possible do to the **reversible** nature of computations using NCL.

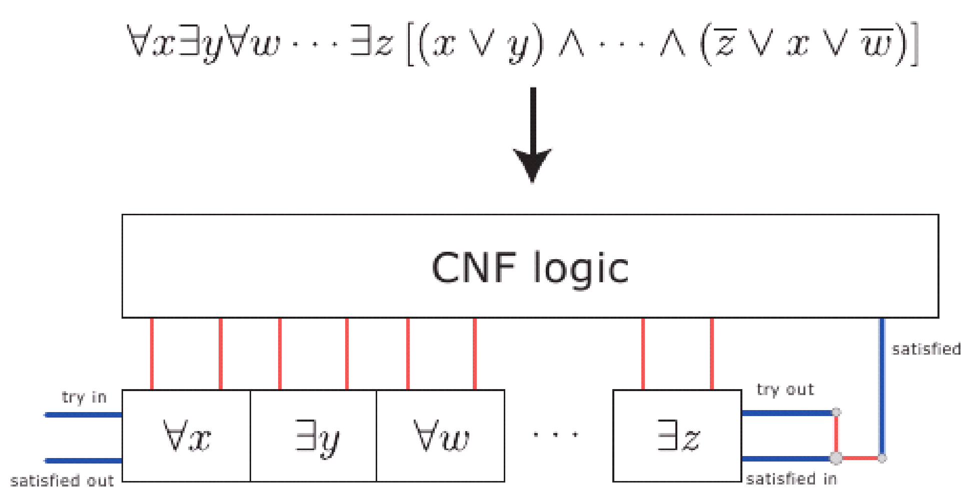
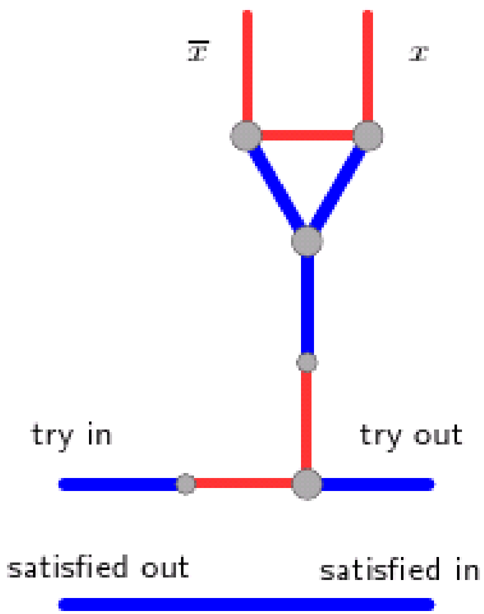
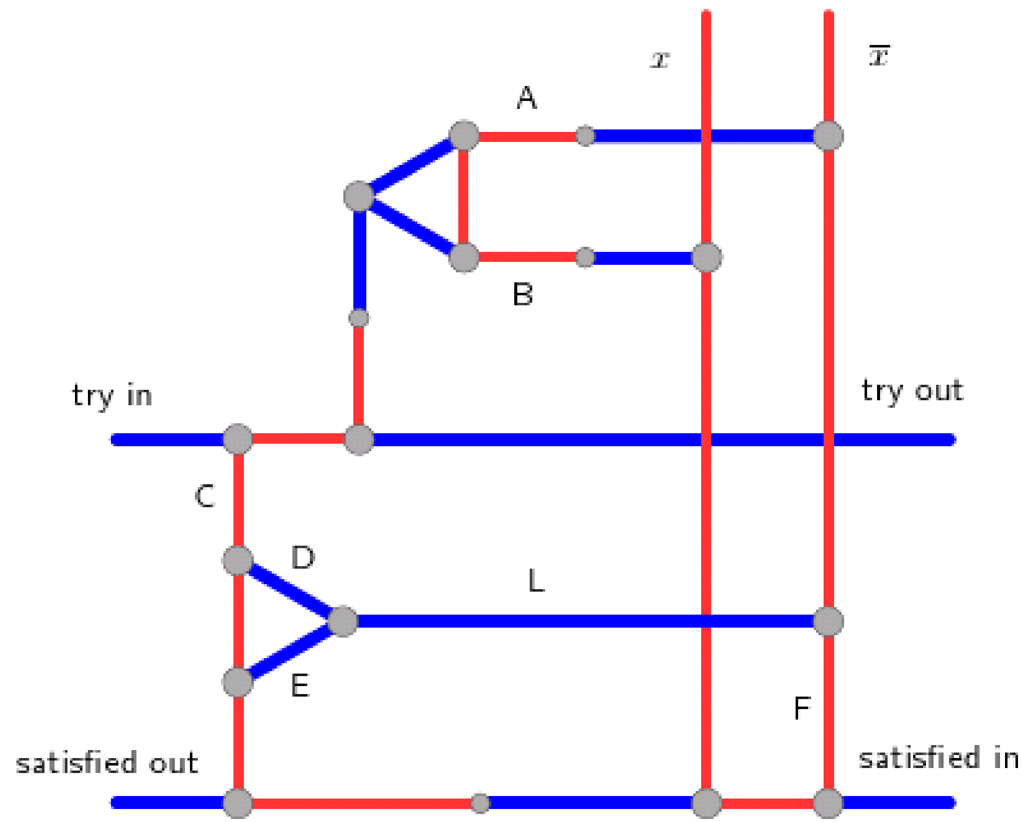


Figure 5-4: Schematic of the reduction from Quantified Boolean Formulas to NCL.

TQBF to NCL Reduction



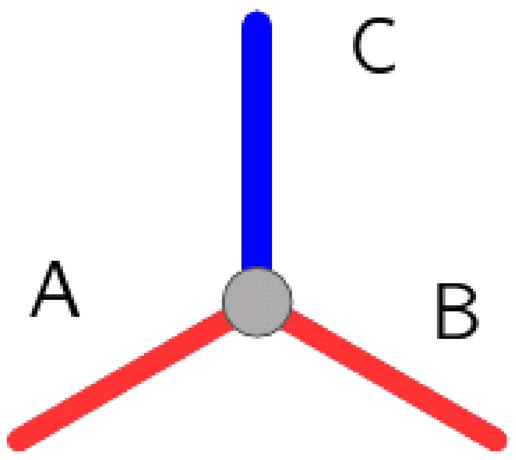
(a) Existential quantifier

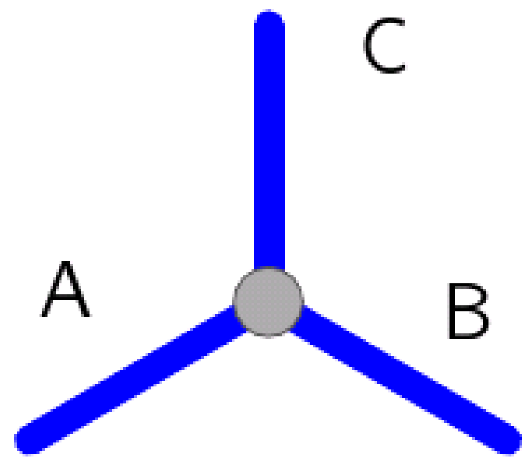


(b) Universal quantifier

AND/OR NCL Variant

- Interestingly they can show that all of their gadgets can be built using only the simple AND and OR gadgets.
- Thus interest in problems which simulate monotone boolean functions.
- These problems only need to emulate these two AND and OR gadgets to reduce to NCL.



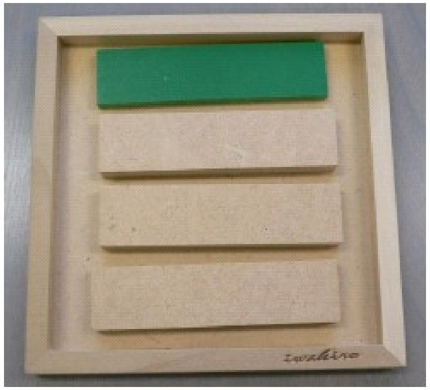


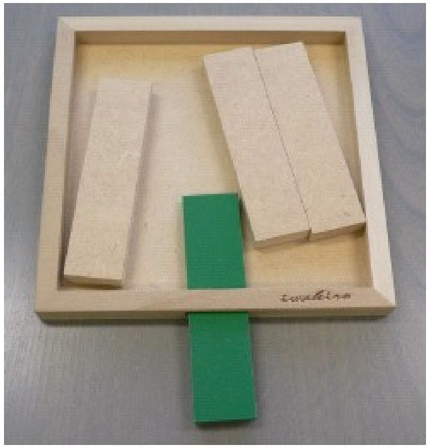
Summary of Proof

- Exploding Sliding Block Puzzle
- ←
- AND/OR NCL
- ←
- TQBF

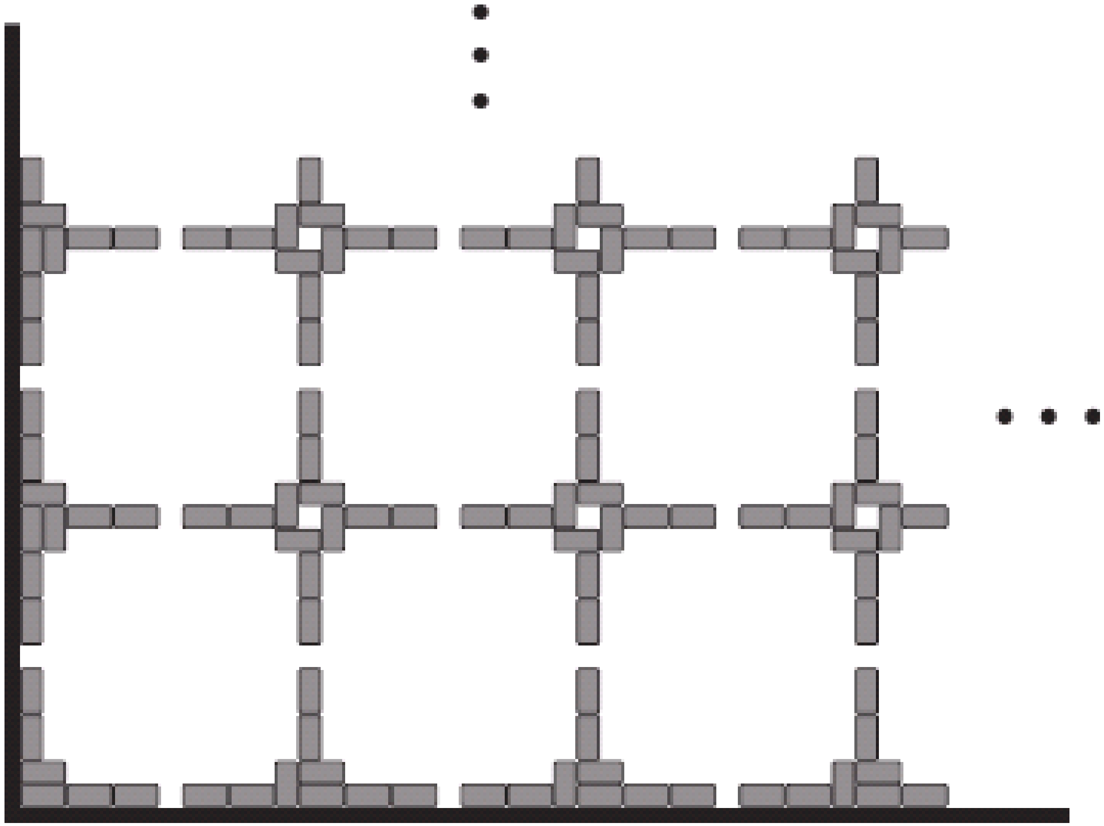
Variants of Interlocked Polygons

- Using the AND/OR NCL reduction one can prove PSPACE-Completeness for even simpler variants of the Exploding Sliding Block Puzzle.
- In particular the authors can show that the problem is hard even when all blocks are rectangles except one.

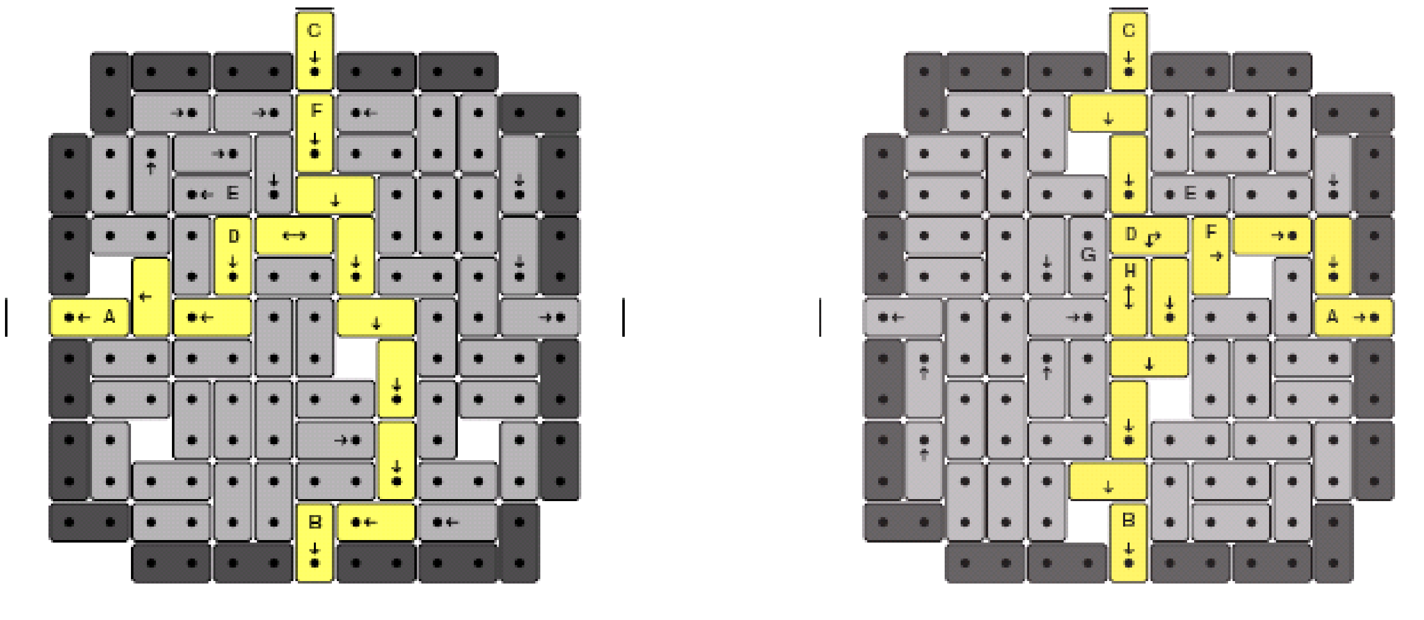




Variants of Interlocked Polygons



Variants of Interlocked Polygons



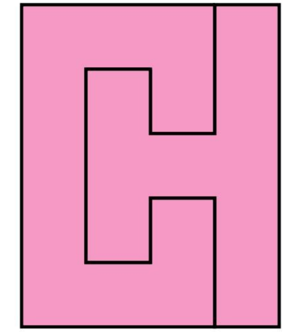
(a) AND

(b) Protected OR

Conclusions

- Deciding if polygons are interlocked yields a surprisingly difficult problem.
- The framework for proving PSPACE-completeness with NCL is impressively simple.
- NCL was used to show many such simple games are PSPACE-complete. So many that I couldn't even begin to cover them or all the various extensions to NCL.

Thank You! Questions?



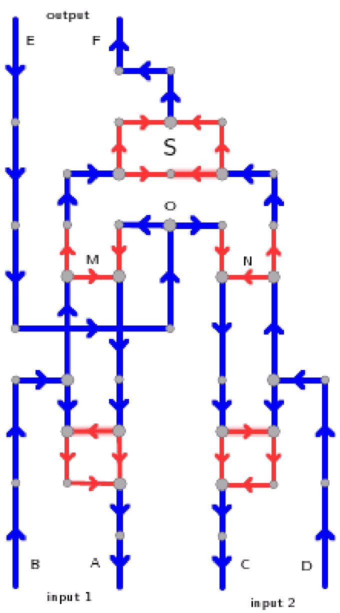
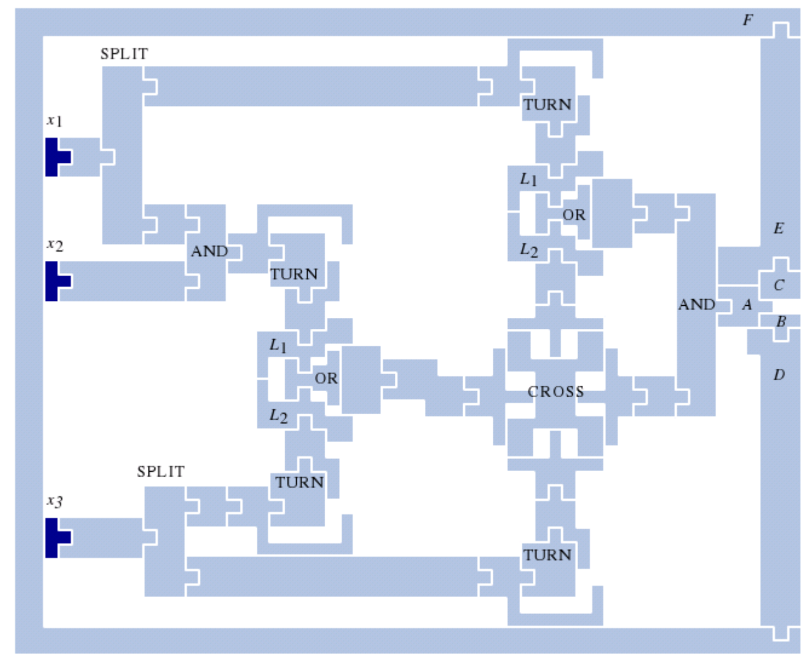
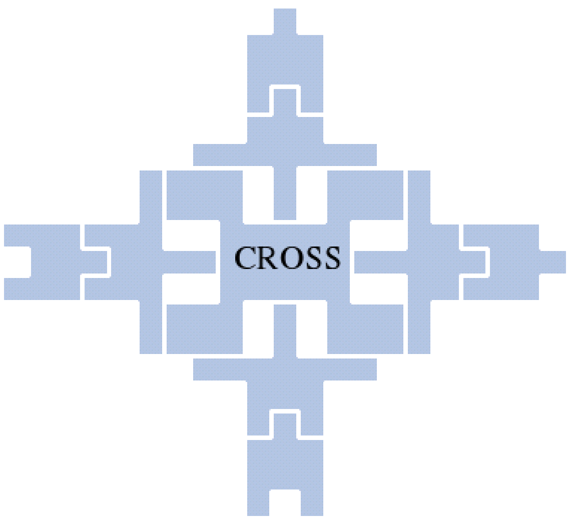
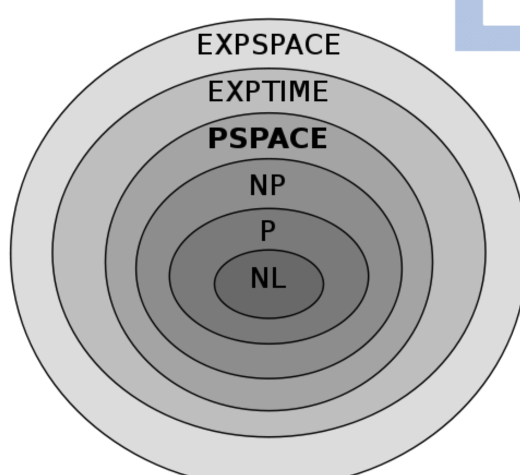
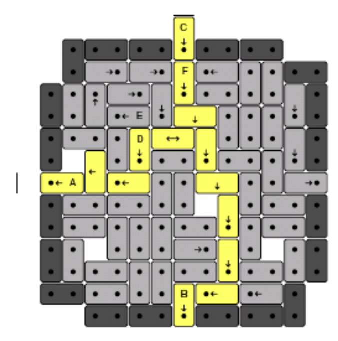


Figure 8. A construction for $f(x_1, x_2, x_3) = ((x_1 \wedge x_2) \vee x_3) \wedge (x_1 \vee x_3)$.

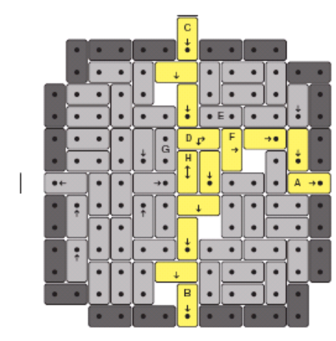






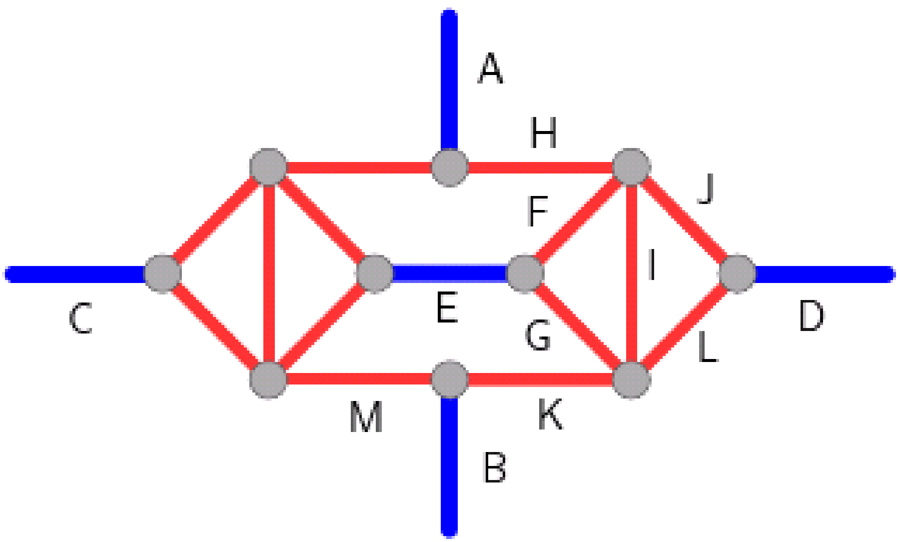


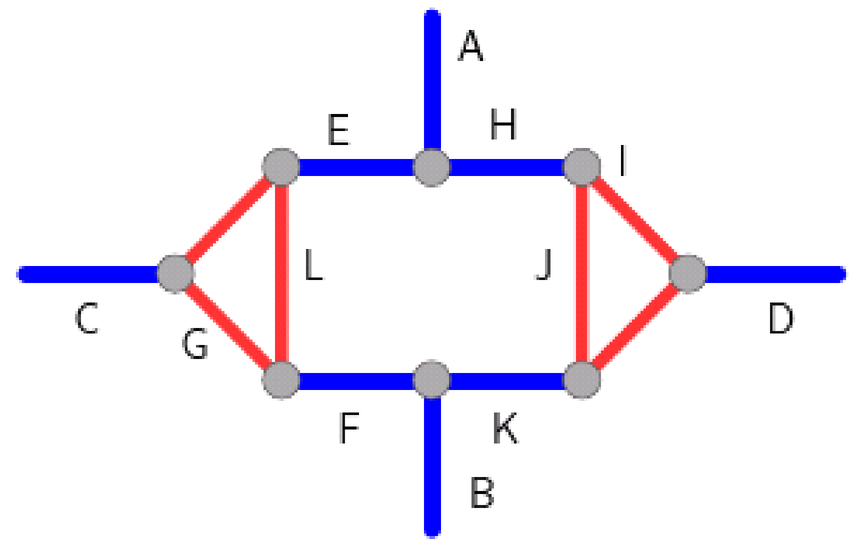
(a) AND



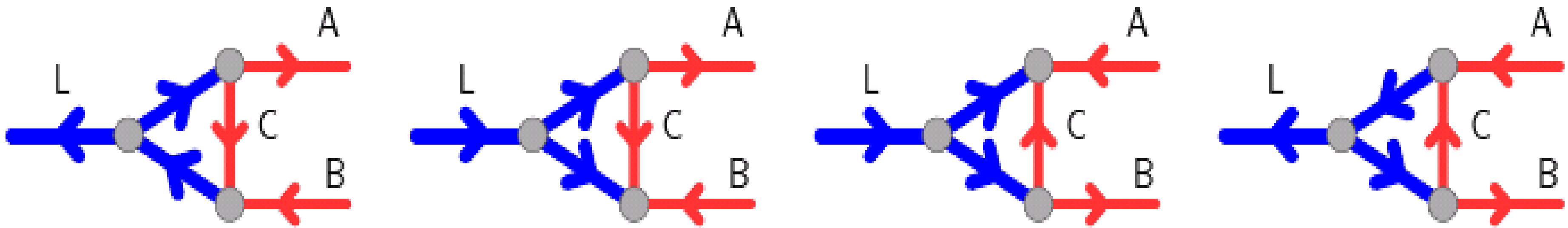
(b) Protected OR

NCL Cross





NCL Latch



(a) Locked, A active (b) Unlocked, A active (c) Unlocked, B active (d) Locked, B active

Figure 5-6: Latch gadget, transitioning from state A to state B.