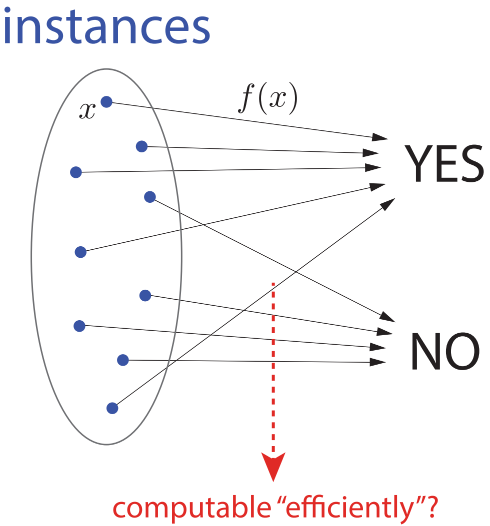
Lesson 11. The Complexity of Video Games

I628E – Information Processing Theory

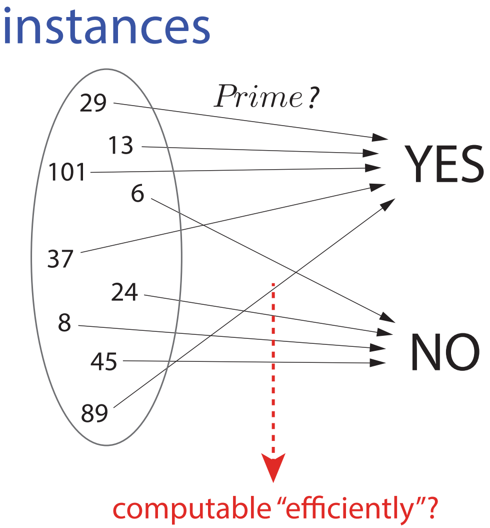
Giovanni Viglietta
johnny@jaist.ac.jp

JAIST – January 27, 2020

Decision problems



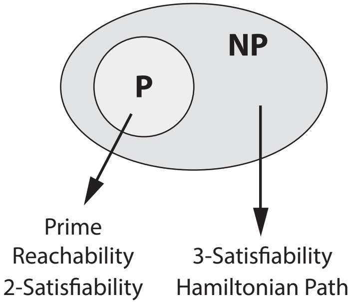
Example: Prime?



Basic complexity classes

P: problems decidable in polynomial time (i.e., "efficiently")

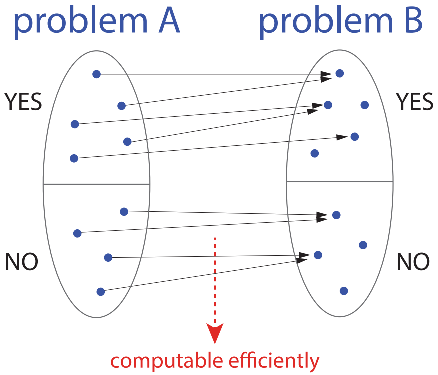
NP: problems verifiable in polynomial time, given a *certificate*



Observation: $\mathbf{P} \subseteq \mathbf{NP}$

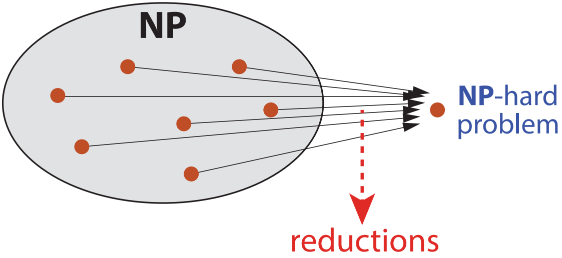
Open problem: is $\mathbf{P} \subsetneq \mathbf{NP}$ or is $\mathbf{P} = \mathbf{NP}$?

Reductions



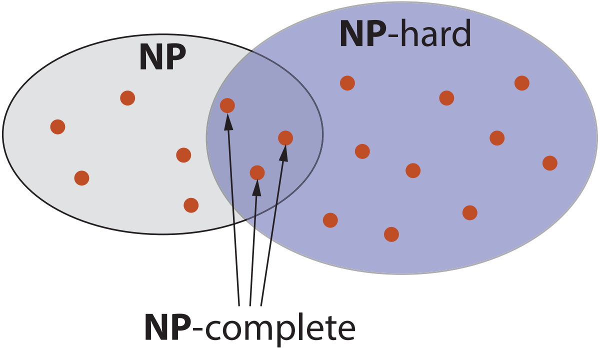
If A is reducible to B and we can solve B efficiently, then we can solve A efficiently (given an instance of A, apply the reduction, and solve the resulting instance of B)

NP-hard problems



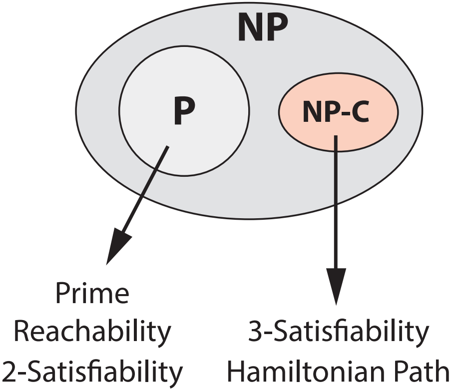
A problem is NP-hard if all the problems in NP can be reduced to it (i.e., it is at least as hard as every NP problem)

NP-complete problems



A problem is NP-complete if it is NP-hard and it is also in NP (i.e., it is the "hardest problem" in NP)

NP-complete problems: examples



Observation: if one NP-complete problem is solvable in polynomial time, then $P = NP = NP$-complete

Perhaps the most important NP-complete problem:

3-SAT
Input: a Boolean expression of the form:

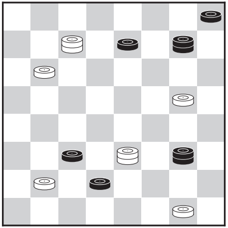
positive literals negative literals

$$(x \wedge \bar{y} \wedge z) \vee \underbrace{(\bar{x} \wedge z \wedge w)}_{\text{clause}} \vee (y \wedge \bar{z} \wedge \bar{w}) \vee (\bar{x} \wedge y \wedge w)$$

Output: YES if there is a truth assignment to the variables that makes the expression true. NO otherwise.

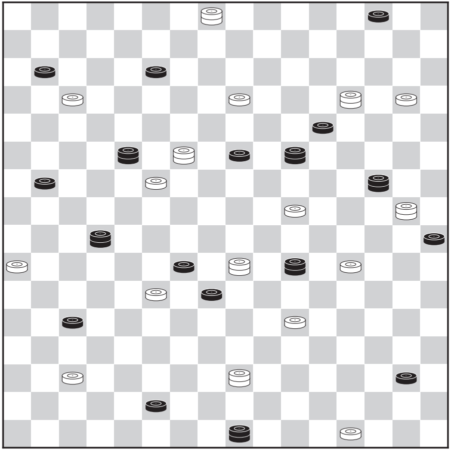
Generalizing (video) games

Guideline: "Expand the board, but do not change the rules"



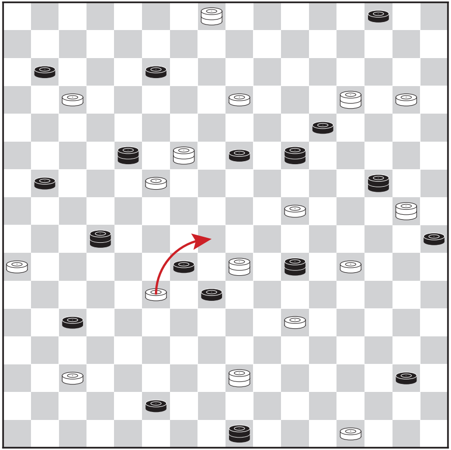
Generalizing (video) games

Guideline: “Expand the board, but do not change the rules”

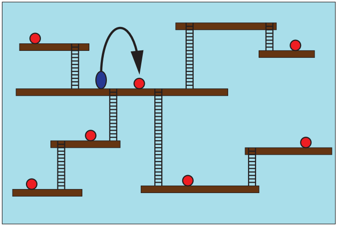


Generalizing (video) games

Guideline: "Expand the board, but do not change the rules"



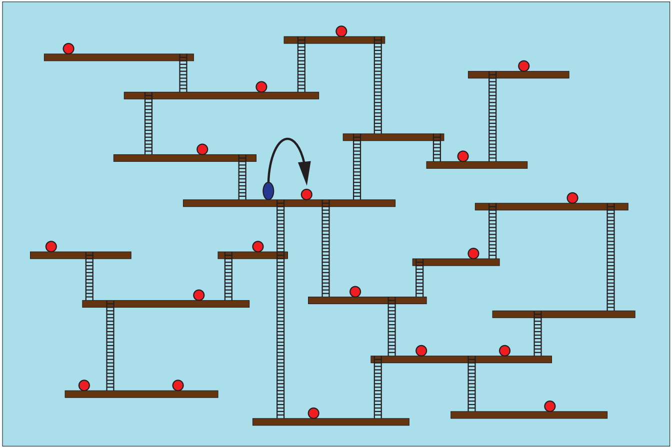
Guideline: "Expand the board, but do not change the rules"



Add more platforms and enemies, but do not change the "physics"

Generalizing (video) games

Guideline: "Expand the board, but do not change the rules"

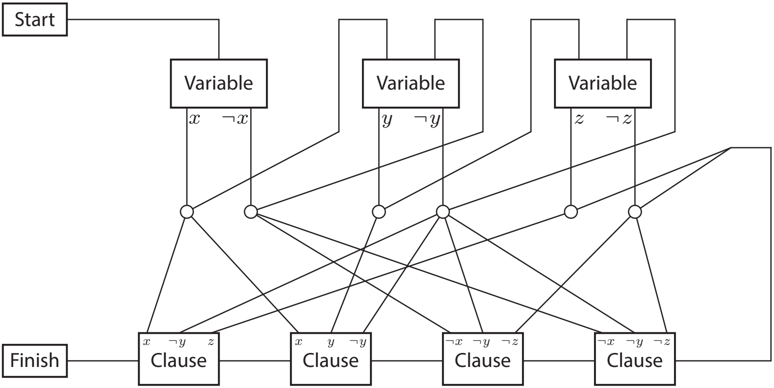


Add more platforms and enemies, but do not change the "physics"

Basic decision problem: is a given level beatable?

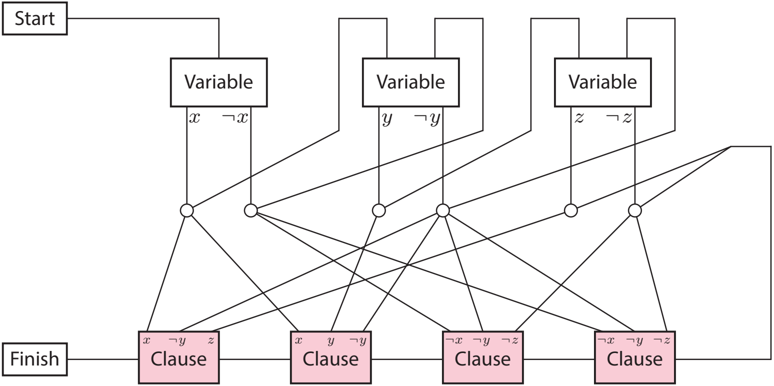
NP-hardness framework for platform games

Reduction from 3-SAT to a generic platform game



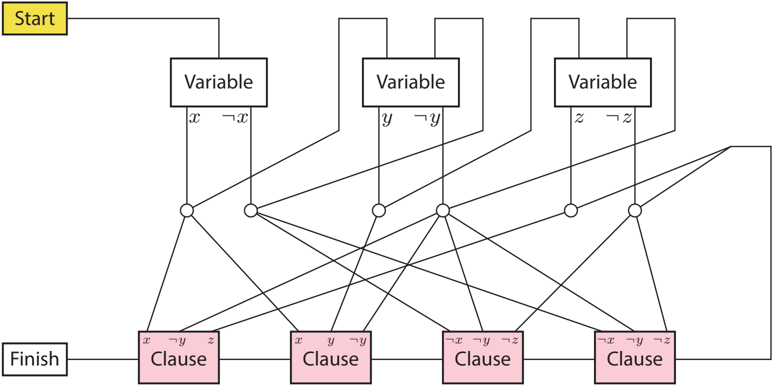
NP-hardness framework for platform games

Reduction from 3-SAT to a generic platform game



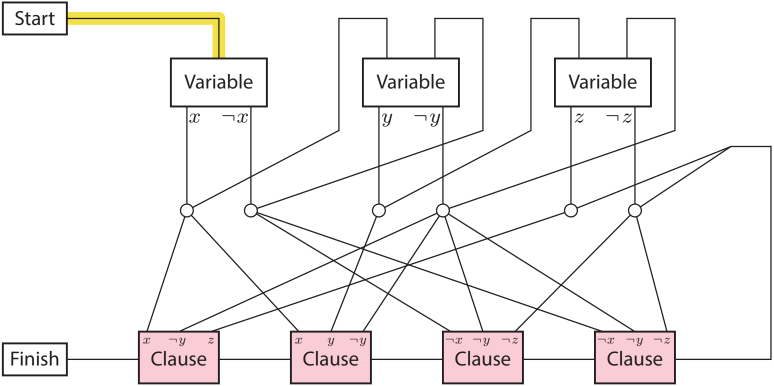
NP-hardness framework for platform games

Reduction from 3-SAT to a generic platform game

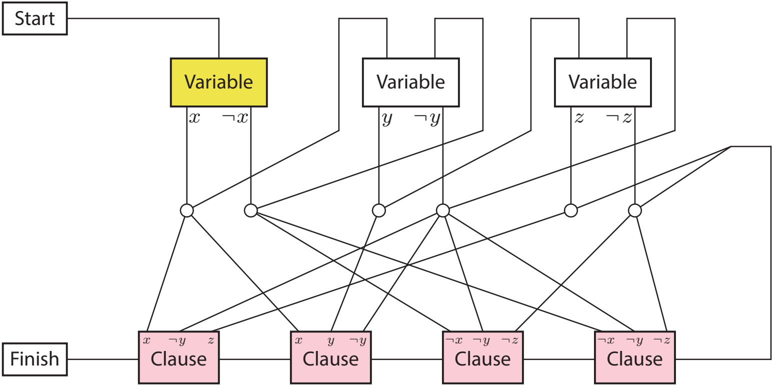


NP-hardness framework for platform games

Reduction from 3-SAT to a generic platform game

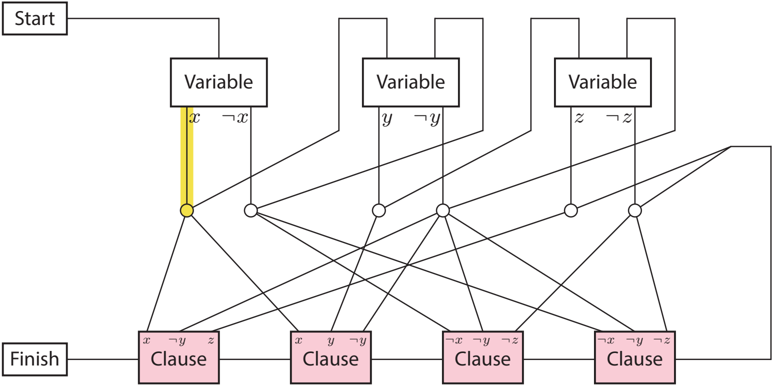


Reduction from 3-SAT to a generic platform game

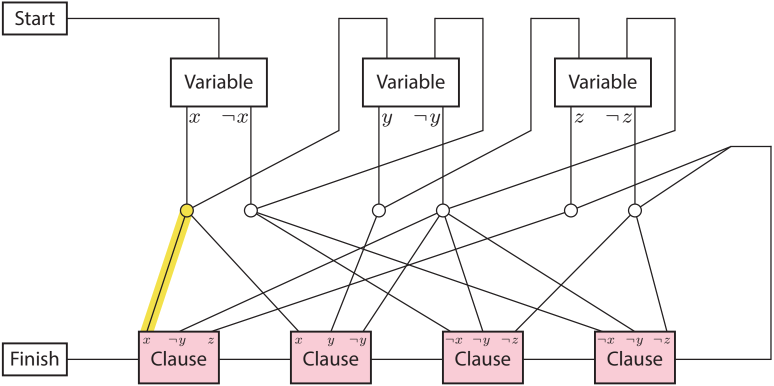


NP-hardness framework for platform games

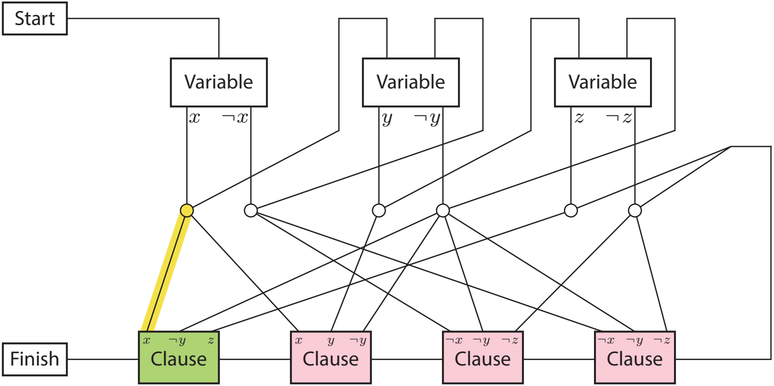
Reduction from 3-SAT to a generic platform game



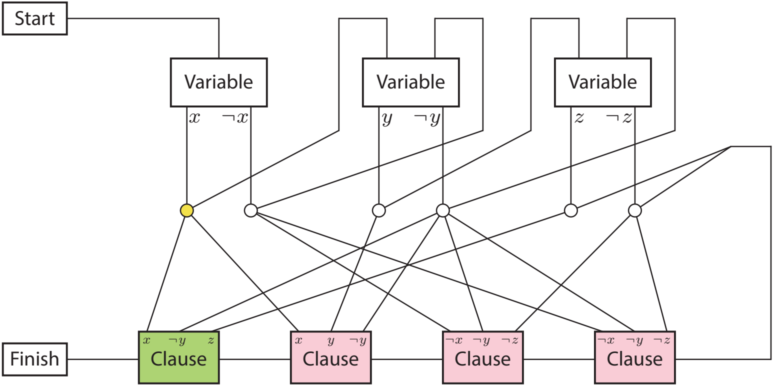
Reduction from 3-SAT to a generic platform game



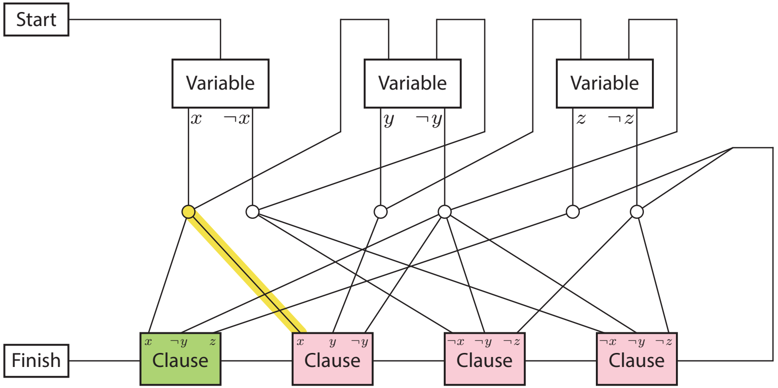
Reduction from 3-SAT to a generic platform game



Reduction from 3-SAT to a generic platform game

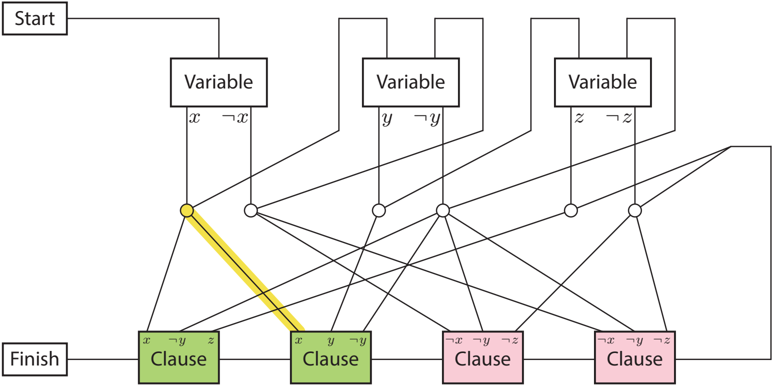


Reduction from 3-SAT to a generic platform game



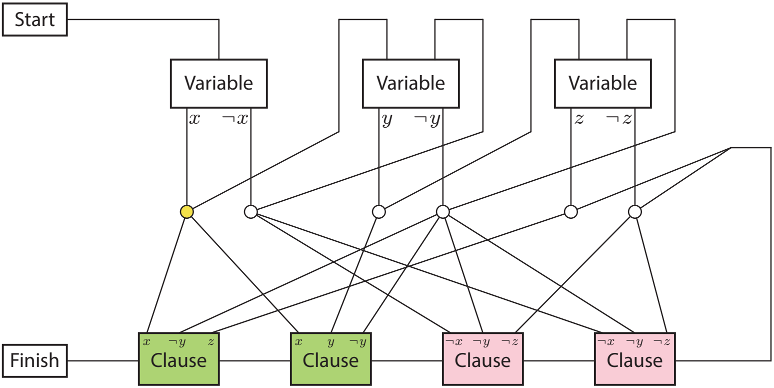
NP-hardness framework for platform games

Reduction from 3-SAT to a generic platform game



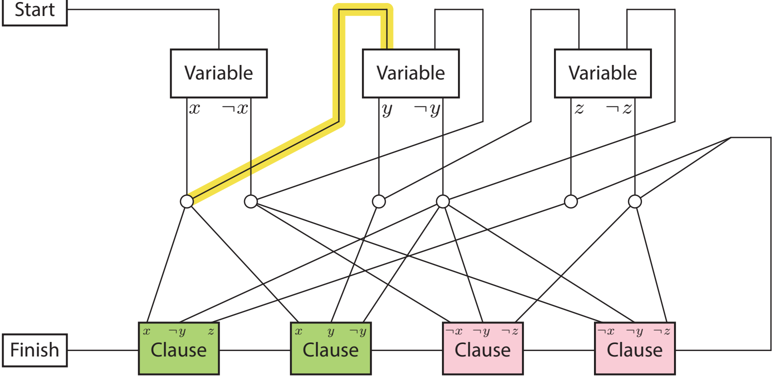
NP-hardness framework for platform games

Reduction from 3-SAT to a generic platform game



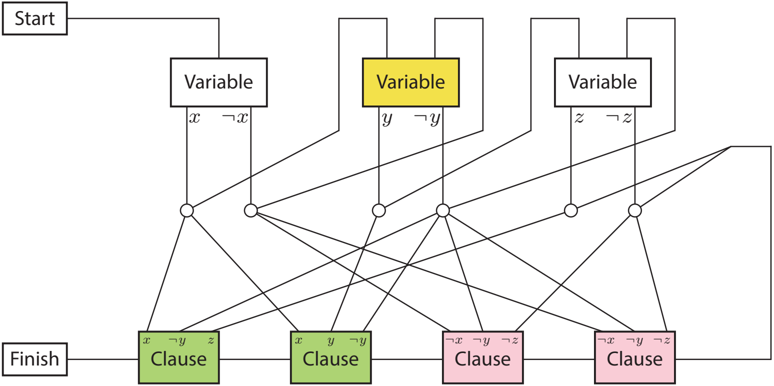
NP-hardness framework for platform games

Reduction from 3-SAT to a generic platform game



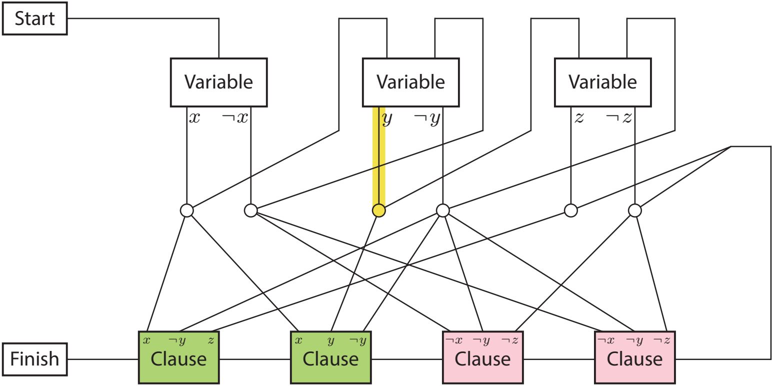
NP-hardness framework for platform games

Reduction from 3-SAT to a generic platform game



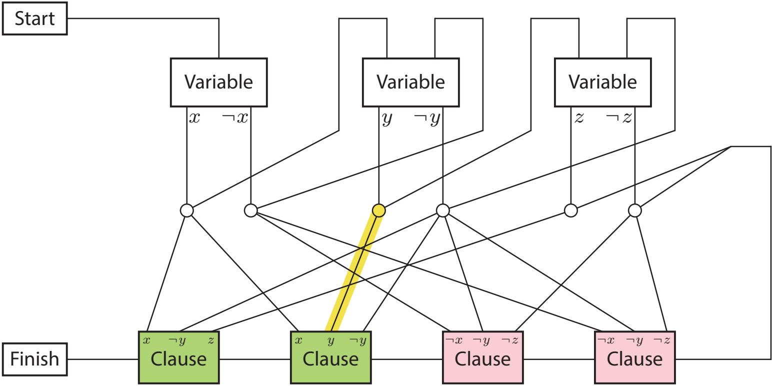
NP-hardness framework for platform games

Reduction from 3-SAT to a generic platform game



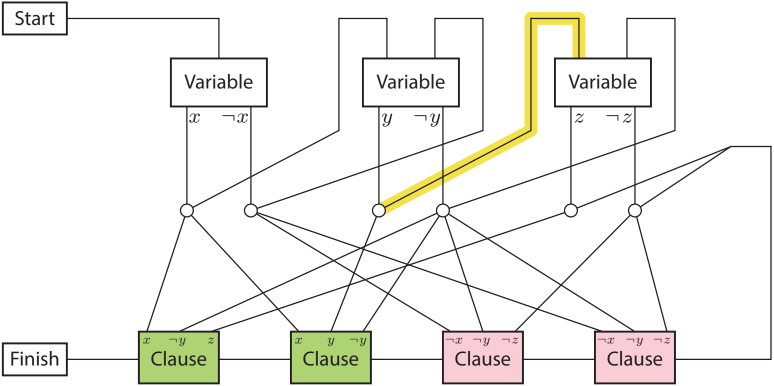
NP-hardness framework for platform games

Reduction from 3-SAT to a generic platform game



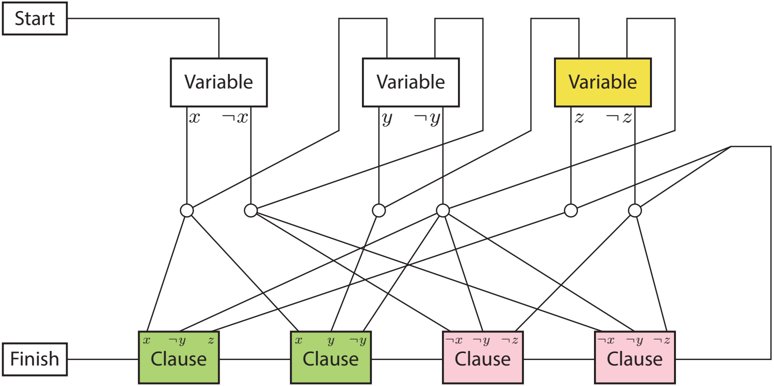
NP-hardness framework for platform games

Reduction from 3-SAT to a generic platform game



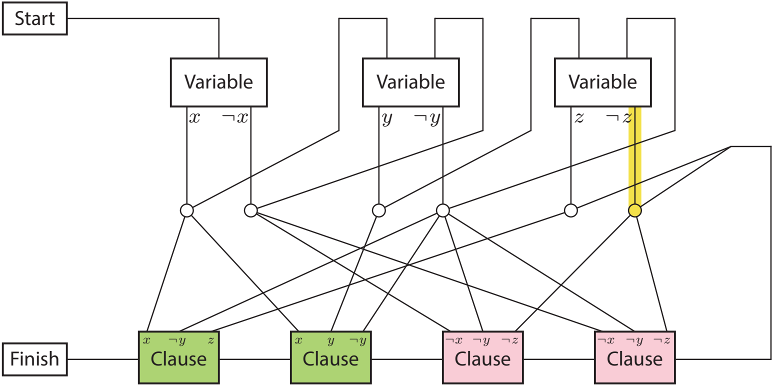
NP-hardness framework for platform games

Reduction from 3-SAT to a generic platform game

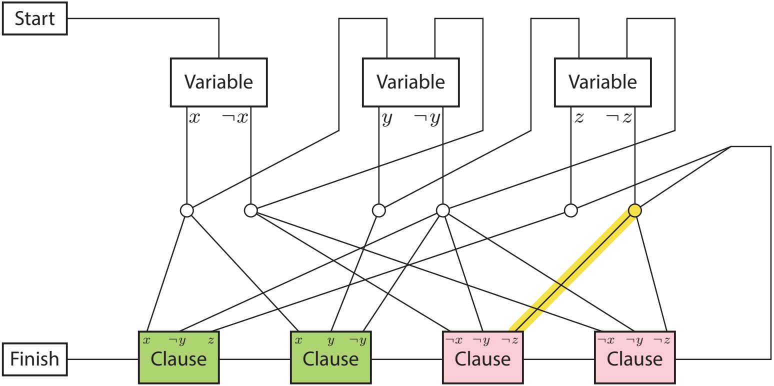


NP-hardness framework for platform games

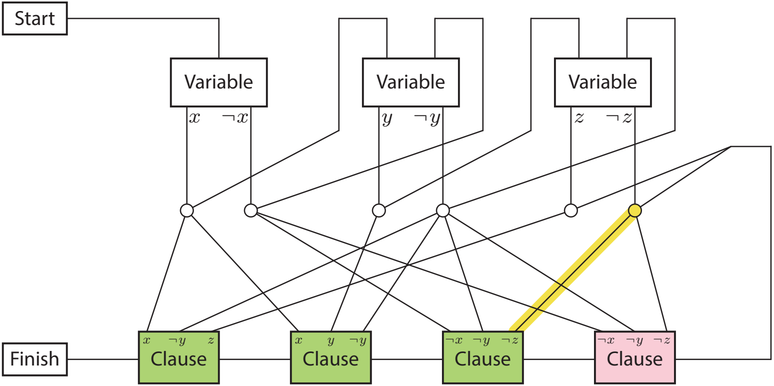
Reduction from 3-SAT to a generic platform game



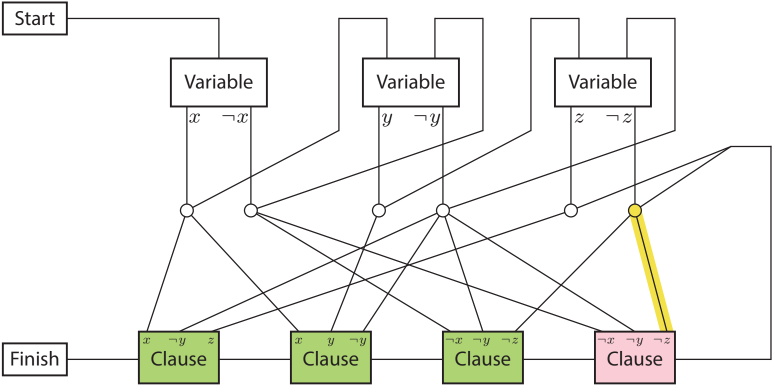
Reduction from 3-SAT to a generic platform game



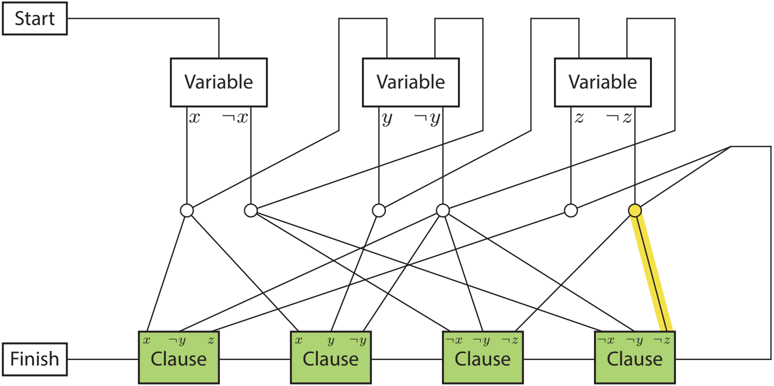
Reduction from 3-SAT to a generic platform game



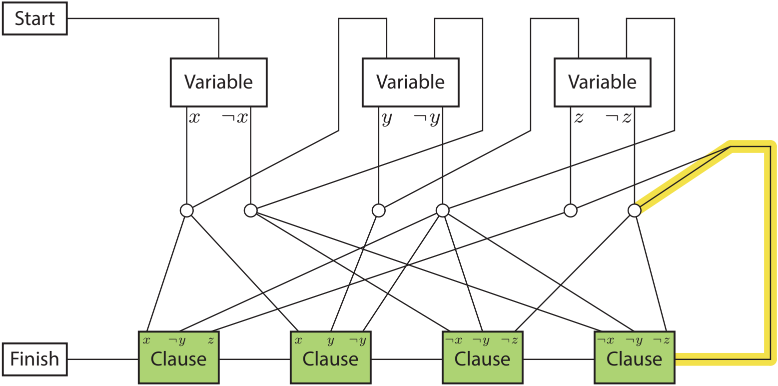
Reduction from 3-SAT to a generic platform game



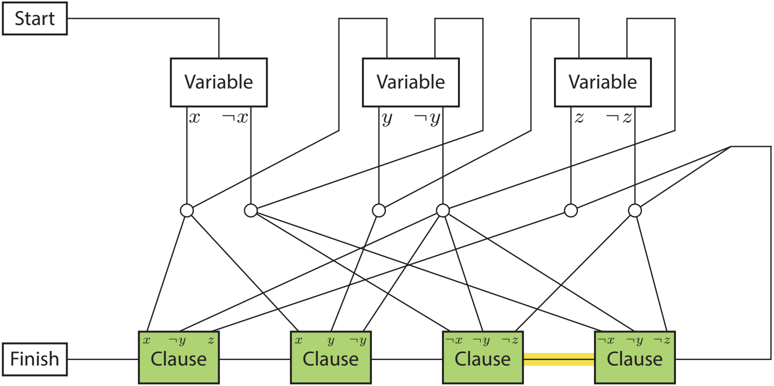
Reduction from 3-SAT to a generic platform game



Reduction from 3-SAT to a generic platform game

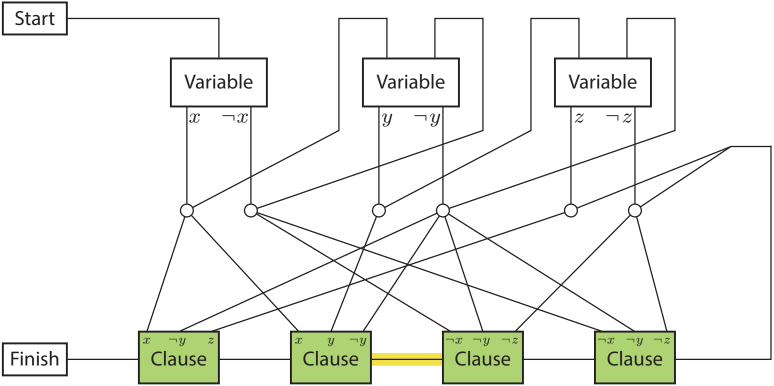


Reduction from 3-SAT to a generic platform game



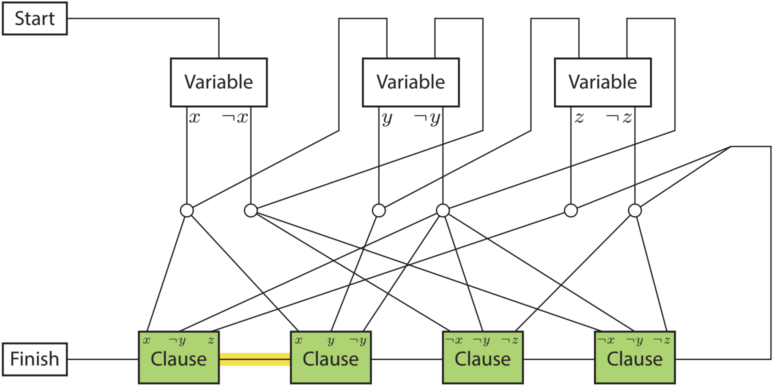
NP-hardness framework for platform games

Reduction from 3-SAT to a generic platform game

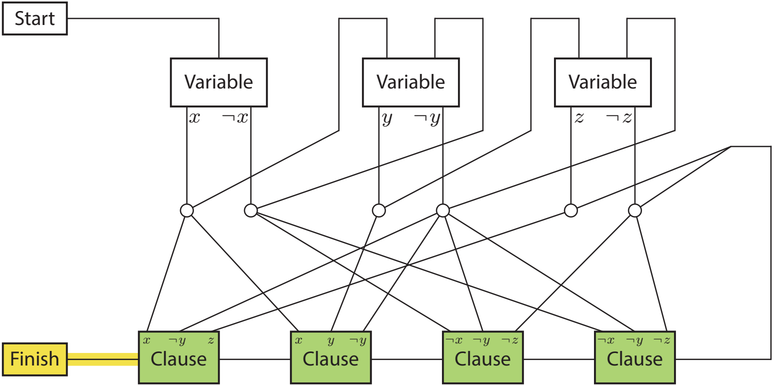


NP-hardness framework for platform games

Reduction from 3-SAT to a generic platform game



Reduction from 3-SAT to a generic platform game



NP-hardness gadgets

- **Start**
- **Finish**
- **Variable**, with mutual exclusion between outgoing paths
- **Clause**, initially locked, and unlockable from three paths
- **Crossover**, unidirectional, single-use, and fixed traversal order

NP-hardness gadgets

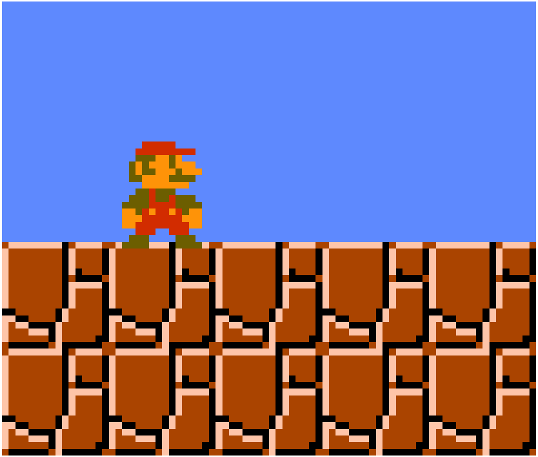
- **Start**
- **Finish**
- **Variable**, with mutual exclusion between outgoing paths
- **Clause**, initially locked, and unlockable from three paths
- **Crossover**, unidirectional, single-use, and fixed traversal order

Theorem (1)

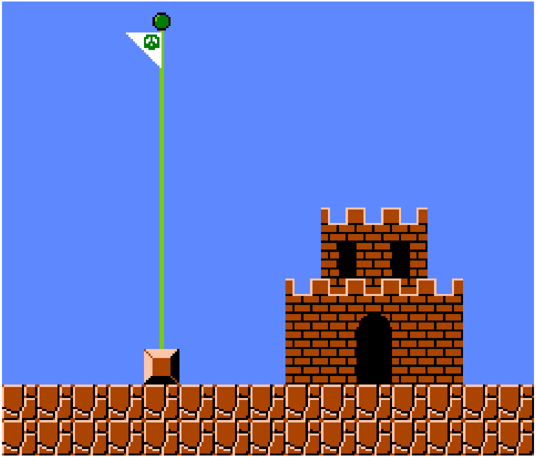
If all the above gadgets are present $\implies$ *NP-hard*

Super Mario Bros. (NES)

Super Mario Bros.: NP-hardness

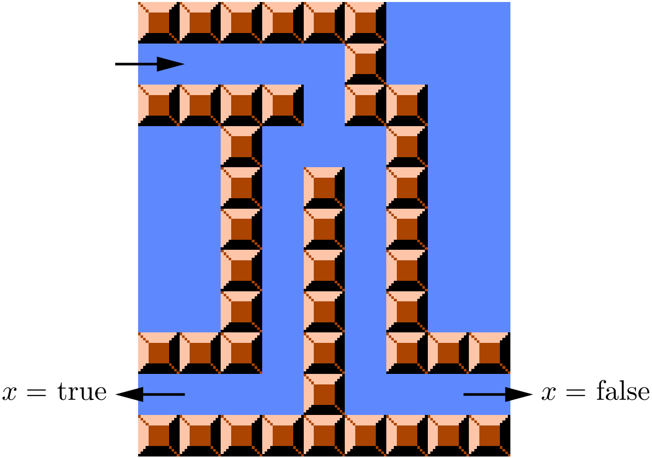


Start gadget

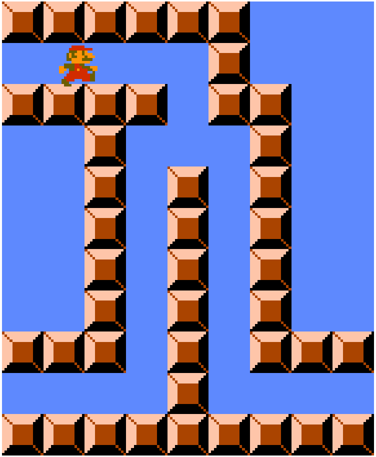


Finish gadget

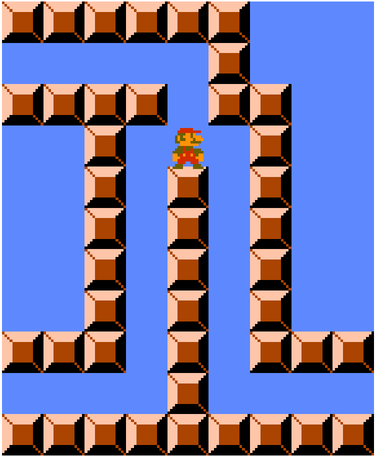
Super Mario Bros.: NP-hardness



Variable gadget

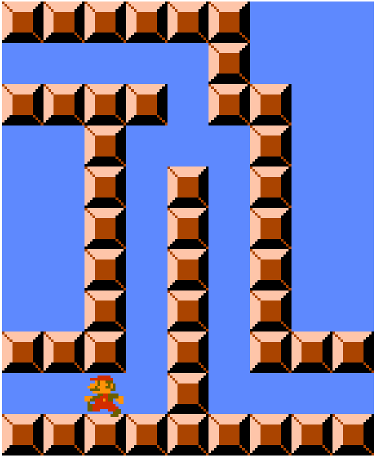


Variable gadget

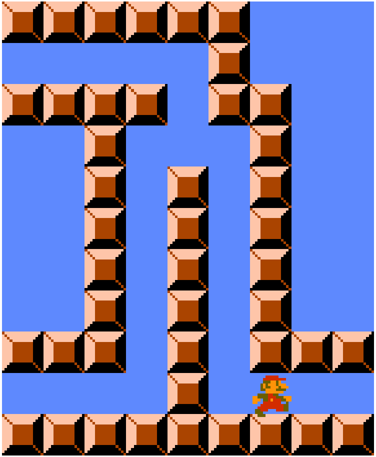


Variable gadget

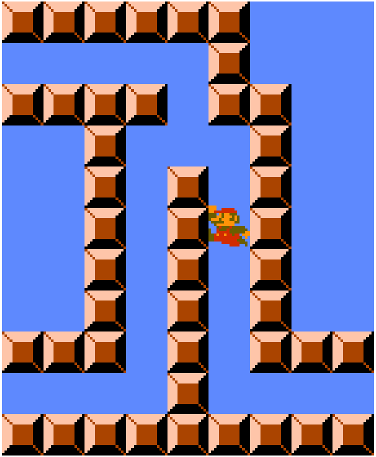
Super Mario Bros.: NP-hardness



Variable gadget

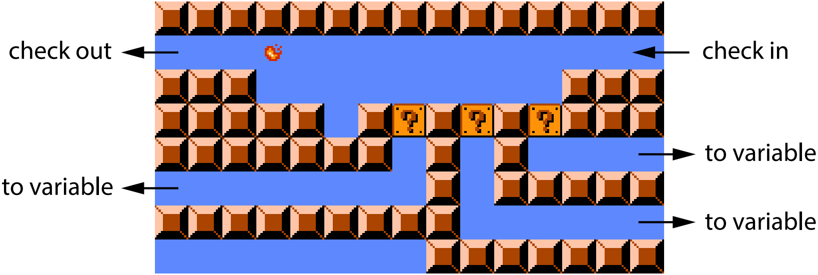


Variable gadget



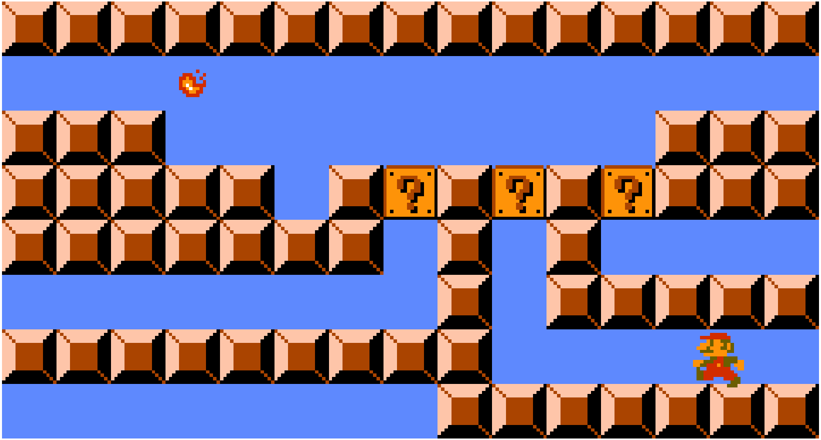
Variable gadget

Super Mario Bros.: NP-hardness

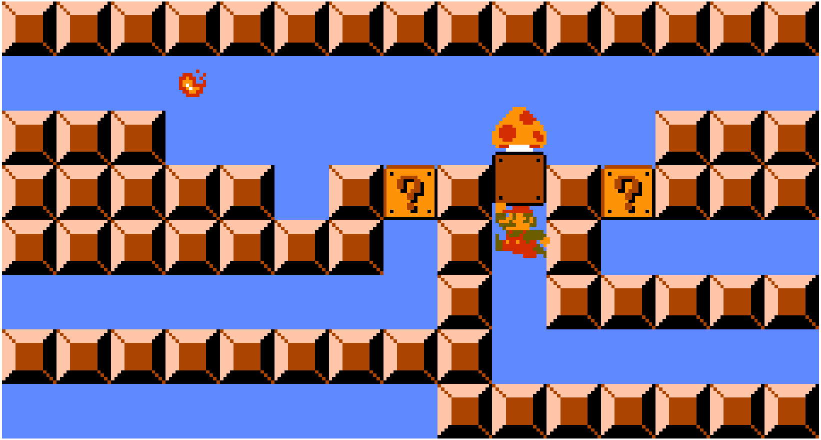


Clause gadget

Super Mario Bros.: NP-hardness

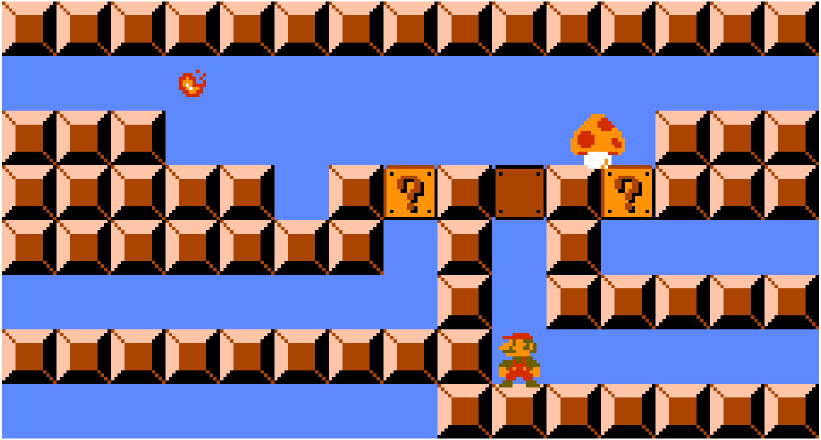


Clause gadget



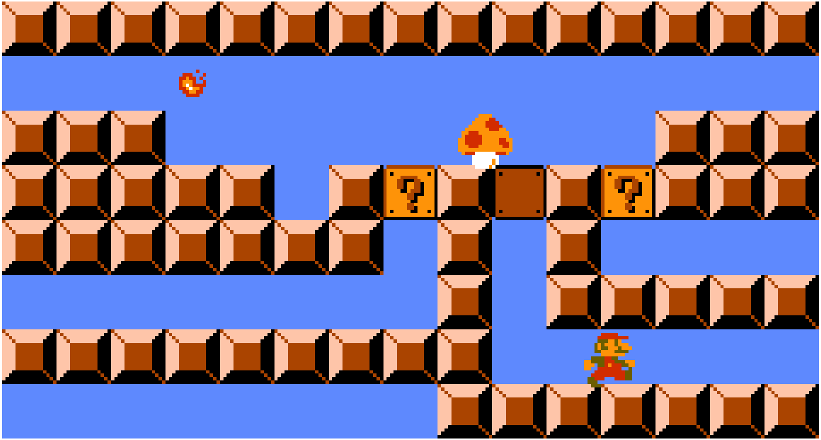
Clause gadget

Super Mario Bros.: NP-hardness



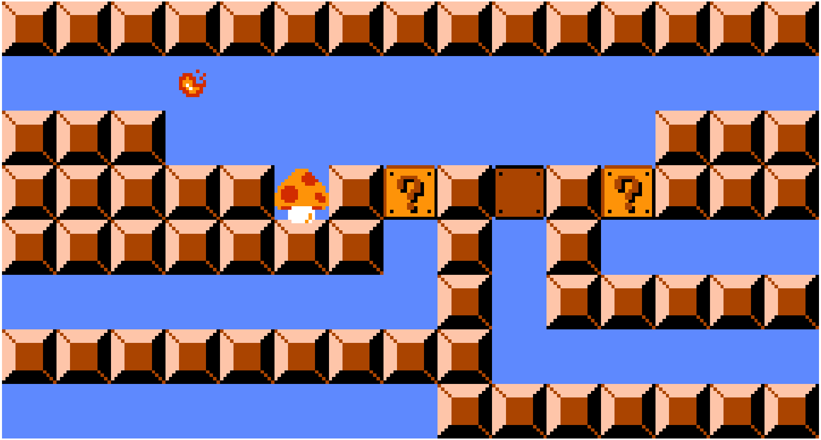
Clause gadget

Super Mario Bros.: NP-hardness



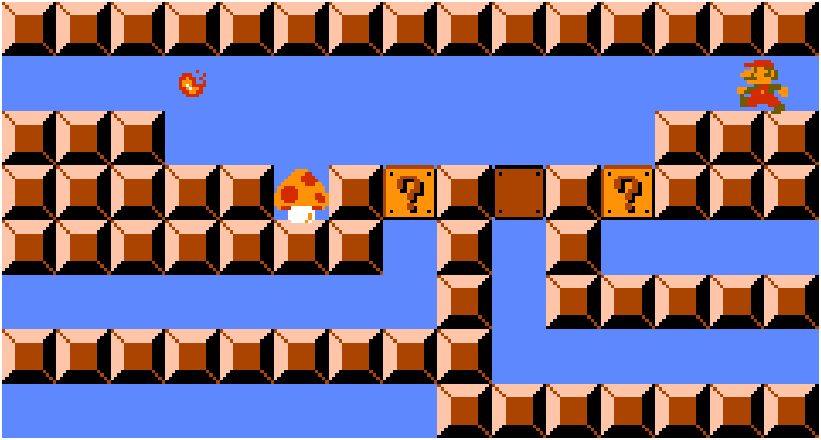
Clause gadget

Super Mario Bros.: NP-hardness

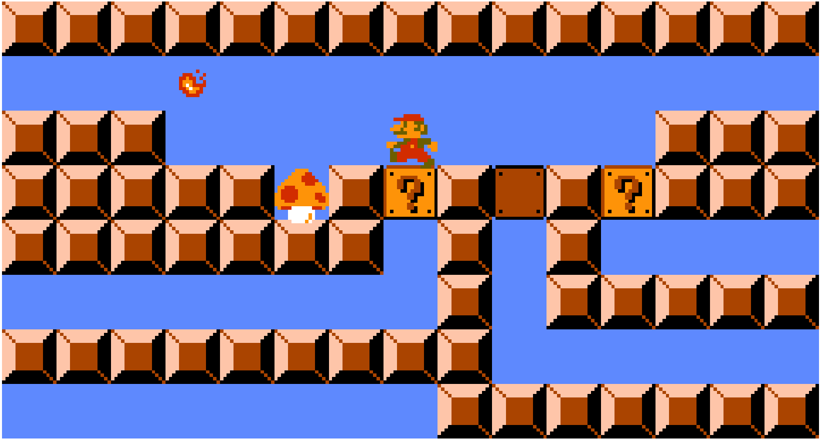


Clause gadget

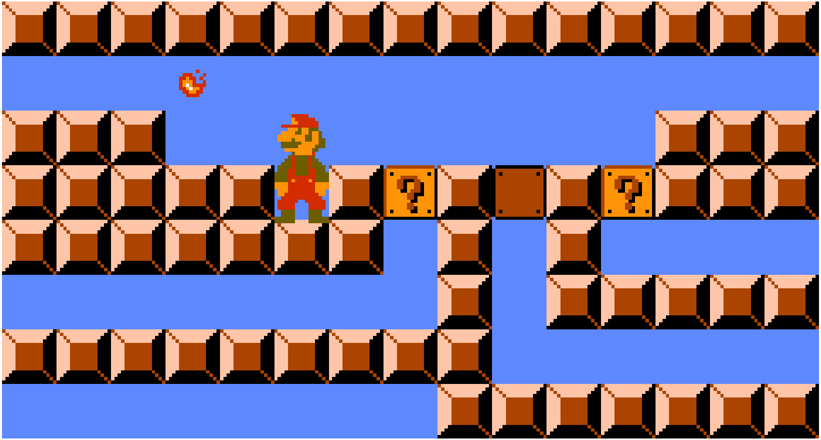
Super Mario Bros.: NP-hardness



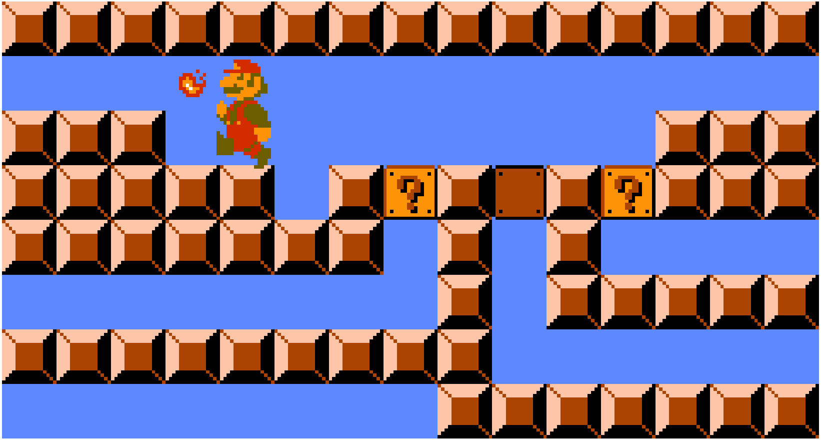
Clause gadget



Clause gadget

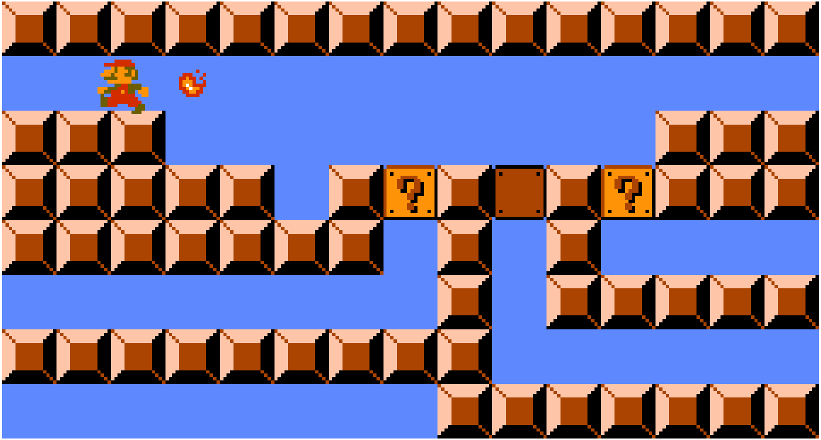


Clause gadget



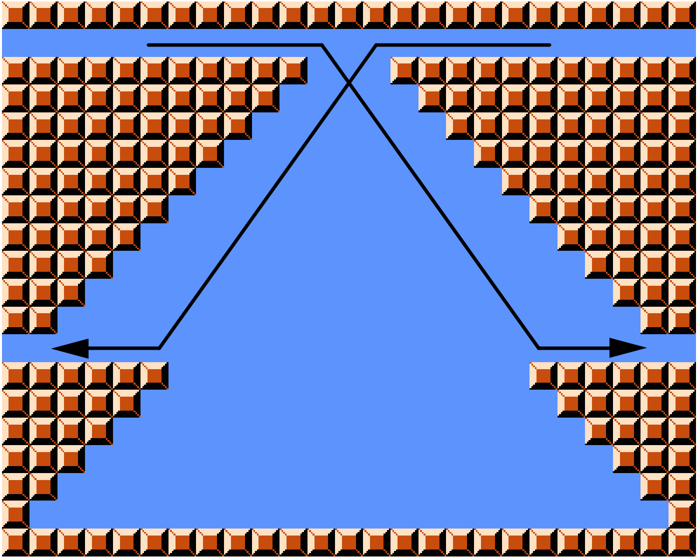
Clause gadget

Super Mario Bros.: NP-hardness

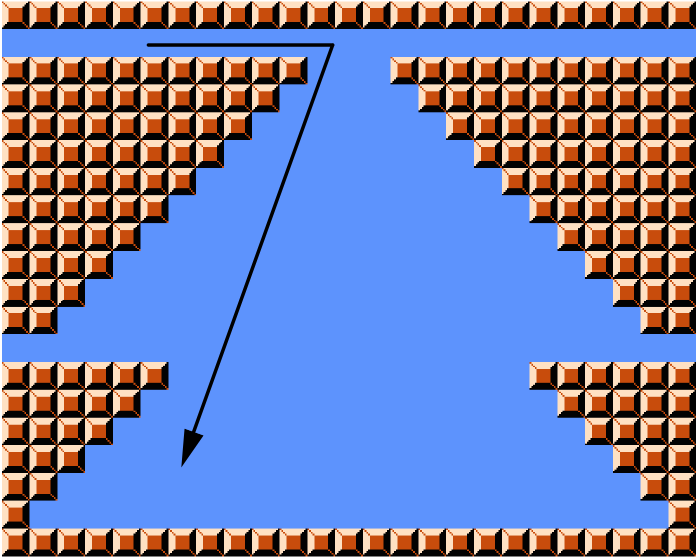


Clause gadget

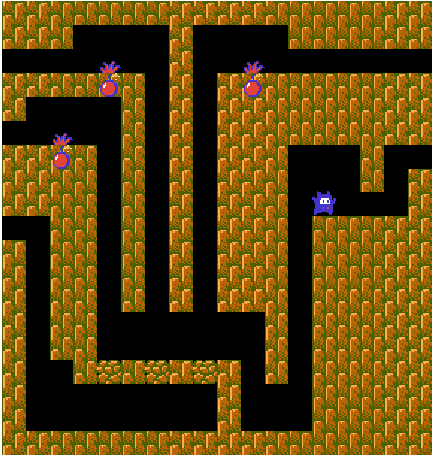
Super Mario Bros.: Crossover gadget



Super Mario Bros.: Crossover gadget



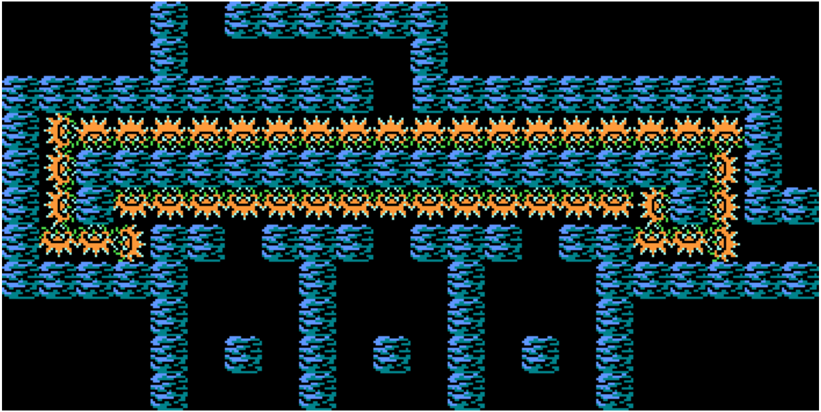
Super Mario Bros. 2 (NES)



Clause gadget

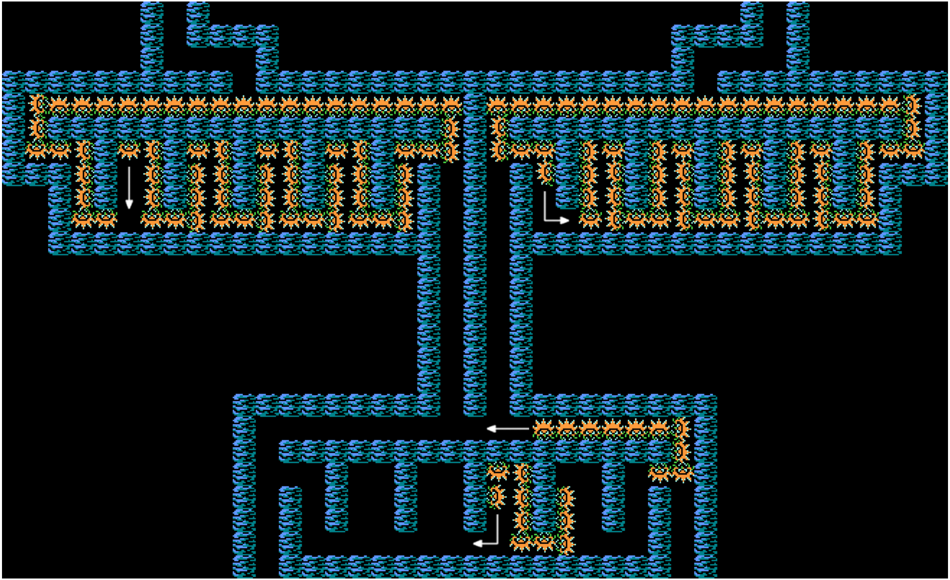
Metroid (NES)

Metroid: NP-hardness



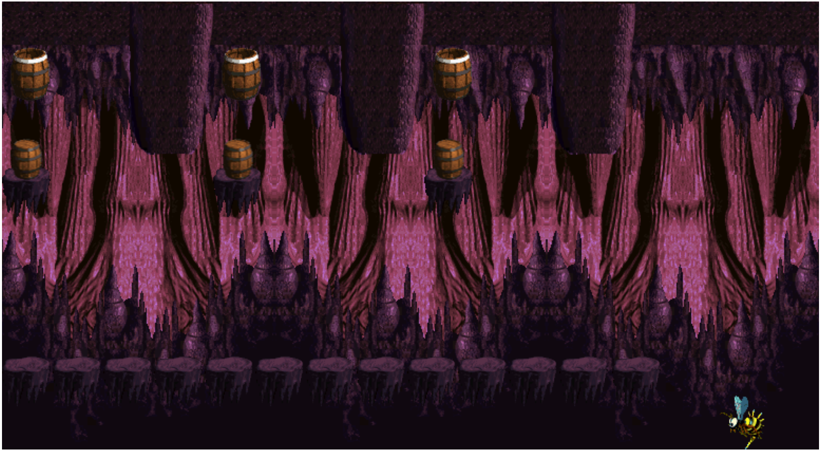
Clause gadget

Metroid: NP-hardness (Crossover gadget)



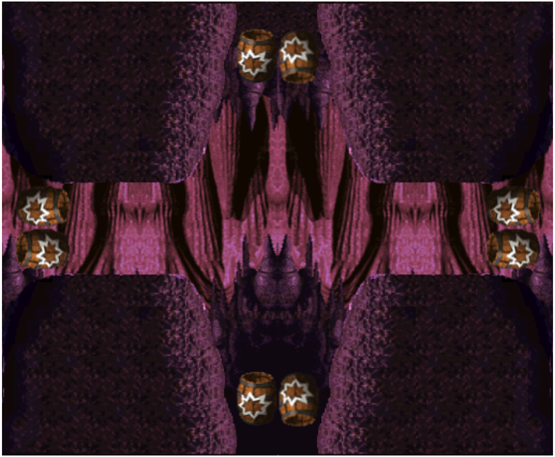
Donkey Kong Country (SNES)

Donkey Kong Country: NP-hardness



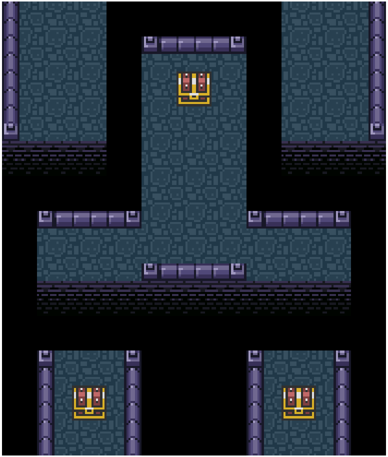
Clause gadget

Donkey Kong Country: NP-hardness



Crossover gadget

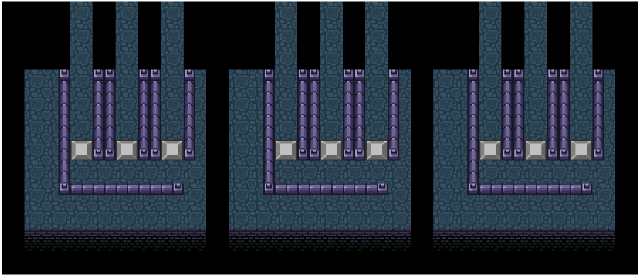
The Legend of Zelda: A Link to the Past (SNES)



Variable gadget

The Legend of Zelda: NP-hardness

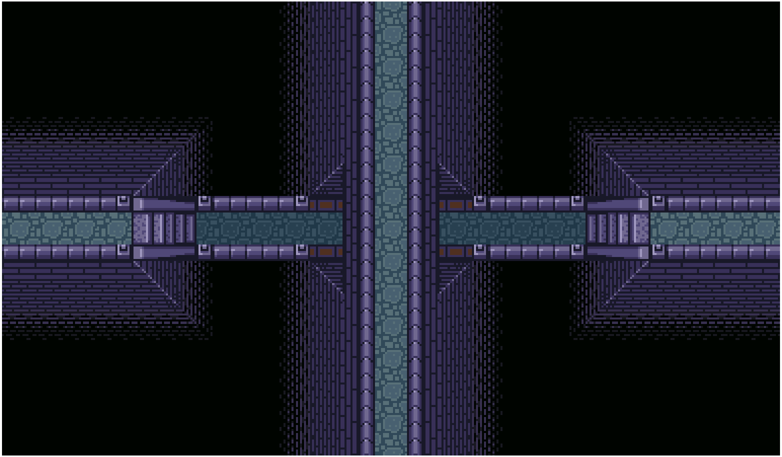
Check out



Check in

Clause gadgets

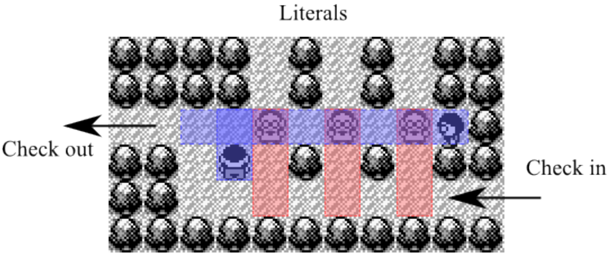
The Legend of Zelda: NP-hardness



Crossover gadget

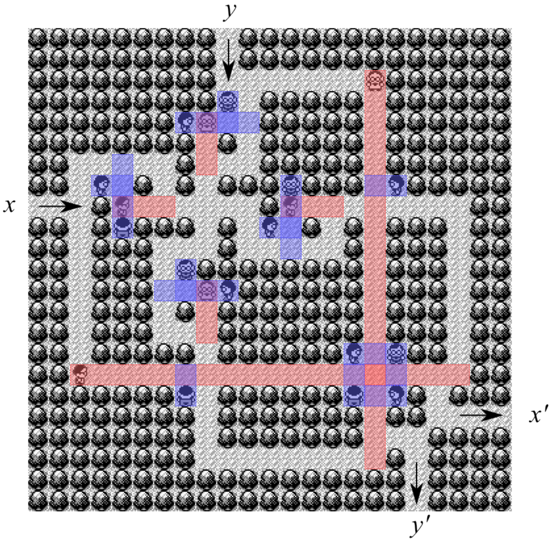
Pokémon (Game Boy)

Pokémon: NP-hardness



Clause gadget

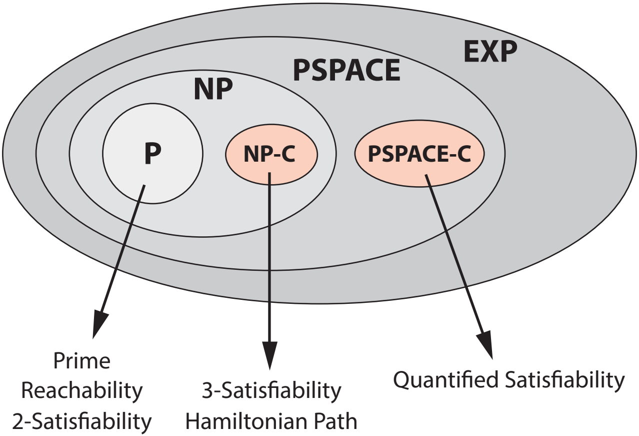
Pokémon: Crossover gadget



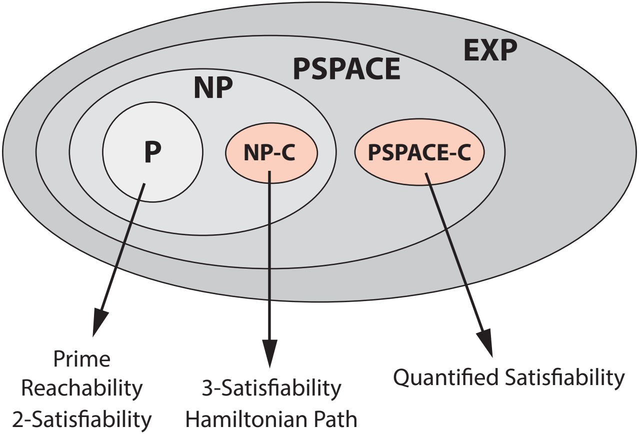
More complexity classes

PSPACE: problems decidable in polynomial space (i.e., using only polynomially many bits of "memory")

EXP: problems decidable in exponential time (i.e., not necessarily "efficiently")



More complexity classes



Observation: $\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXP}$

Theorem: $\mathbf{P} \subsetneq \mathbf{EXP}$

Hence at least one of the inclusions in the chain is strict

Quantified Satisfiability

Recall that the 3-SAT problem asks if there is a truth assignment that satisfies an expression of the form:

$$(x \wedge \bar{y} \wedge z) \vee (\bar{x} \wedge z \wedge w) \vee (y \wedge \bar{z} \wedge \bar{w}) \vee (\bar{x} \wedge y \wedge w).$$

In other words, it asks if the following formula is true:

$$\exists x, \exists y, \exists z, \exists w, (x \wedge \bar{y} \wedge z) \vee (\bar{x} \wedge z \wedge w) \vee (y \wedge \bar{z} \wedge \bar{w}) \vee (\bar{x} \wedge y \wedge w).$$

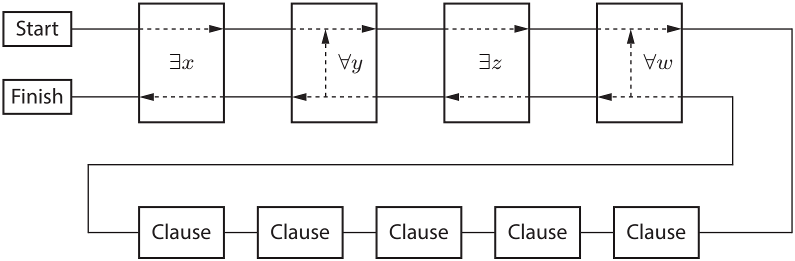
Generalizing, we can alternate $\exists$ and $\forall$ quantifiers, and ask if the following formula is true:

$$\exists x, \forall y, \exists z, \forall w, (x \wedge \bar{y} \wedge z) \vee (\bar{x} \wedge z \wedge w) \vee (y \wedge \bar{z} \wedge \bar{w}) \vee (\bar{x} \wedge y \wedge w).$$

The QSAT problem asks if a fully quantified Boolean expression is true. It is the canonical PSPACE-complete problem.

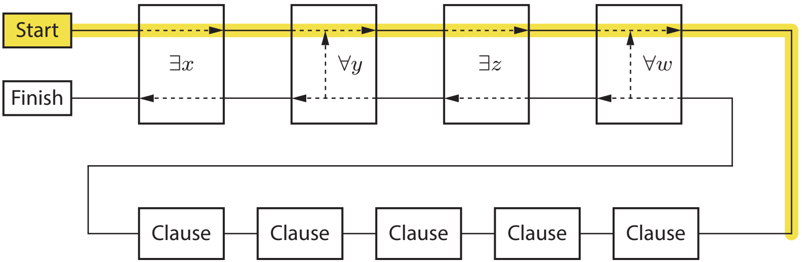
PSPACE-hardness framework

Blueprint of a reduction from QSAT



PSPACE-hardness framework

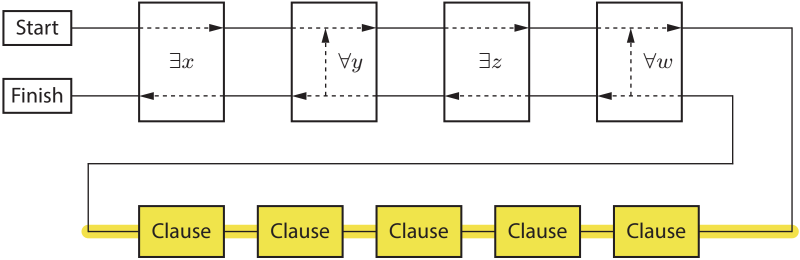
Blueprint of a reduction from QSAT



$y = w =$ true

PSPACE-hardness framework

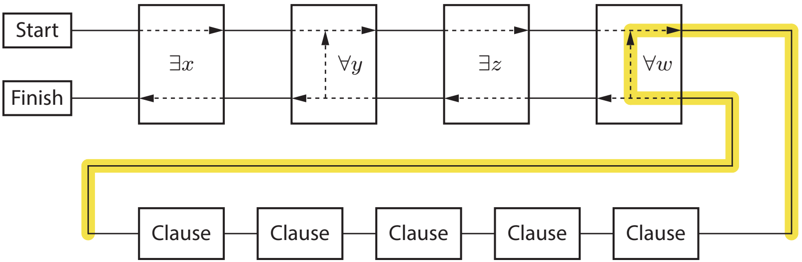
Blueprint of a reduction from QSAT



$y = w = \text{true}$

PSPACE-hardness framework

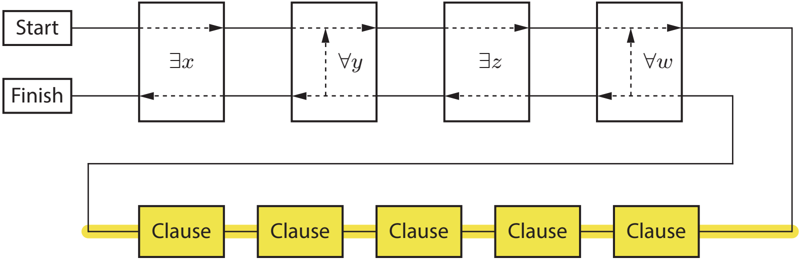
Blueprint of a reduction from QSAT



$y =$ true, $w =$ false

PSPACE-hardness framework

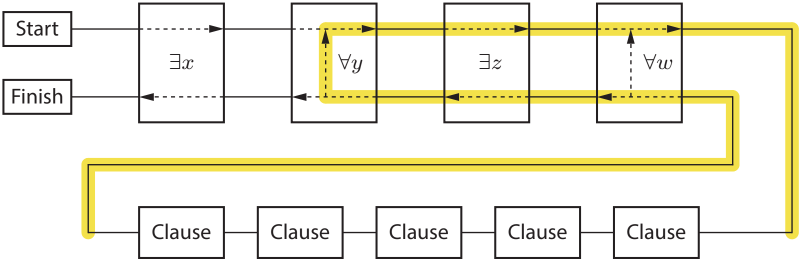
Blueprint of a reduction from QSAT



$y =$ true, $w =$ false

PSPACE-hardness framework

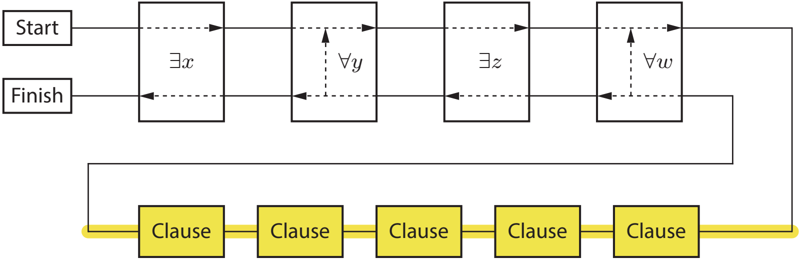
Blueprint of a reduction from QSAT



$y =$ false, $w =$ true

PSPACE-hardness framework

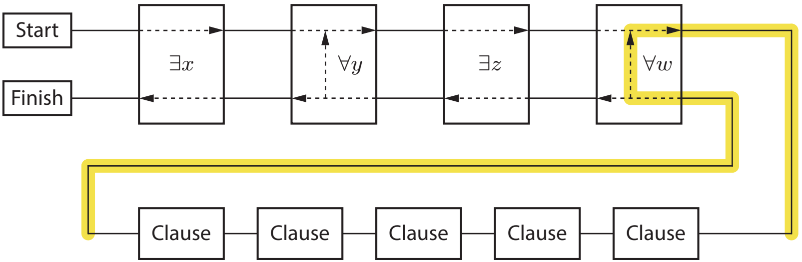
Blueprint of a reduction from QSAT



$y =$ false, $w =$ true

PSPACE-hardness framework

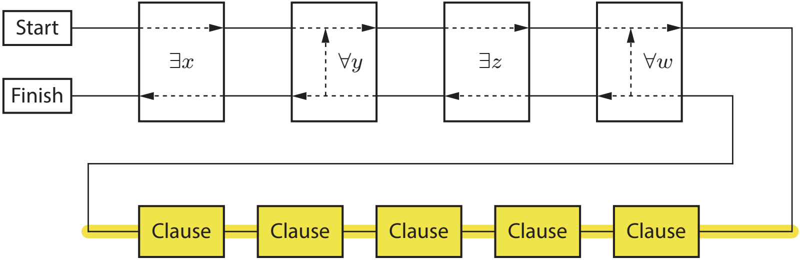
Blueprint of a reduction from QSAT



$y =$ false, $w =$ false

PSPACE-hardness framework

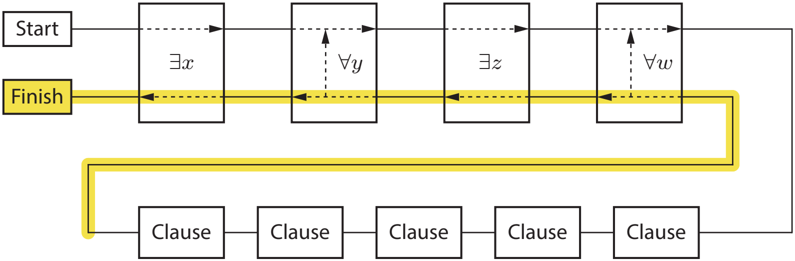
Blueprint of a reduction from QSAT



$y =$ false, $w =$ false

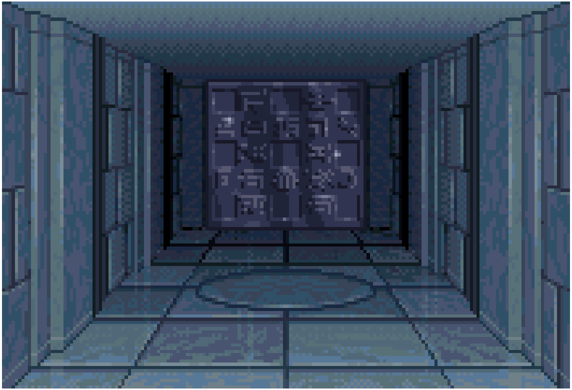
PSPACE-hardness framework

Blueprint of a reduction from QSAT



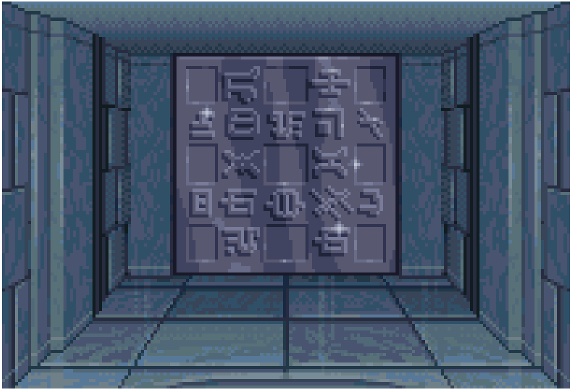
Doors and Pressure plates

When stepping on a *Pressure plate*,
a specific *Door* opens (or closes)



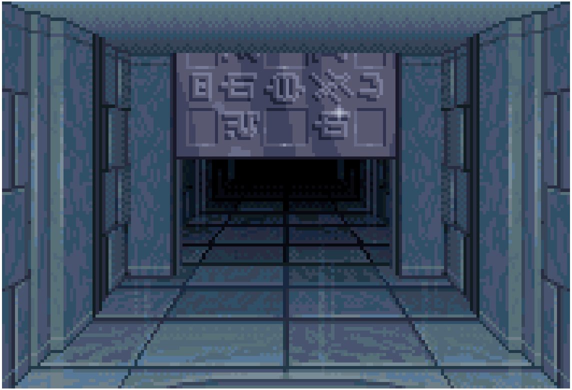
Doors and Pressure plates

When stepping on a *Pressure plate*,
a specific *Door* opens (or closes)



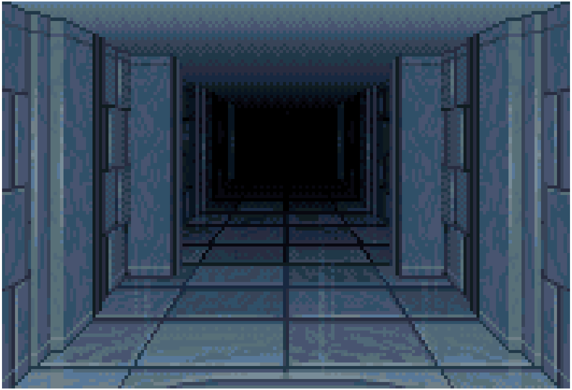
Doors and Pressure plates

When stepping on a *Pressure plate*,
a specific *Door* opens (or closes)



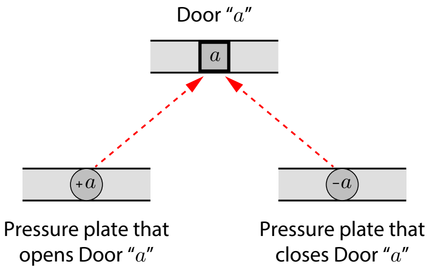
Doors and Pressure plates

When stepping on a *Pressure plate*,
a specific *Door* opens (or closes)



Doors and Pressure plates

Pressure plates can be found anywhere in the level, and they can act on Doors located arbitrarily far from them:

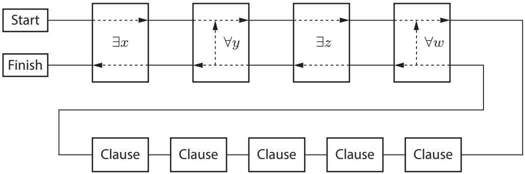


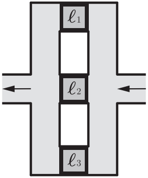
Next we will see how to implement the PSPACE-hardness framework with only Doors and Pressure Plates:

Theorem (2)

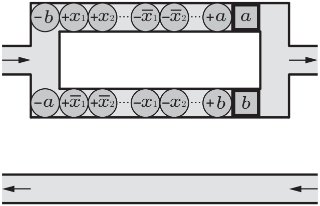
$$\textit{Doors} + \textit{Pressure plates} \implies \textit{PSPACE-hard}$$

PSPACE-hardness framework: implementation

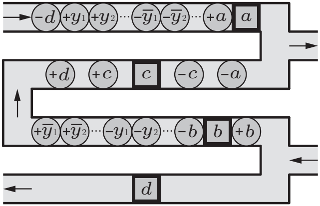




Clause



Existential gadget

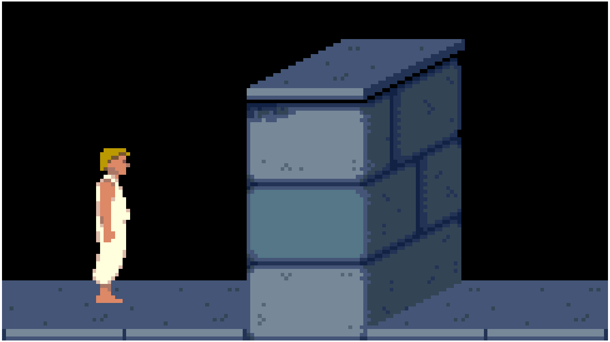


Universal gadget

Prince of Persia (PC, Amiga)

Prince of Persia (PC, Amiga)

Prince of Persia (PC, Amiga)



Pressure plate that must be pressed

Quake (PC)

Quake (PC)

Sonic (Sega Genesis)

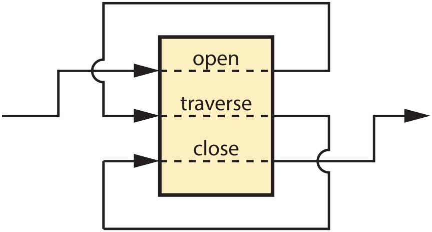
Sonic (Sega Genesis)

Adventure games by LucasArts and Sierra

From Pressure plates to Stand-alone Doors

Observation: in our implementation of the PSPACE-hardness framework, we only used two Pressure plates per Door: one that opens it, and one that closes it. We can incorporate these three elements in a single *Stand-alone Door* gadget:

$\equiv$



The "open" path may even give the player the option to leave the door closed: indeed, choosing not to open a door is never helpful!

Theorem (3)

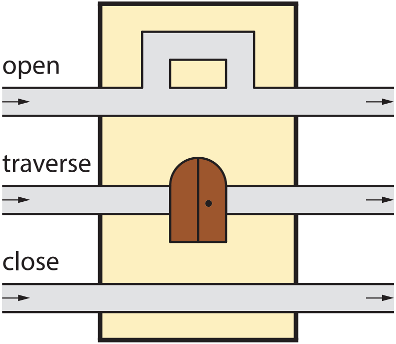
Stand-alone Doors + Crossovers $\implies$ *PSPACE-hard*

PSPACE-hardness gadgets

- **Start** & **Finish**
- **Crossover**, usable multiple times
- **Stand-alone Door**: can be opened, closed, and traversed if and only if it is open

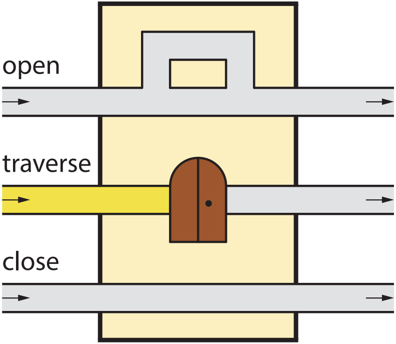
PSPACE-hardness gadgets

- **Start** & **Finish**
- **Crossover**, usable multiple times
- **Stand-alone Door**: can be opened, closed, and traversed if and only if it is open



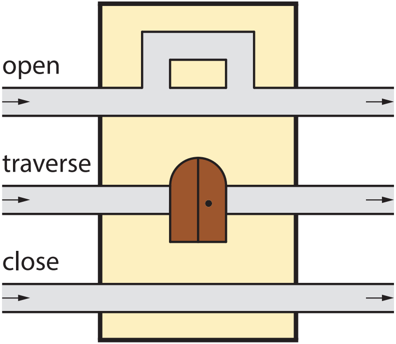
PSPACE-hardness gadgets

- **Start** & **Finish**
- **Crossover**, usable multiple times
- **Stand-alone Door**: can be opened, closed, and traversed if and only if it is open



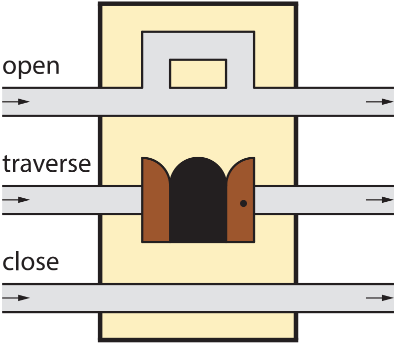
PSPACE-hardness gadgets

- **Start** & **Finish**
- **Crossover**, usable multiple times
- **Stand-alone Door**: can be opened, closed, and traversed if and only if it is open



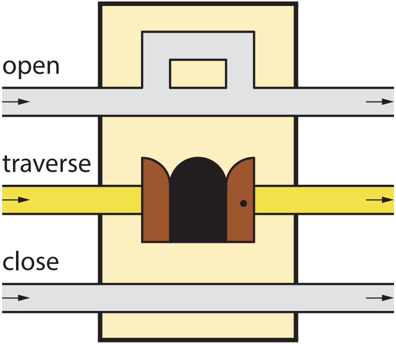
PSPACE-hardness gadgets

- **Start** & **Finish**
- **Crossover**, usable multiple times
- **Stand-alone Door**: can be opened, closed, and traversed if and only if it is open



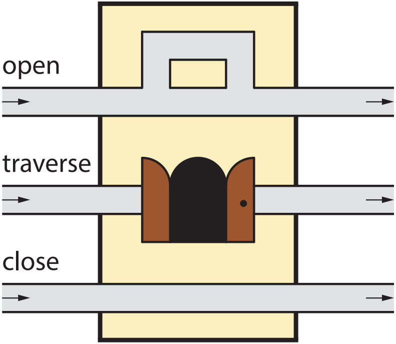
PSPACE-hardness gadgets

- **Start** & **Finish**
- **Crossover**, usable multiple times
- **Stand-alone Door**: can be opened, closed, and traversed if and only if it is open



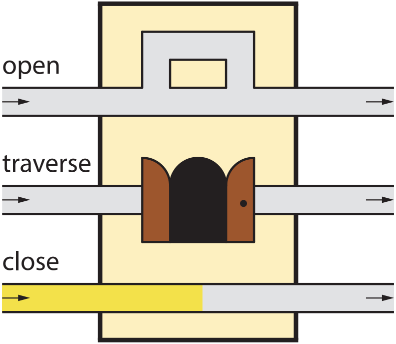
PSPACE-hardness gadgets

- **Start** & **Finish**
- **Crossover**, usable multiple times
- **Stand-alone Door**: can be opened, closed, and traversed if and only if it is open



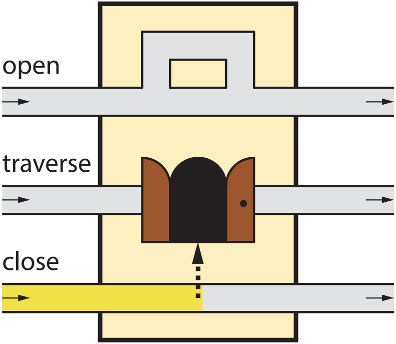
PSPACE-hardness gadgets

- **Start** & **Finish**
- **Crossover**, usable multiple times
- **Stand-alone Door**: can be opened, closed, and traversed if and only if it is open



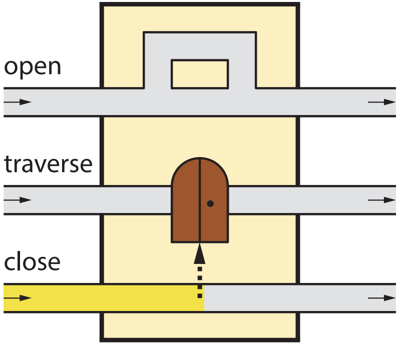
PSPACE-hardness gadgets

- **Start** & **Finish**
- **Crossover**, usable multiple times
- **Stand-alone Door**: can be opened, closed, and traversed if and only if it is open



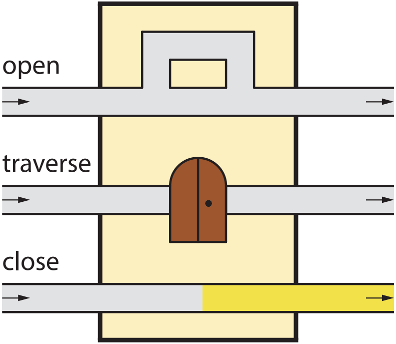
PSPACE-hardness gadgets

- **Start** & **Finish**
- **Crossover**, usable multiple times
- **Stand-alone Door**: can be opened, closed, and traversed if and only if it is open



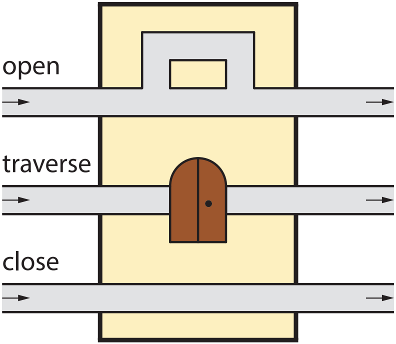
PSPACE-hardness gadgets

- **Start** & **Finish**
- **Crossover**, usable multiple times
- **Stand-alone Door**: can be opened, closed, and traversed if and only if it is open



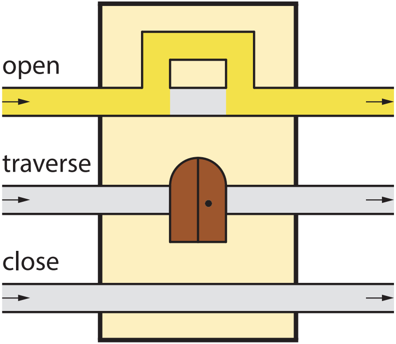
PSPACE-hardness gadgets

- **Start** & **Finish**
- **Crossover**, usable multiple times
- **Stand-alone Door**: can be opened, closed, and traversed if and only if it is open



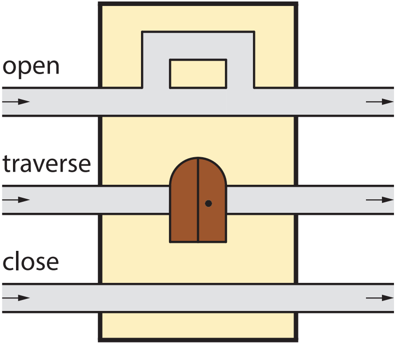
PSPACE-hardness gadgets

- **Start** & **Finish**
- **Crossover**, usable multiple times
- **Stand-alone Door**: can be opened, closed, and traversed if and only if it is open



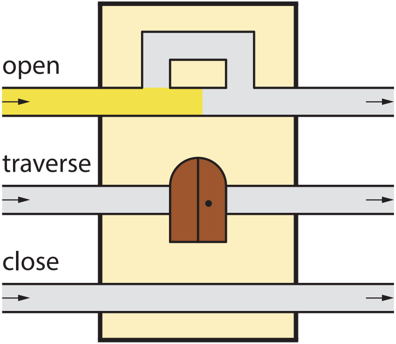
PSPACE-hardness gadgets

- **Start** & **Finish**
- **Crossover**, usable multiple times
- **Stand-alone Door**: can be opened, closed, and traversed if and only if it is open



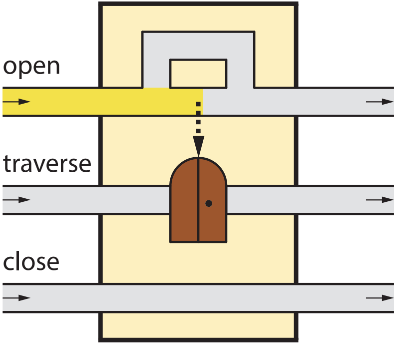
PSPACE-hardness gadgets

- **Start** & **Finish**
- **Crossover**, usable multiple times
- **Stand-alone Door**: can be opened, closed, and traversed if and only if it is open



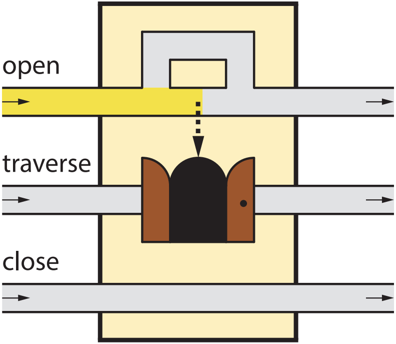
PSPACE-hardness gadgets

- **Start** & **Finish**
- **Crossover**, usable multiple times
- **Stand-alone Door**: can be opened, closed, and traversed if and only if it is open



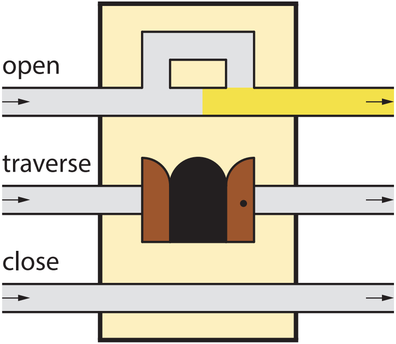
PSPACE-hardness gadgets

- **Start** & **Finish**
- **Crossover**, usable multiple times
- **Stand-alone Door**: can be opened, closed, and traversed if and only if it is open

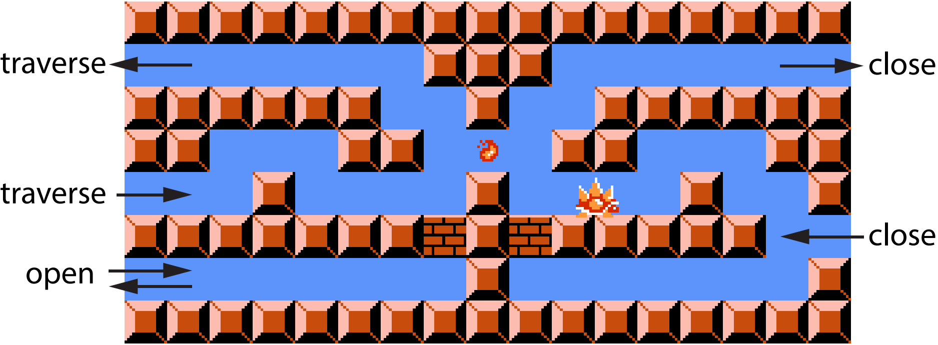


PSPACE-hardness gadgets

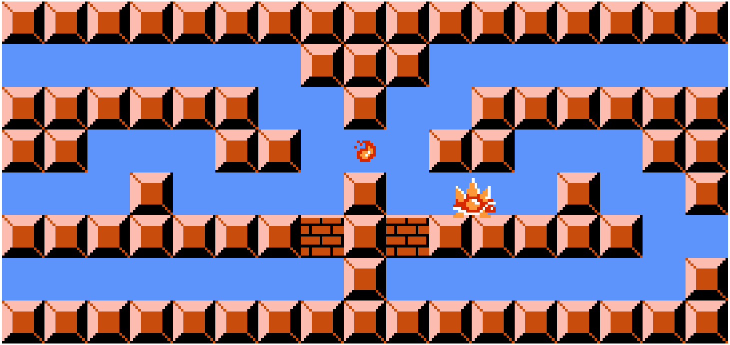
- **Start** & **Finish**
- **Crossover**, usable multiple times
- **Stand-alone Door**: can be opened, closed, and traversed if and only if it is open



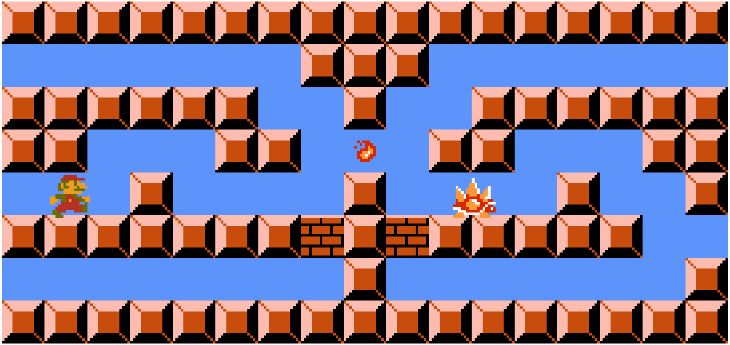
Super Mario Bros.: Door gadget



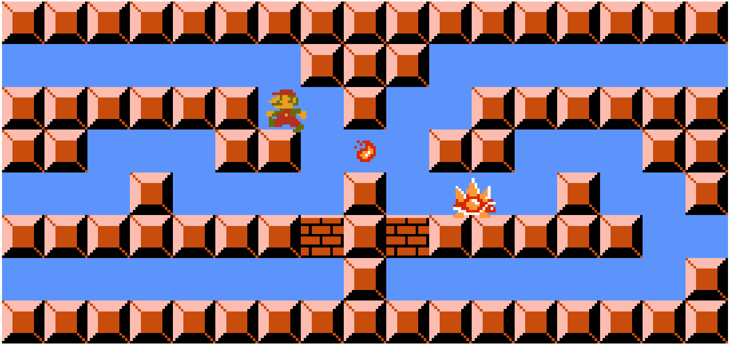
Super Mario Bros.: Door gadget



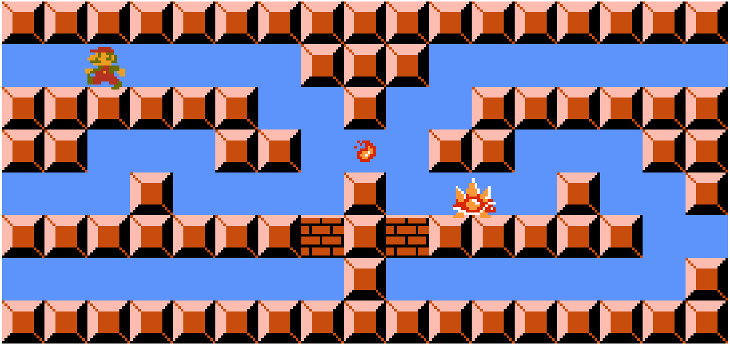
Super Mario Bros.: Door gadget



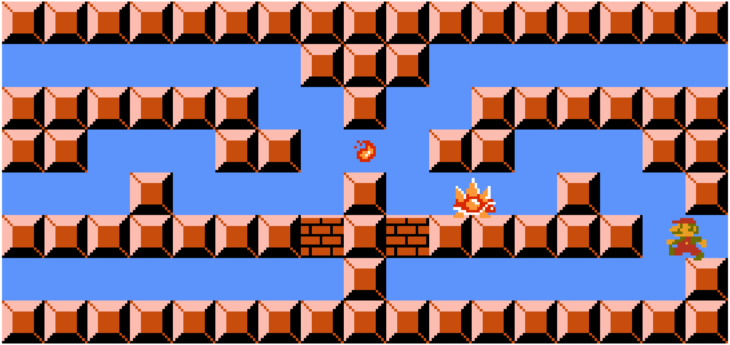
Super Mario Bros.: Door gadget



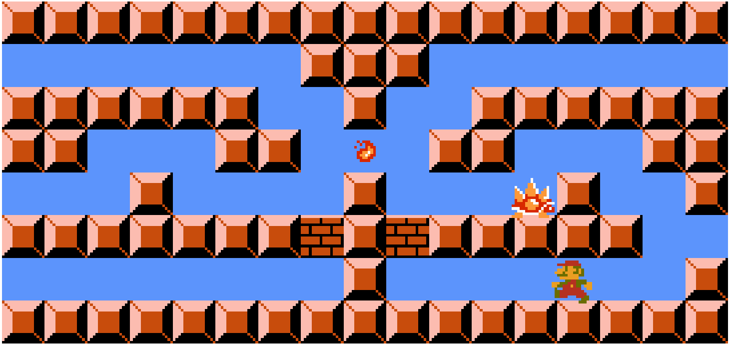
Super Mario Bros.: Door gadget



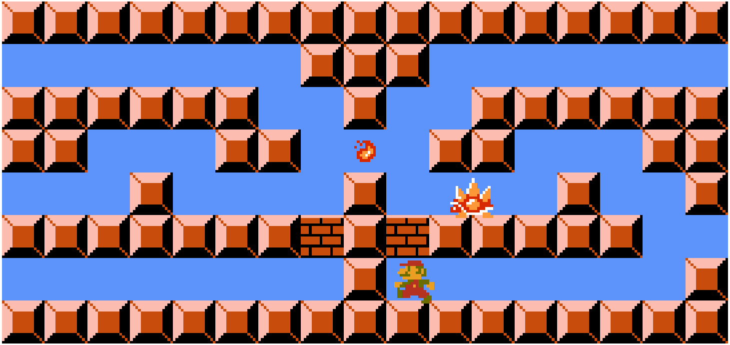
Super Mario Bros.: Door gadget



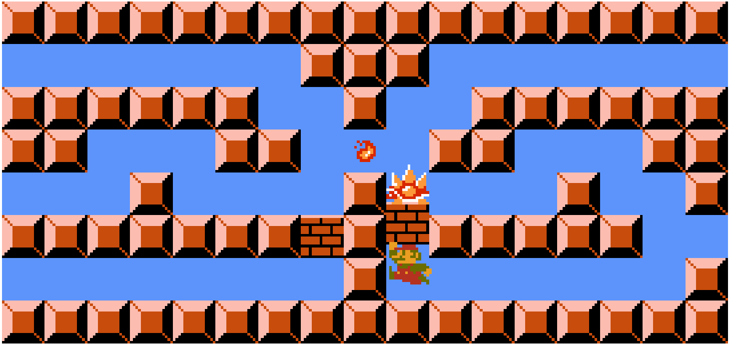
Super Mario Bros.: Door gadget



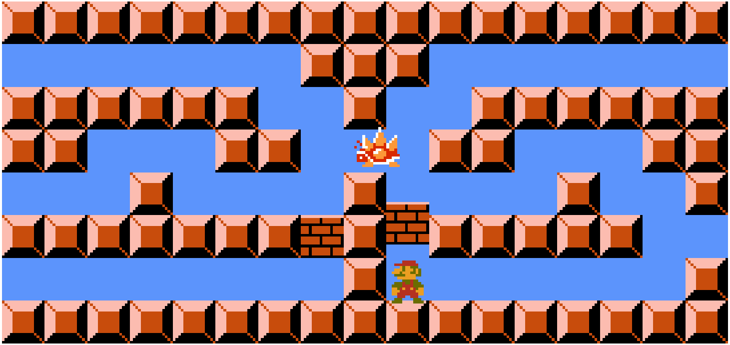
Super Mario Bros.: Door gadget



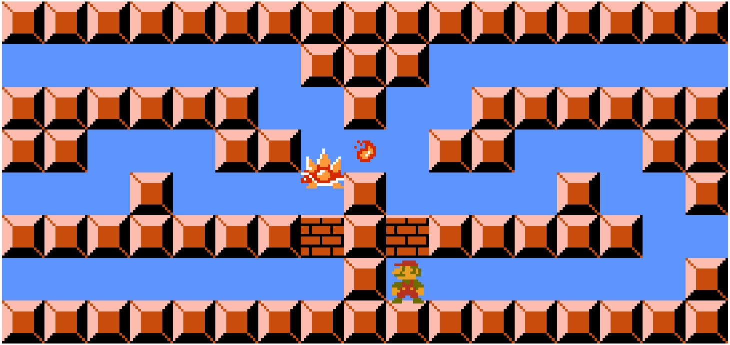
Super Mario Bros.: Door gadget



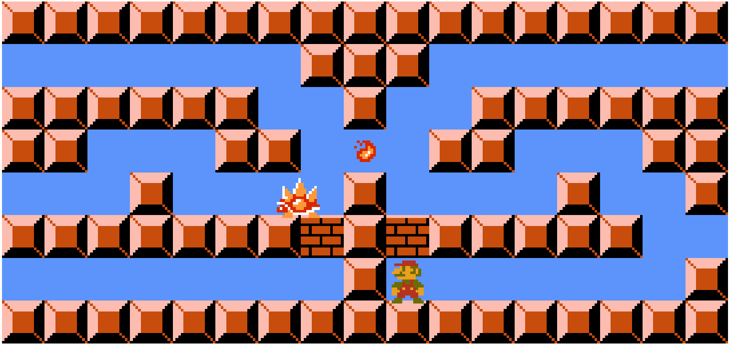
Super Mario Bros.: Door gadget



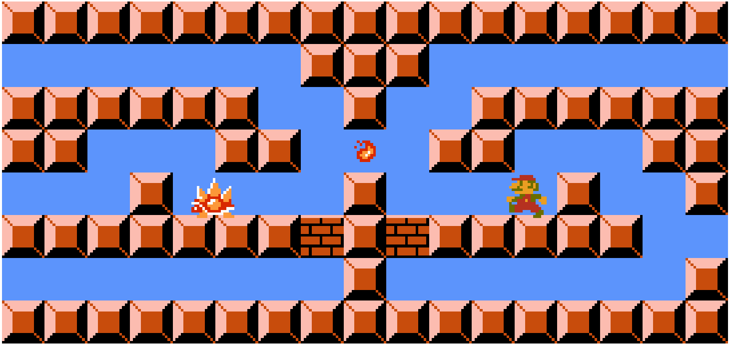
Super Mario Bros.: Door gadget



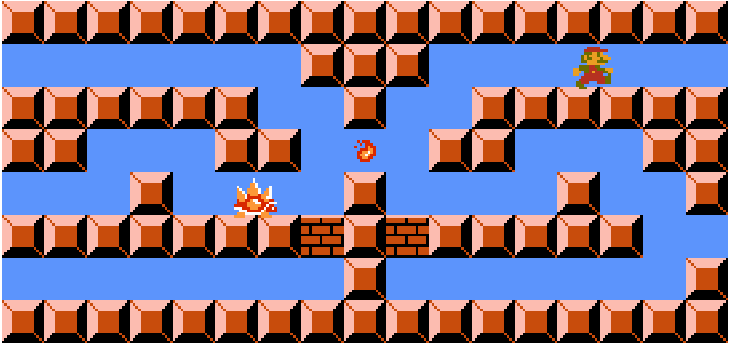
Super Mario Bros.: Door gadget



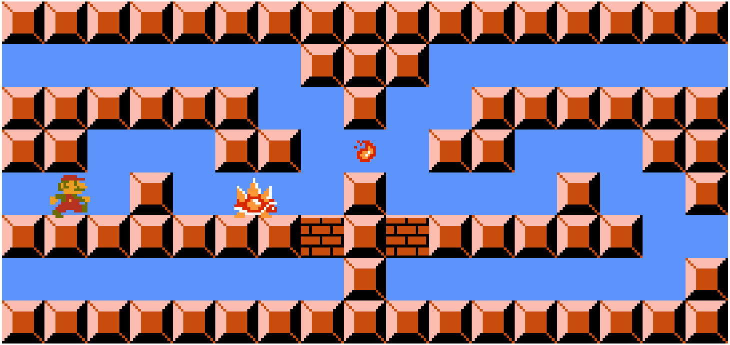
Super Mario Bros.: Door gadget



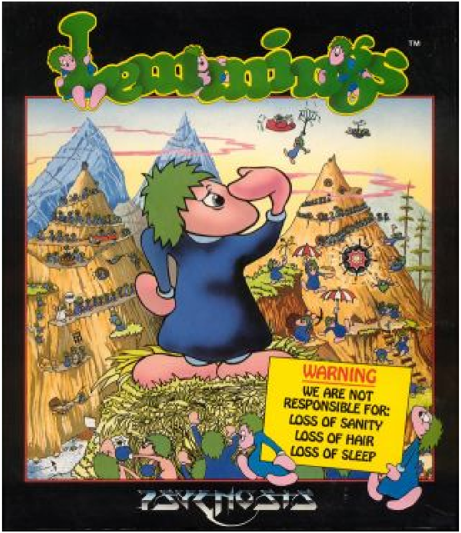
Super Mario Bros.: Door gadget



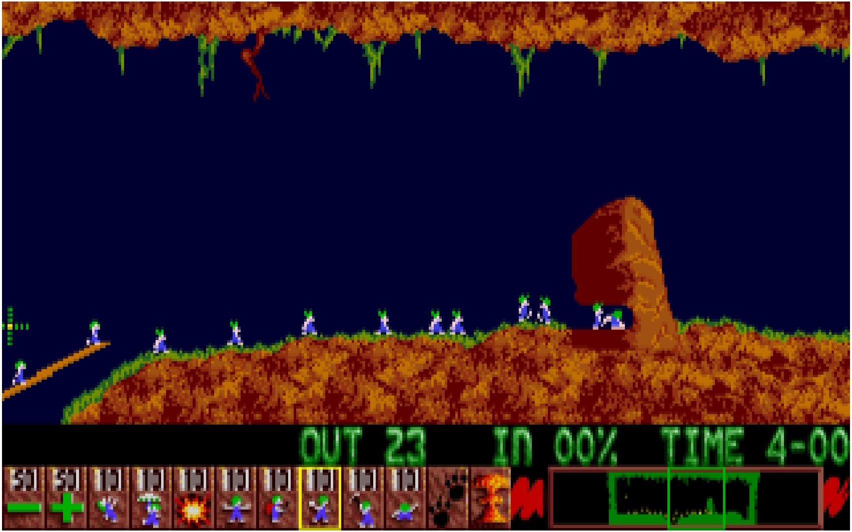
Super Mario Bros.: Door gadget



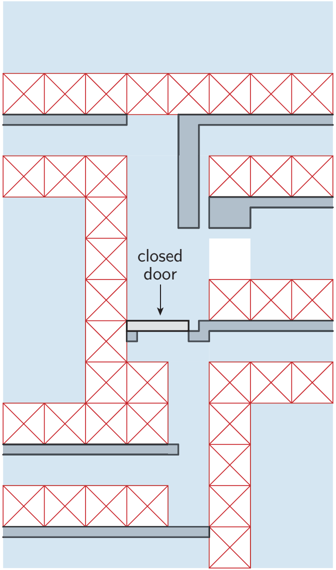
Lemmings (PC, Amiga)



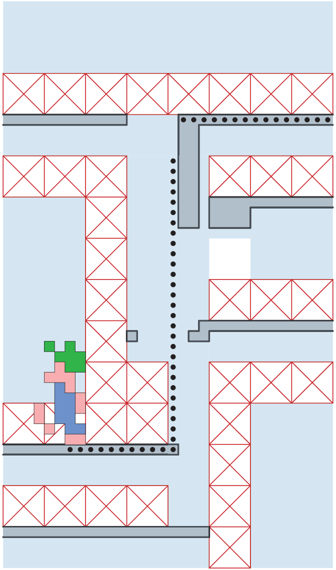
Lemmings (PC, Amiga)



Lemmings: Door gadget

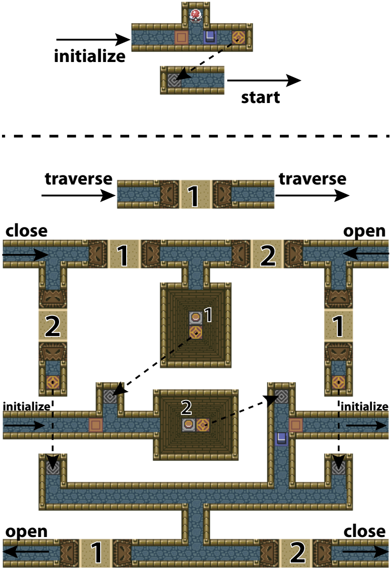


Lemmings: Door gadget



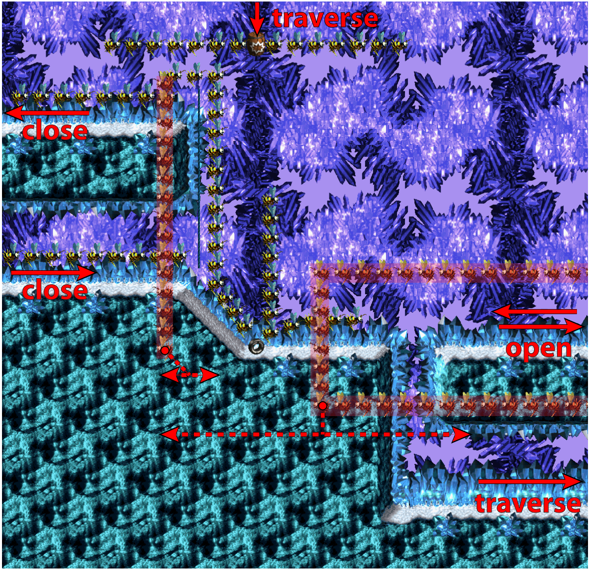
The Legend of Zelda: Door gadget

The Legend of Zelda: Door gadget

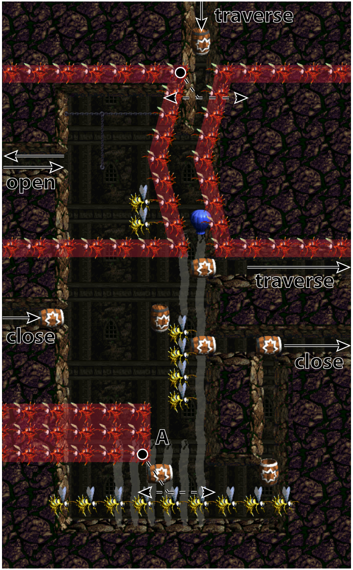


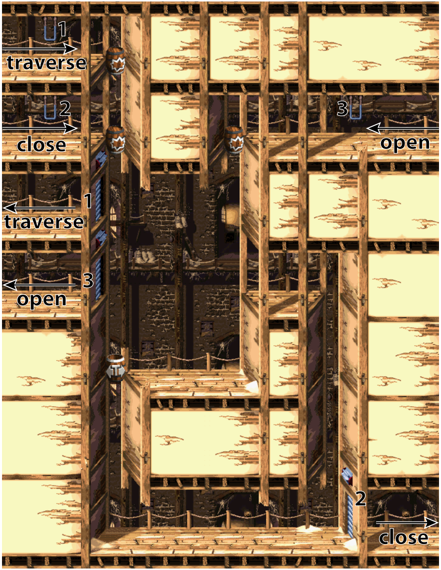
Donkey Kong Country (SNES)

Donkey Kong Country: Door gadget



Donkey Kong Country 2 & 3: Door gadgets





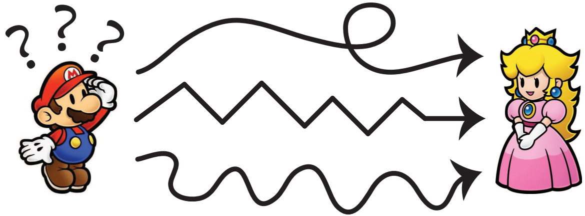
Why do we care?

We know for a fact that every level of any “real” game is solvable.
So, why do we study level solvability?

Why do we care?

We know for a fact that every level of any "real" game is solvable. So, why do we study level solvability?

- **Player's perspective:** "Which way should I go?"



One of the paths is certainly right, but which one?

- **Designer's perspective:** "Is my new level solvable at all?"

Conclusions

Single-player games:

- Games in P: they have a "simple" winning algorithm. When this algorithm is discovered, they become uninteresting.
  $\longrightarrow$ Games with short longevity. :(
- NP-complete games: they keep challenging the player, and they have "short" solutions whose discovery requires creativity.
  $\longrightarrow$ Fun games! :)
- PSPACE-complete games: they still require ingenuity, but solving them may take exponentially many "moves".
  $\longrightarrow$ Challenging games, but may be tedious. :/

What's next?

Two-player games:

- Games in NP: they have "simple" winning strategies.

  $\longrightarrow$ Games with short longevity. :(

- PSPACE-complete games: they are challenging, and the winner is determined after polynomially many moves.

  $\longrightarrow$ Fun games! :)

  Examples: Reversi, Gomoku, Go (without ko)

- EXP-complete games: they still require ingenuity, but games may last exponentially many "moves".

  $\longrightarrow$ Challenging games, but may be tedious. :/

  Examples: Checkers, Chess, Go (with Japanese ko)


GAME OVER

References

G. Viglietta
Gaming is a hard job, but someone has to do it!
Theory of Computing Systems 54, 2014

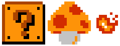
G. Viglietta
Lemmings is PSPACE-complete
Theoretical Computer Science 586, 2015

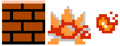
G. Aloupis, E. D. Demaine, A. Guo, and G. Viglietta
Nintendo games are (computationally) hard
Theoretical Computer Science 586, 2015

E. D. Demaine, G. Viglietta, and A. Williams
Super Mario Bros. is harder/easier than we thought
FUN 2016

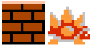
Assignment 3

In this lesson we saw that Super Mario Bros. levels that contain only Question-Mark Blocks, Super Mushrooms, and Fire Bars are NP-hard, and levels that contain only Breakable Blocks, Spinies, and Fire Bars are PSPACE-hard.

 $\Longrightarrow$ NP-hard

 $\Longrightarrow$ PSPACE-hard

Prove that levels that contain only Breakable Blocks and Spinies are NP-hard.

 $\Longrightarrow$ NP-hard