complexiteit en spellen

complexiteit 2024

Hendrik Jan Hoogeboom

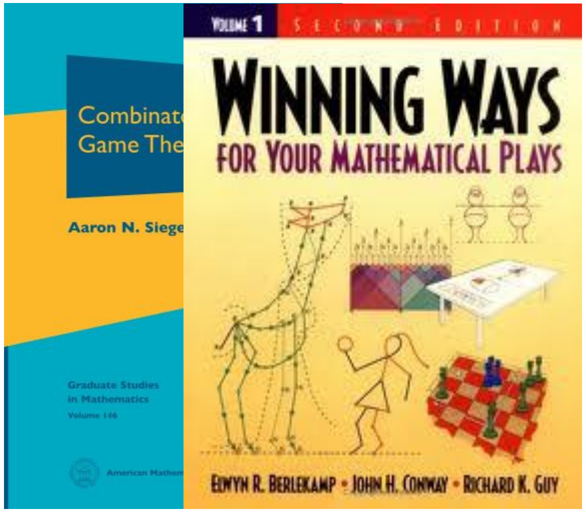
15 mei 2024

Gastcollege bij Complexiteit
(2024, J.de Graaf en L.Edixhoven).
Geen tentamenstof.

Thema: Complexiteitsklassen die ‘horen’ bij puzzels en spellen.
Naar het boek van Hearn en Demaine over “Constraint Logic”.

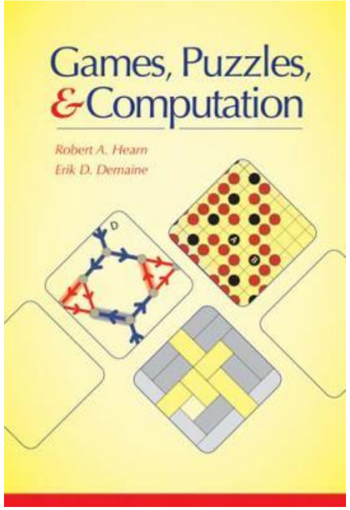
Eerder ook in Seminar Combinatorial Algorithms (W.Kosters en HJ.Hoogeboom).

‘game theory’ fields



combinatorial game theory
algorithms
mathematical theory

economic game theory
von Neumann, Nash
strategy, optimization expected profit



computational complexity
models of computation: *games*
turing machine

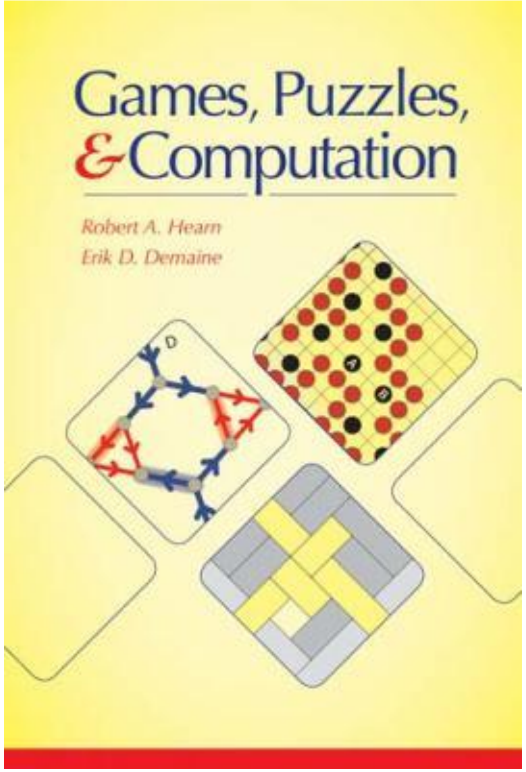
https://liacs.leidenuniv.nl/~kosterswa/semalg/editions.html

reference

Games, Puzzles,
& Computation

Robert A. Hearn
Erik D. Demaine

(2009, AKPeters)



E. Demaine and R.A. Hearn. Constraint Logic: A Uniform Framework for Modeling Computation as Games. In: Proceedings of the 23rd Annual IEEE Conference on Computational Complexity, June 2008.

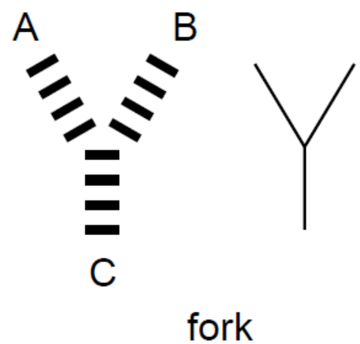
R.A. Hearn. Games, Puzzles, and Computation PhD thesis, MIT, 2006.

youtube “a computer built with dominos” (2020)

domino computing

Computing with Planar Toppling Domino Arrangements

William M. Stevens



challenge:
(no) timing & (no) bridges

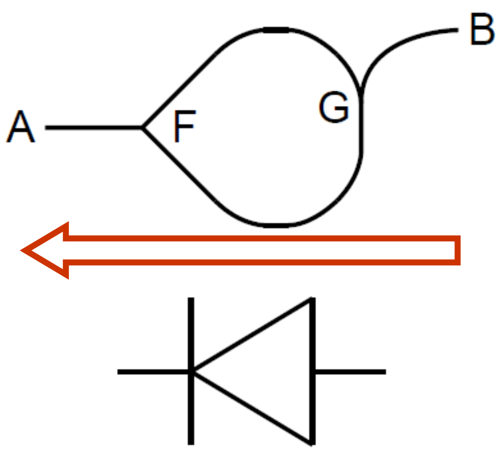


Fig. 3. A one way line

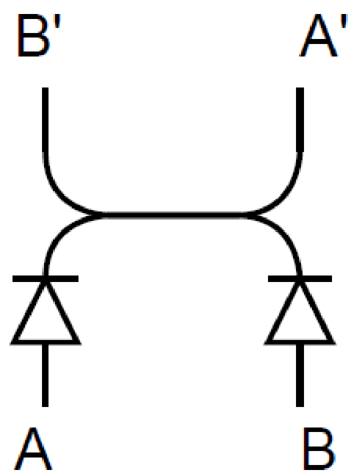


Fig. 4. A single line crossover

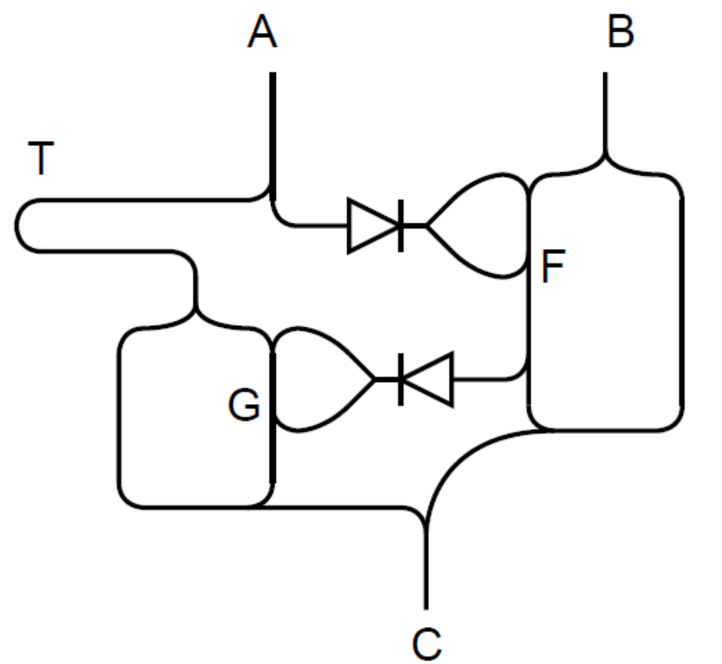
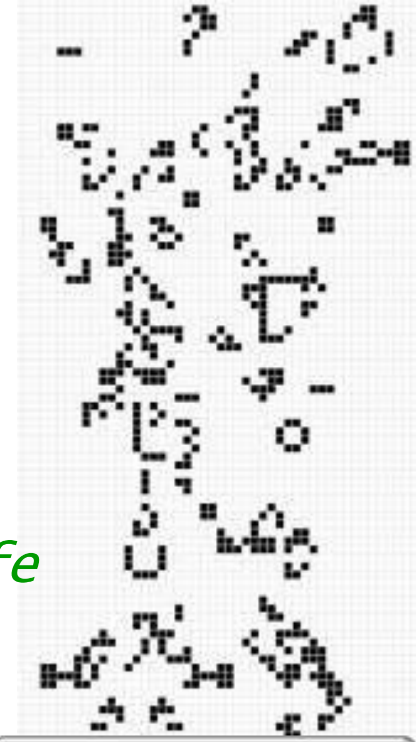


Fig. 5. A both mechanism

Unconventional Computation, 10th International Conference
UC 2011, Turku, Finland, June 6-10, 2011

what is a game?



characteristics
- bounded state
- moves, repetition
- players
- goal

study the complexity of

- simulation (0p) domino, *game of life*
- puzzles (1p) *rush hour*
- board games (2p) ‘generalized’ *chess*
- teams

We gaan uit van een eindig aantal toestanden.
We onderscheiden het aantal spelers

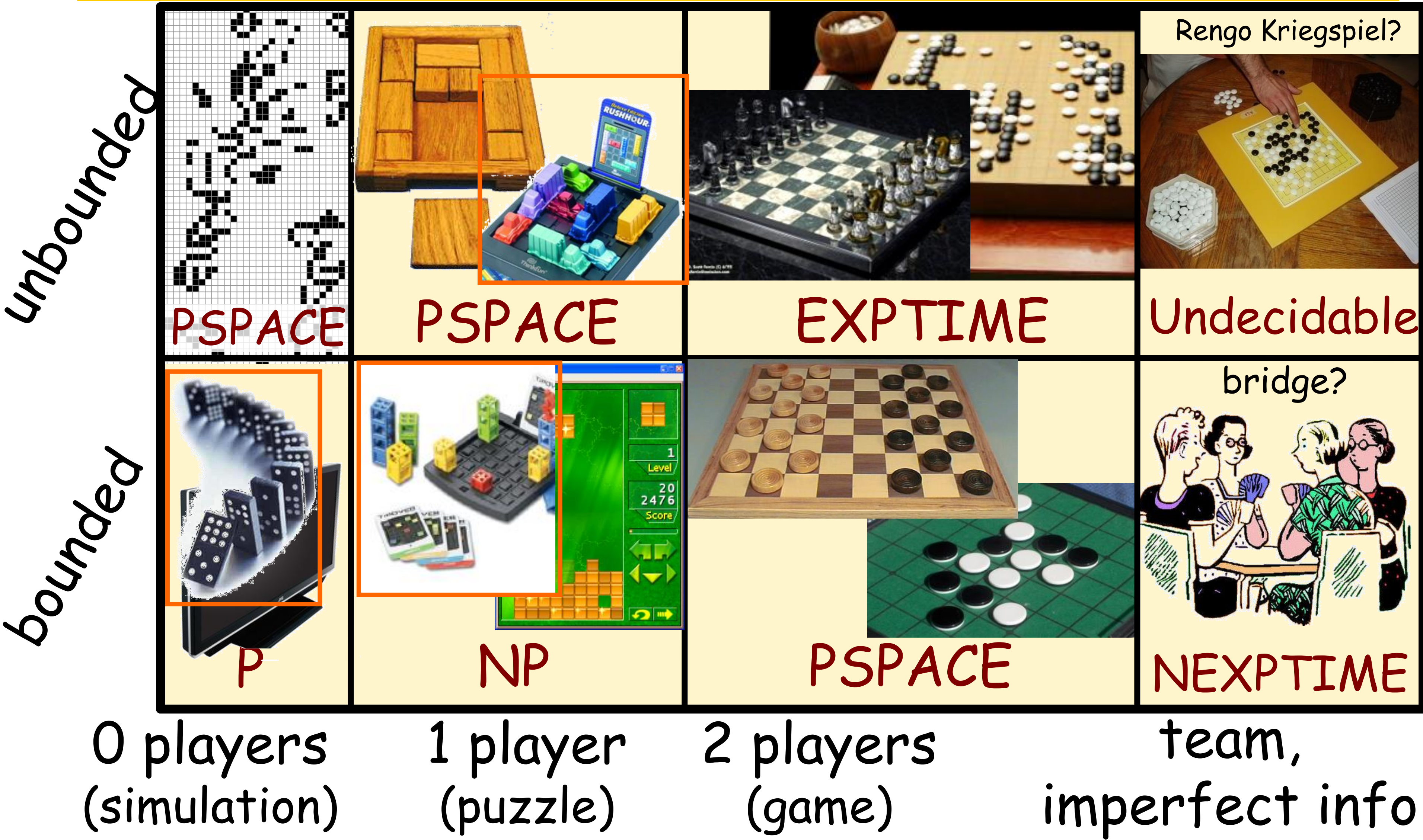
- nul: simulaties
- één: puzzels
- twee: spellen

en ook of zetten wel/niet herhaald mogen worden

- bounded / unbounded

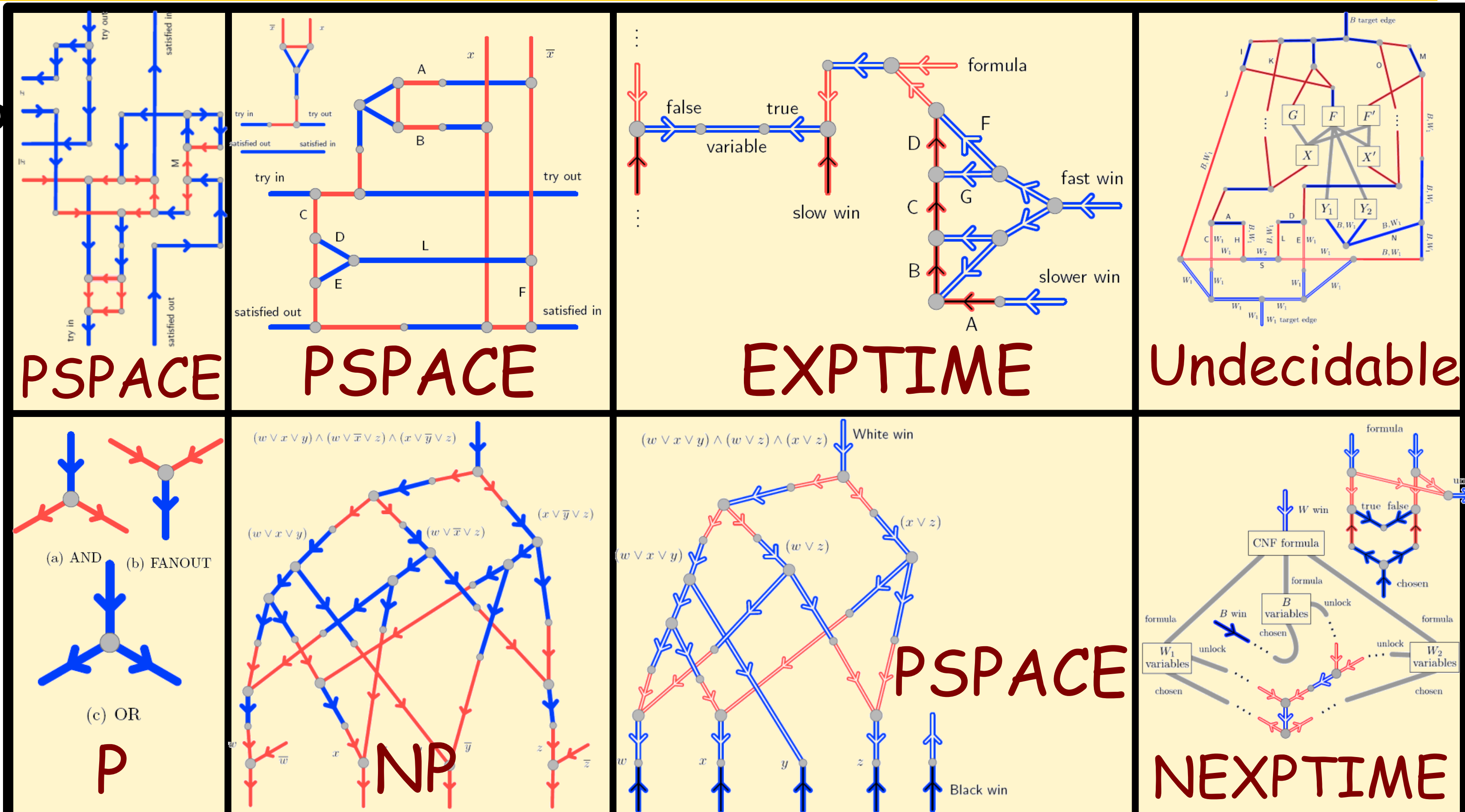
Daarbij horen zes klassen spellen en passende complexiteitsklassen.
Voor elke klasse is een Constraint Logic die deze complexiteit heeft, en gebruikt kan worden om naar concrete spellen te reduceren.
Constraint Logic is een spel/puzzel waarin takken in een graaf kunnen worden omgedraaid (volgens simpele regels)

Complexity of Games & Puzzles
[Demaine, Hearn & many others]



Constraint Logic

[Hearn & Demaine 2009]



unbounded

bounded

0 players (simulation) | 1 player (puzzle) | 2 players (game) | team, imperfect info

We gaan eerst kijken naar
- tijdscomplexiteit (voor bounded)
- ruimtecomplexiteit (unbounded)
en
- deterministische,
- niet-deterministische en
- alternerende(!) TuringMachine berekeningen.

De bijbehorende complexiteitsklassen passen precies op de zes klassen van spellen.

deel 1: tijd- en ruimtecomplexiteit

boeken

algemeen leerboek (ook talen en automaten):
Michael Sipser, Introduction to the Theory of Computation.

Ch.7&8 Time & Space complexity.
Ch.10.3 beschrijft Alternation.

gevorderd:
Arora and Barak, Computational Complexity A modern approach. Cambridge University Press. 2009

maar eerst …

- Complexiteit (dit vak)
  - → hoeveel werk kost deze oplossing? (tijd)complexiteit
  - → hoeveel geheugen? ruimtecomplexiteit

basic complexity classes

game complexity classes
vs.
TM resources: *space* & *time*

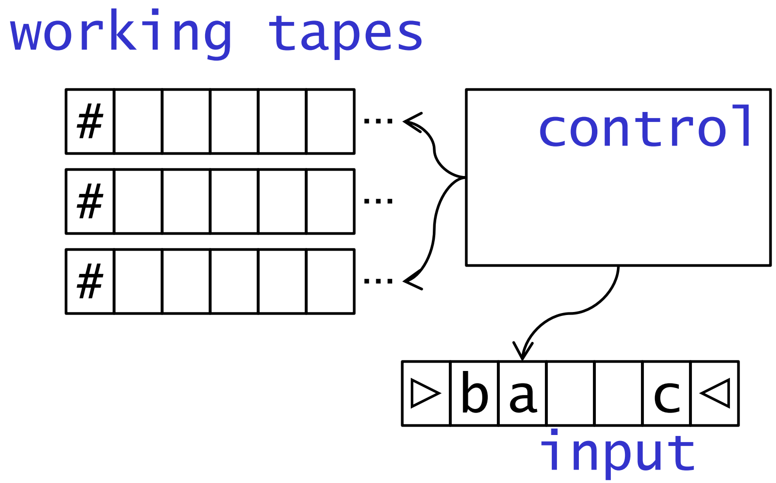
Cook/Levin	NP completeness SAT
Savitch	PSPACE = NPSPACE

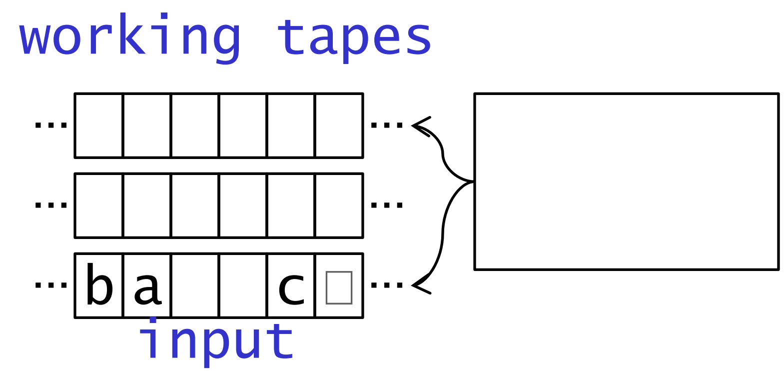
Er zijn verschillende TM modellen mogelijk. Invoer tape, meerdere werktapes, eenzijdig, dubbelzijdig.
Voor ruimte complexiteit onderscheiden we een werktape naast de invoertape. Dit maakt logaritmische complexiteit mogelijk: véél minder schrijfruimte dan de invoer lang is.

Polynomiale ruimte complexiteit
- PSPACE
- NPSPACE niet deterministische berekeningen

Savitch: ruimte kan hergebruikt worden, we kunnen nauwkeurig alle berekeningen ‘recursief’ nagaan.
PSPACE = NPSPACE (!)

TM models (H&U)





<u>space complexity</u>

DSPACE(f) NSPACE(f)

offline
multiple working tapes
single side infinite

<u>time complexity</u>

DTIME(f) NTIME(f)

input on tape
multiple working tapes
double sided

for every input word of length n, …

M scans at most f(n) cells on any storage tape …

M makes at most f(n) moves before halting …

Savitch

$\text{NSPACE}(\ s(n)\ ) \subseteq \text{SPACE}(\ s^2(n)\ )$

can we reach a halting configuration?
at most exponentially many steps $s(n)|\Sigma|^{s(n)}$

solve recursively "re-use space"

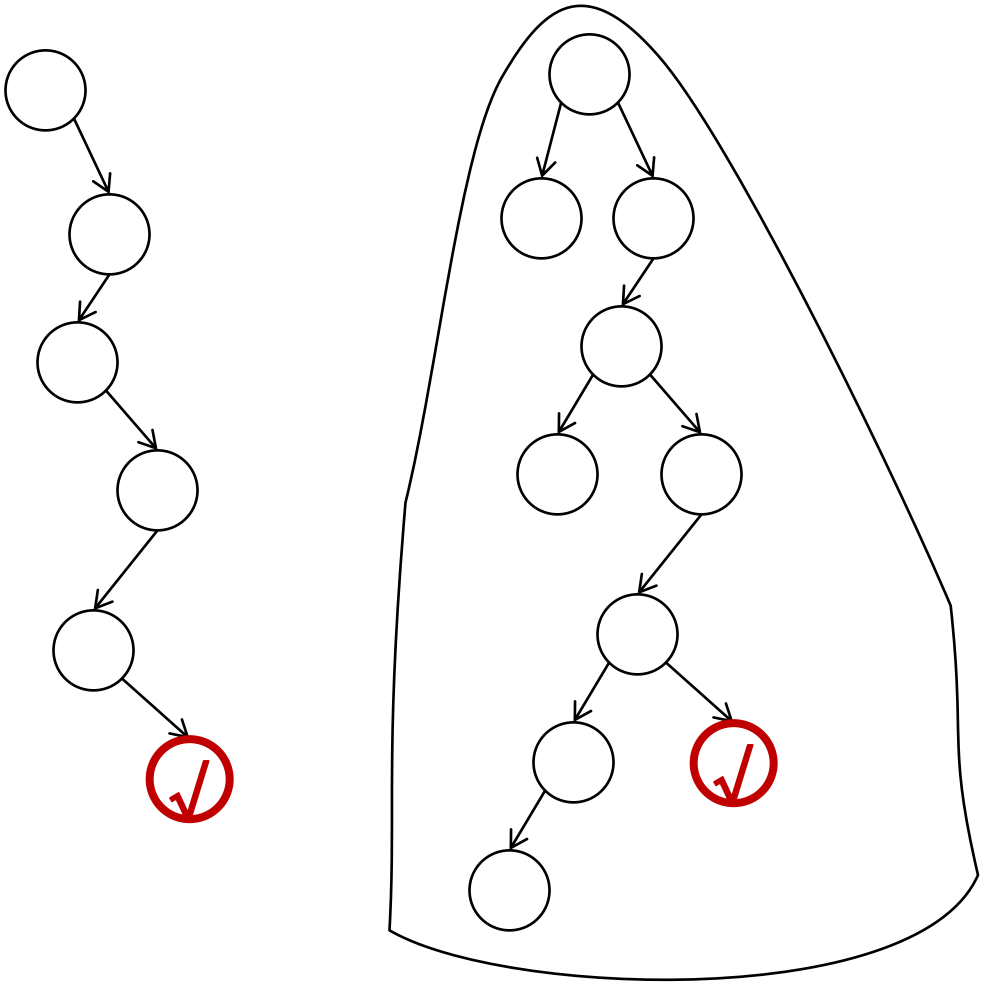
reach(*ini*,*fin*,1) = step(*ini*,*fin*)
reach(*ini*,*fin*,2k)
  foreach configuration mid
    test reach(*ini*,*mid*,k) ∧ reach(*mid*,*fin*,k)

stack depth s(n) of configs, each size s(n)

NPSPACE = PSPACE

$\text{NSPACE}(\ s(n)\ ) \subseteq \text{ATIME}(\ s^2(n)\ )$ "parallel in time"

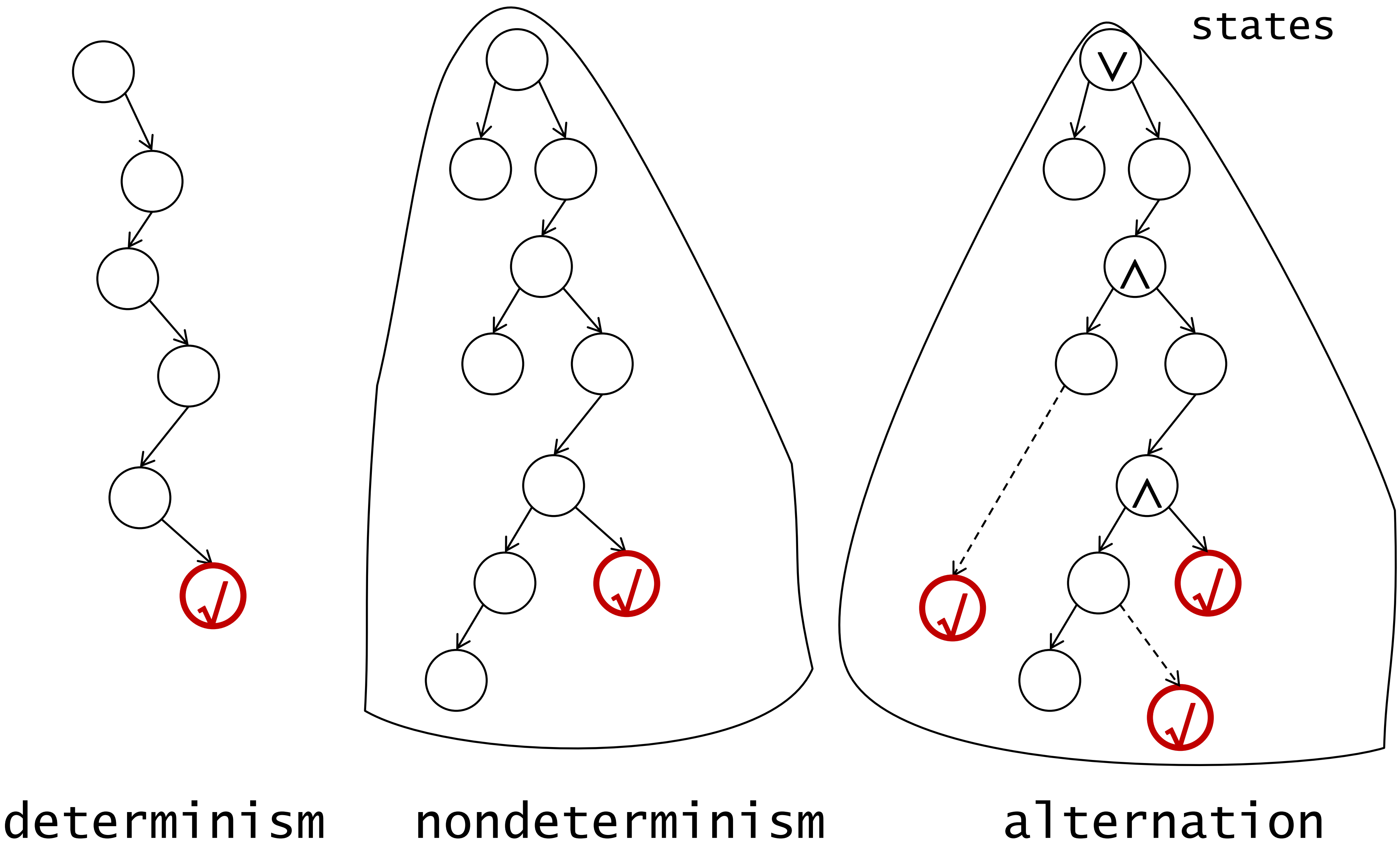
computation tree



determinism nondeterminism

computation tree

existential and *universal* states



niet-det TM: existentieel: kan er een accepterende berekening gevonden worden.

Alternerend: ook universeel: elke volgende stap heeft een accepterende berekening.

P, NP, AP
PSPACE=NPSPACE, APSPACE

Óók LOGSPACE=L, NL, AL

Alterneren is krachtig, stapt een nivo omhoog
AL=P, AP=PSPACE, APSPACE=EXPTIME
Gelijkheid, wonderlijk in complexiteitsland.
Constructie door de hele berekeningsboom na te rekenen.

$NSPACE(s(n)) \subseteq ATIME(s^2(n))$ "parallel in time"

$reach_{2k}(c_1, c_2)$

exists configuration c

write in time s(n)

$reach_k(c_1, c) \wedge reach_k(c, c_2)$

parallel in time

(!!) technisch is correct: $\bigcup_c ATIME(c \cdot s^2(n))$

$ATIME(s(n)) \subseteq DSPACE(s(n))$

simulate computation tree

dimensions

existential and *universal* states
computation = tree

	log.	*polynomial*		*exp.*
	space	time	space	time
determinism	L	P	PSPACE	EXPTIME
nondeterminism	NL	NP	NPSPACE	NEXPTIME
alternation	AL	AP	APSPACE	AEXPTIME

AL AP APSPACE AEXPTIME

P ⊆ NP ⊆ PSPACE ⊆ EXPTIME ⊆ NEXPTIME ⊆ EXPSPACE

NPSPACE NEXPSPACE

A.K. Chandra, D.C. Kozen, and L.J. Stockmeyer. 'Alternation', Journal of the ACM, Volume 28, Issue 1, pp. 114–133, 1981.

Cook-Levin: SAT is NP-compleet probleem.

deterministische TM: HORNSAT is P-compleet.
HORN clause: max één positieve litteral.

Let op: P-compleet maakt gebruik van (bv) *log-space reducties*. Met pol-time reducties zijn (bijna) alle P-problemen equivalent. ☺

NC = Nick's class is gemotiveerd door logaritmische tijd polynomiale grootte parallele circuits, zie bijvoorbeeld de lezing van Demaine
Demaine MIT Complexity: P-complete

literal

(negated) variable

clause

$(w \vee x \vee y) \wedge (w \vee \neg x \vee z) \wedge (x \vee \neg y \vee z) \wedge \ldots$

3 conjunctive normalform

3SAT

given: given formula ϕ in 3CNF

question: is ϕ satisfiable?

(can we find a variable assignment making formula true)

Cook/Levin 3SAT is NP-complete

$T_{iak} \rightarrow \neg T_{ibk}$ $\quad$ $T_{iak} \wedge T_{ib.k+1} \rightarrow H_{ik}$

$H_{ik} \wedge Q_{pk} \wedge T_{iak} \rightarrow \bigvee_{(p,a,q,b,d)} H_{i+d.k+1} \wedge Q_{q.k+1} \wedge T_{ib.k+1}$

CNF conjunctive normal form
clauses: disjunction literals

$a \rightarrow \neg b$ iff $\neg a \vee \neg b$

$a \wedge b \rightarrow c$ iff $\neg a \vee \neg b \vee c$

$a \wedge b \wedge c \rightarrow \bigvee_I (d_I \wedge e_I \wedge f_I)$ sat-iff*

$(\neg a \vee \neg b \vee \neg c \vee \bigvee_I z_I) \wedge$

$\bigwedge_I (\neg z_I \wedge d_I) \wedge \bigwedge_I (\neg z_I \wedge e_I) \wedge \bigwedge_I (\neg z_I \wedge f_I)$

3SAT is NP hard

$(a \vee b \vee c \vee d \vee e)$ sat-iff*

$(a \vee b \vee x_1) \wedge (\neg x_1 \vee c \vee x_2) \wedge (\neg x_2 \vee d \vee e)$

$T_{iak} \rightarrow \neg T_{ibk}$ $\quad$ $T_{iak} \wedge T_{ib.k+1} \rightarrow H_{ik}$

$H_{ik} \wedge Q_{pk} \wedge T_{iak} \rightarrow \bigvee_{(p,a,q,b,d)} H_{i+d.k+1} \wedge Q_{q.k+1} \wedge T_{ib.k+1}$

determinism

HORN-SAT is P hard

CNF conjunctive normal form

Horn clauses: at most one positive literal

$a \rightarrow \neg b$ iff $\neg a \vee \neg b$

$a \wedge b \rightarrow c$ iff $\neg a \vee \neg b \vee c$

$a \wedge b \wedge c \rightarrow (d \wedge e \wedge f)$ iff

$(\neg a \vee \neg b \vee \neg c \vee d) \wedge \ldots \wedge (\neg a \vee \neg b \vee \neg c \vee f)$

HORN-SAT is in P

unit propagation

empty formula: formula True
empty clause: formula False

unit clause (ℓ): set ℓ True
remove clauses with ℓ
remove $\neg\ell$ from clauses

all remaining variables negative

Quantified Boolean Formula.
Reeks quantoren gevolgd door formule (conjunctie clausen)

Dit lijkt op spellen: Ik win als
er is een zet, zodat
voor alle zetten van de tegenstander,
er is een zet, zodat …

QBF: is de formule waar?
Dit probleem is PSPACE compleet.

L.J. Stockmeyer, A.R. Meyer. ‘Word problems requiring exponential time’, ACM STOC 1987, p.77-82.

formula games – complete problems

NL **2SAT**

$(x_1 \vee x_3) \wedge (\neg x_5 \vee \neg x_3) \wedge (x_5 \vee x_1)$

P **HORN-SAT**

$(\neg x_3 \vee \neg x_2 \vee \neg x_5 \vee x_1)$ i.e. $(x_3 \wedge x_2 \wedge x_5 \rightarrow x_1)$

NP **SAT** satisfiablity

$\exists x_1\ \exists x_3\ \exists x_5\ (x_1 \vee x_3 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_3) \wedge (x_5 \vee x_1)$

(N)PSPACE **QBF** aka QSAT

$\exists x_1\ \forall x_3\ \exists x_5\ (x_1 \vee x_3 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_3) \wedge (x_5 \vee x_1)$

true quantified boolean formula

SAT is NP hard satisfiablity

$\exists x_1\ \exists x_3\ \exists x_5\ (x_1 \vee x_3 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_3) \wedge (x_5 \vee x_1)$

QBF is PSPACE hard aka QSAT

$\exists x_1\ \forall x_3\ \exists x_5\ (x_1 \vee x_3 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_3) \wedge (x_5 \vee x_1)$

input length $p(n)$ #steps $\leq c^{p(n)}$

configuration c sequence variables T_{iak} Q_{qk} H_{ik}

$\Phi_k(c_1,c_2)$ from c_1 to c_2 in $\leq k$ steps
k=1 see SAT construction Cook/Levin

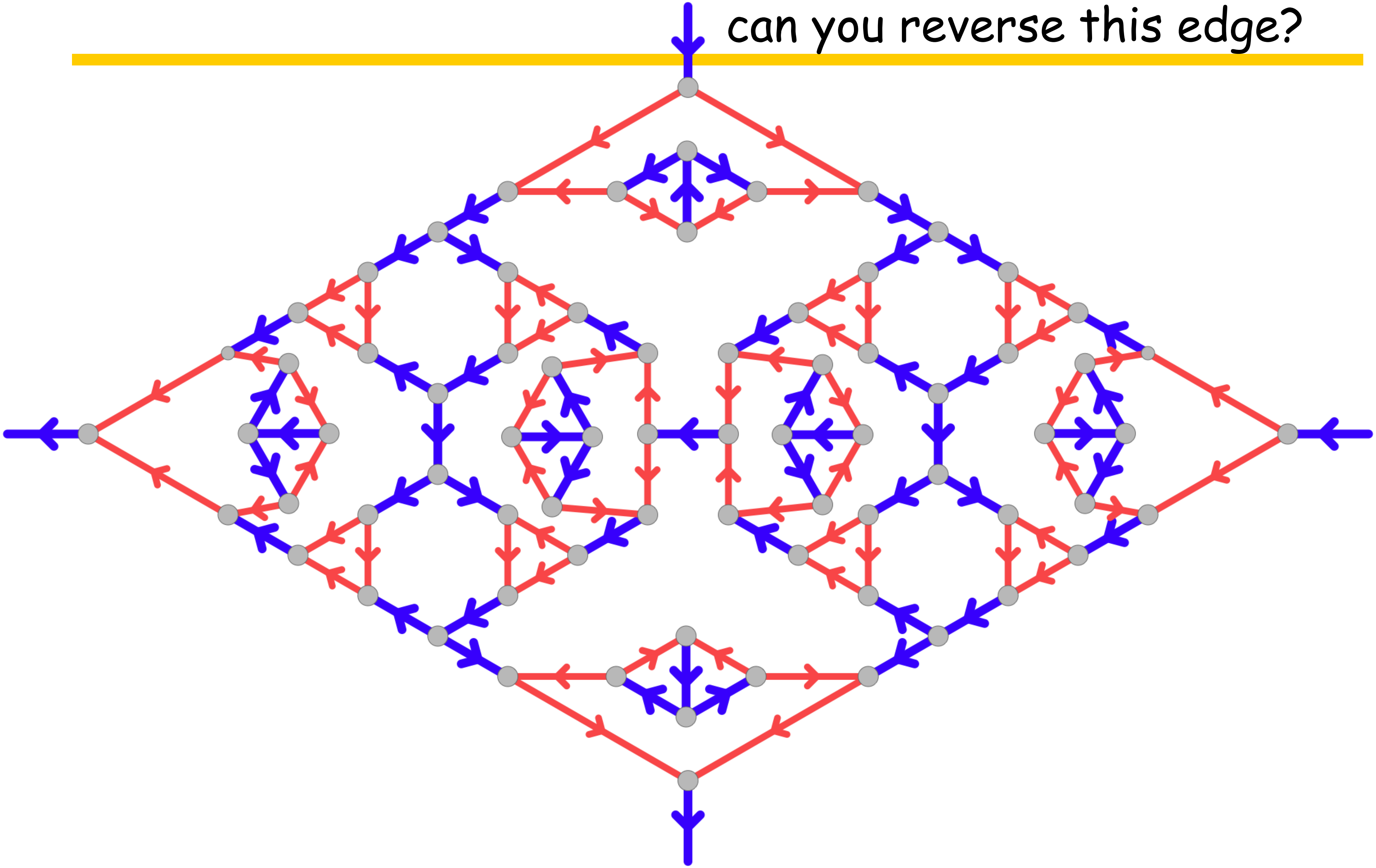
$\Phi_{2k}(c_1,c_2) = (\exists c)(\ \Phi_k(c_1,c) \wedge \Phi_k(c,c_2)\ )$
close, but *doubles size*

$\Phi_{2k}(c_1,c_2) = (\exists c)(\forall c')(\forall c'')$
$[\ (c',c'')=(c,c_1) \vee (c',c'')=(c_1,c)\ ] \rightarrow \Phi_k(c',c'')$

deel 2: constraint logic

Decision Problem

can you reverse this edge?



constraint logic

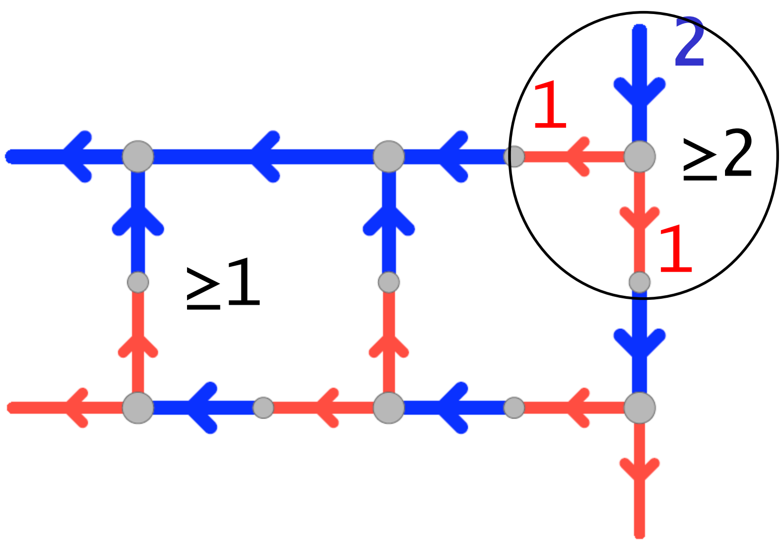
NCL - nondet constraint logic

instance: constraint graph G, edge e

question: sequence which reverses e

BOUNDED NCL

… reverses each edge *at most once*



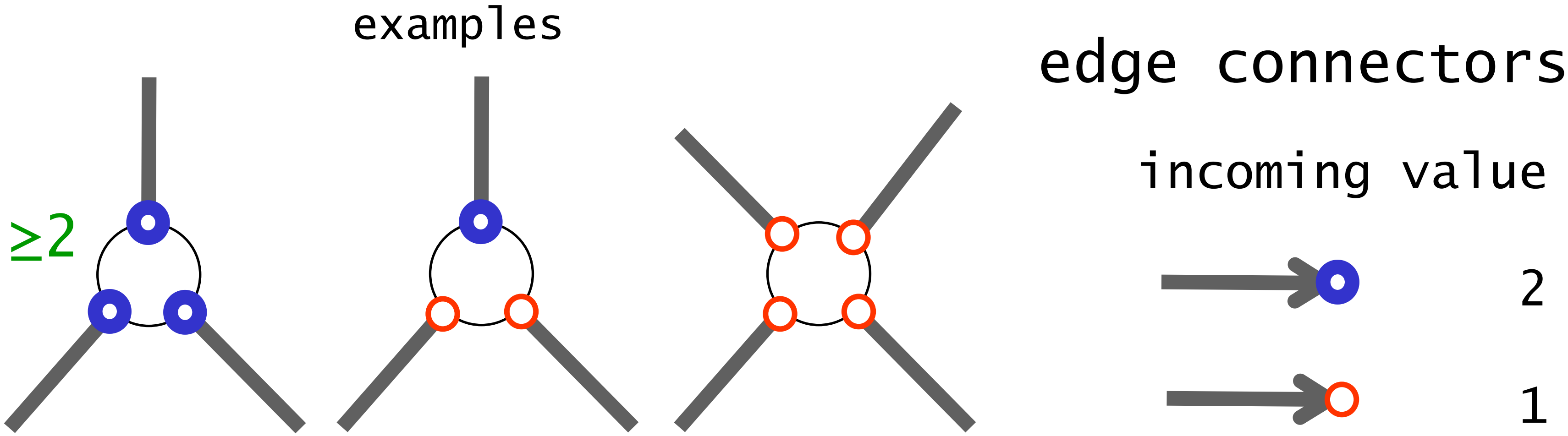
Constraint Logic.
Gerichte graaf.
Gekleurde takken/aanhechtingen.
Waarde rood=1, blauw=2.
Elke knoop heeft tenminste waarde twee ingaande takken (dus twee rood of één blauw)

- NCL ‘nondeterministic constraint logic’ past bij eenpersoons spellen=puzzels

Zet: omkeren tak, als aan de eis voldaan blijft.
Doel: omkeren speciale tak.

- BNCL ‘bounded’ elke tak maximaal één keer omkeren

basic constraint logic



constraint graph

oriented/directed edges + connectors

vertex constraint

inflow value ≥ 2

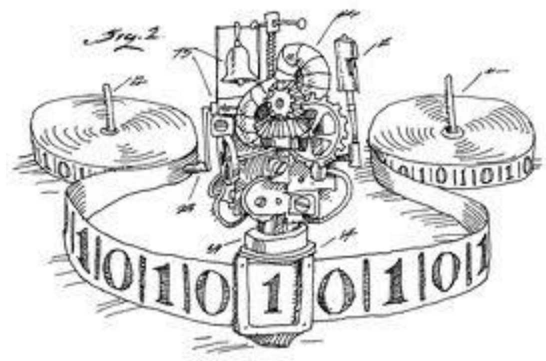
game: legal move

edge reversal satisfying constraint

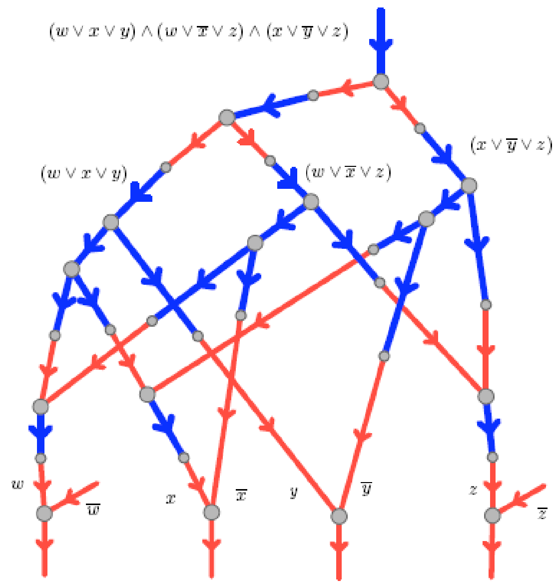
goal

reversal given edge

NP & TipOver



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



NP

3SAT

part I

constraint logic
'graph games'

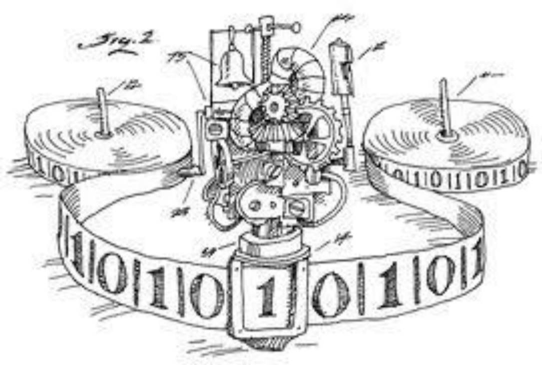
Bounded NCL

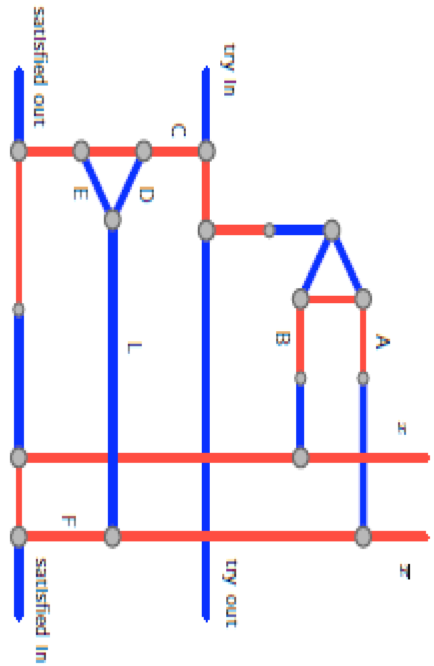
part II

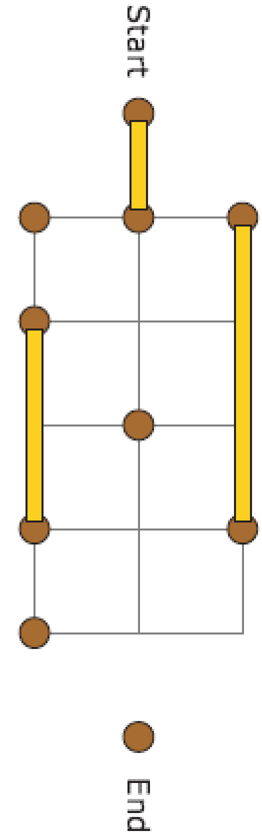
games in particular

TipOver

PSPACE & Plank Puzzle







PSPACE

QBF

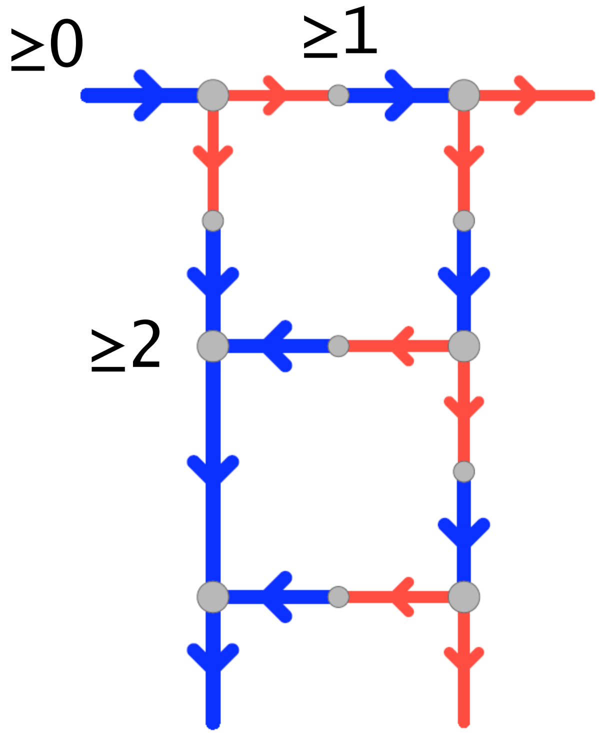
part I
constraint logic
‘graph games’

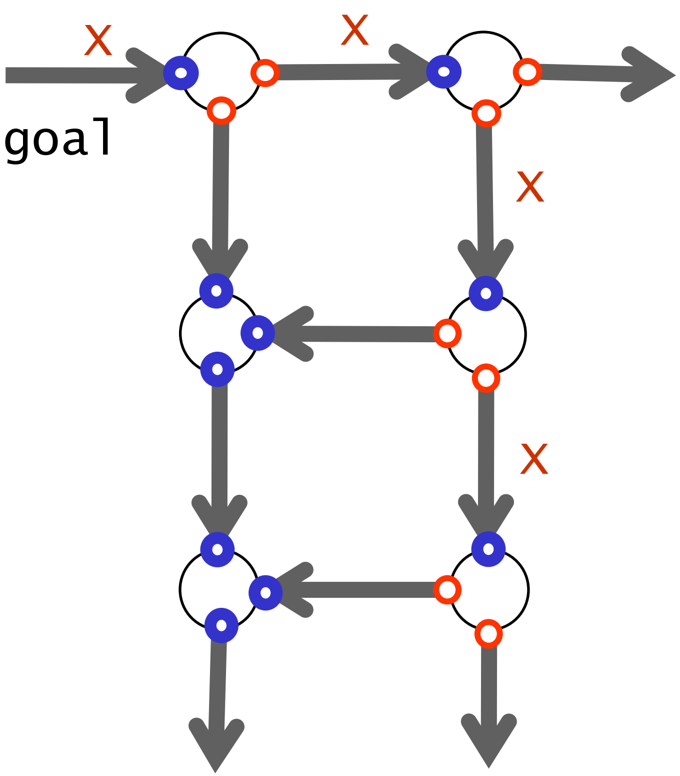
NCL

part II
games in particular

plank puzzle
(river crossing)

‘special’ vertex constraints?





colour conversion
dangling edges *edge terminators*

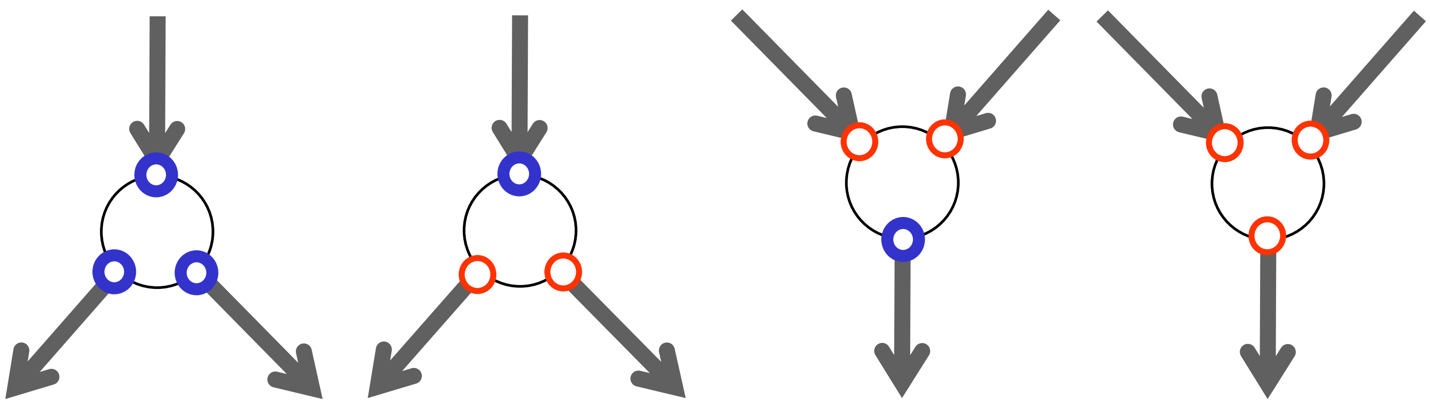
We kunnen een aantal soorten knopen onderscheiden, met een bepaalde intuitie.
OR, AND, FANOUT, CHOICE.

De state space van zo’n knoop voldoet aan die intuitie.
Bijvoorbeeld: om de tak boven de OR naar buiten te keren, moet tenminste één van de onderste twee naar binnen gekeerd worden.

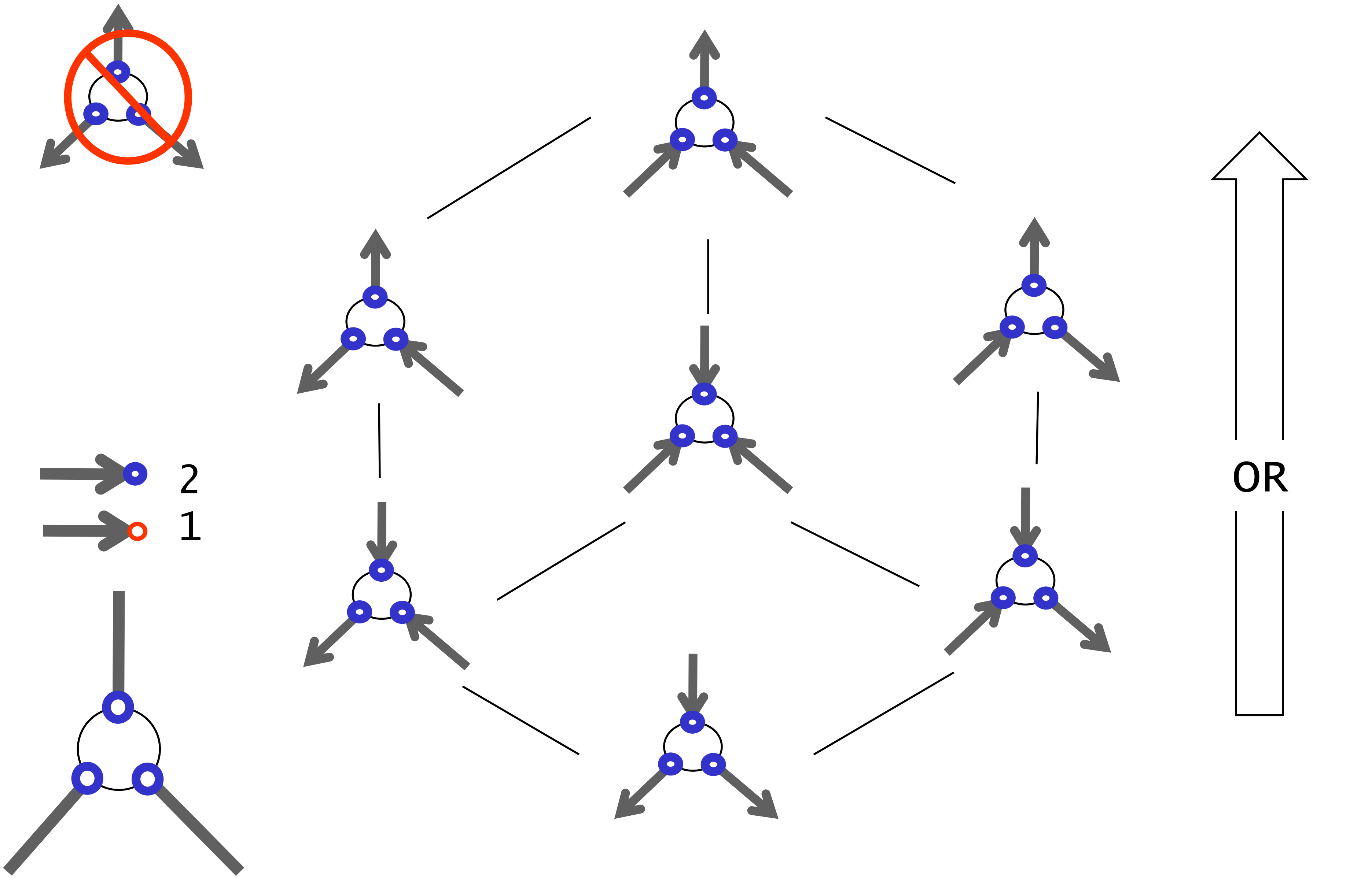
normal form vertices

NCL

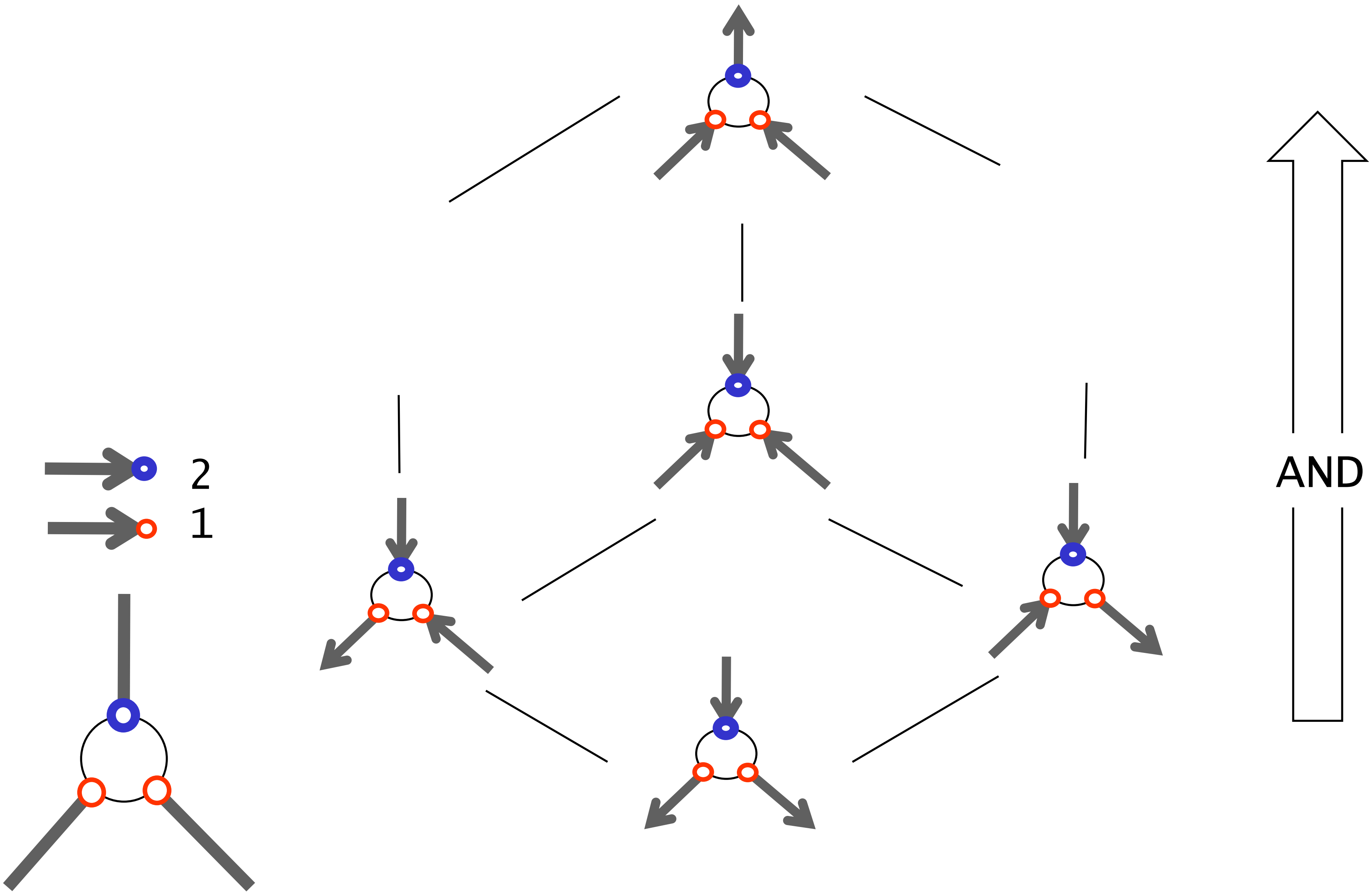
bounded NCL (reverse only once)



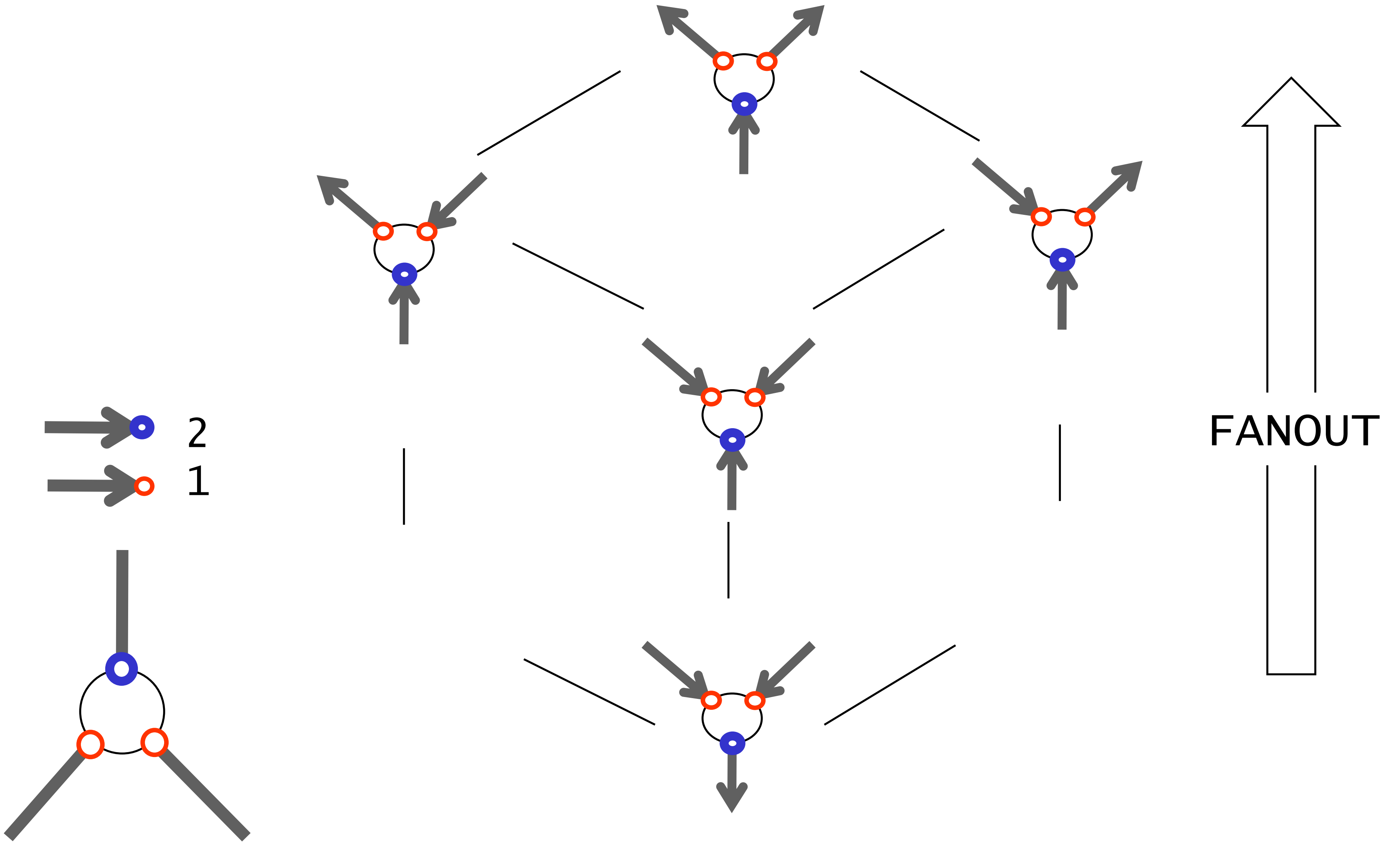
legal configurations



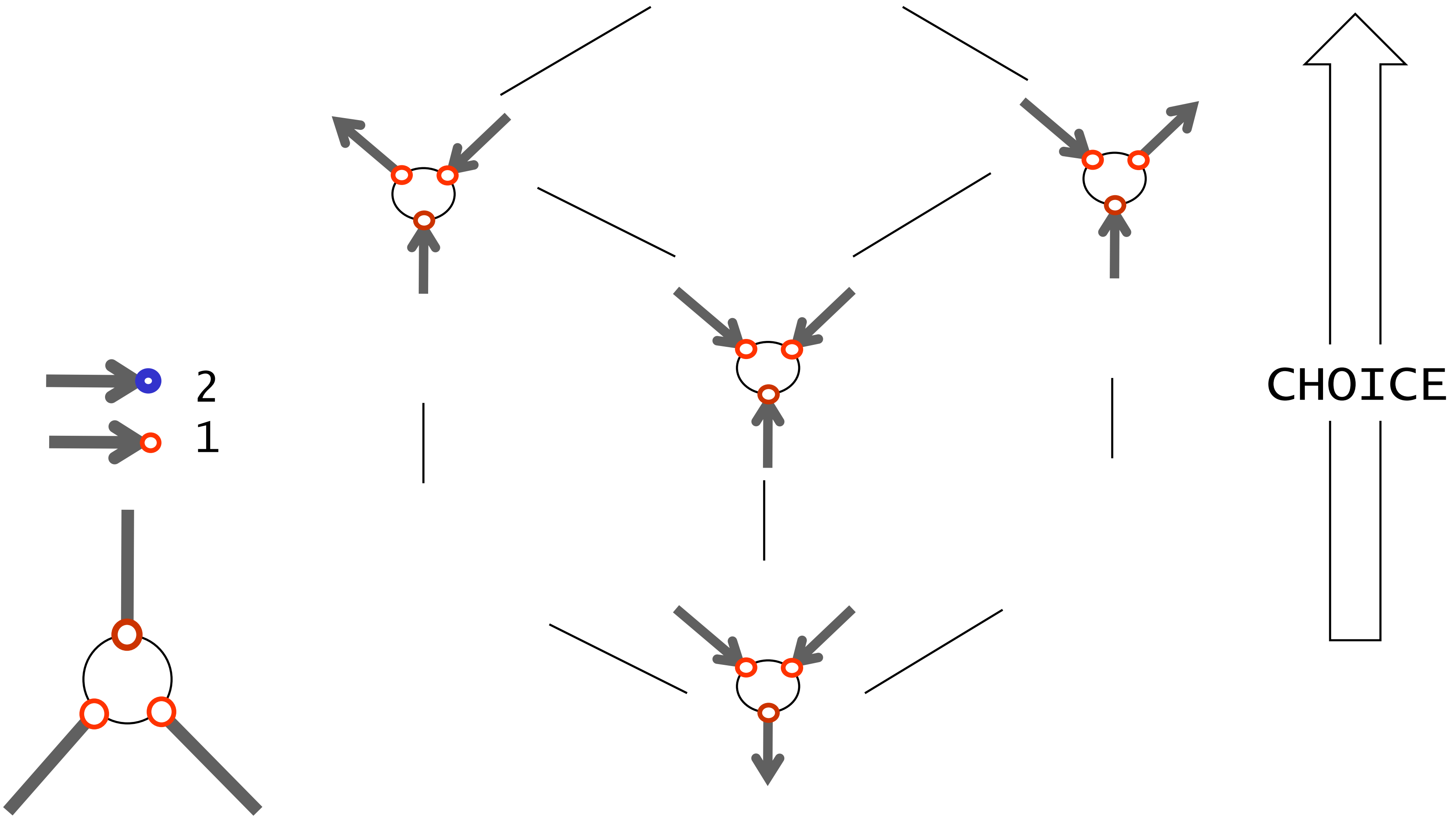
legal configurations



legal configurations



legal configurations



Met de vier soorten knopen kan eenvoudig een graaf gemaakt worden die een formule representeert.
De bovenste tak kan omgekeerd worden als de formule waargemaakt kan worden.
Onderaan geven de takken de waardering van de variabelen weer.

Dus BNCL simuleert SAT en is NP complete

Als we meerdere keren takken mogen omdraaien kunnen we gadgets maken die kwantoren nadoen.
(Niet erg snel in te zien.)

Dus NCL simuleert QBF/QSAT en is PSPACE complete

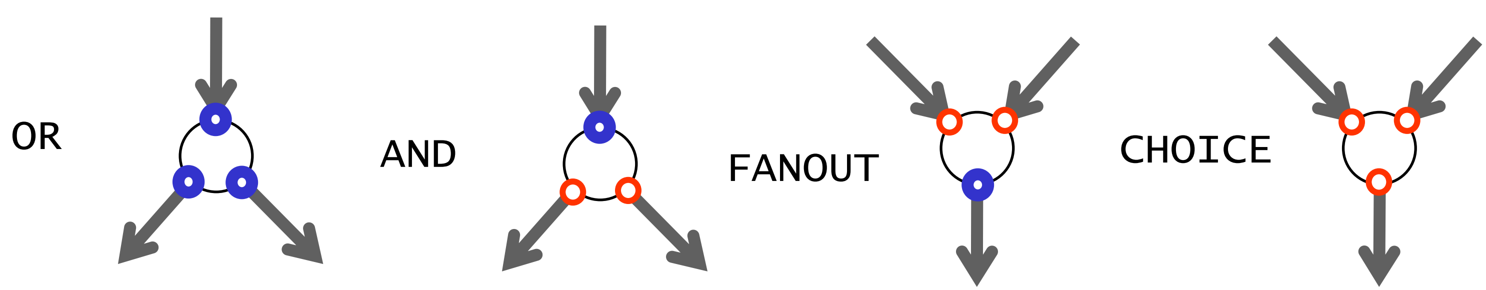
basic observation

“emulate” a logical formula as graph game

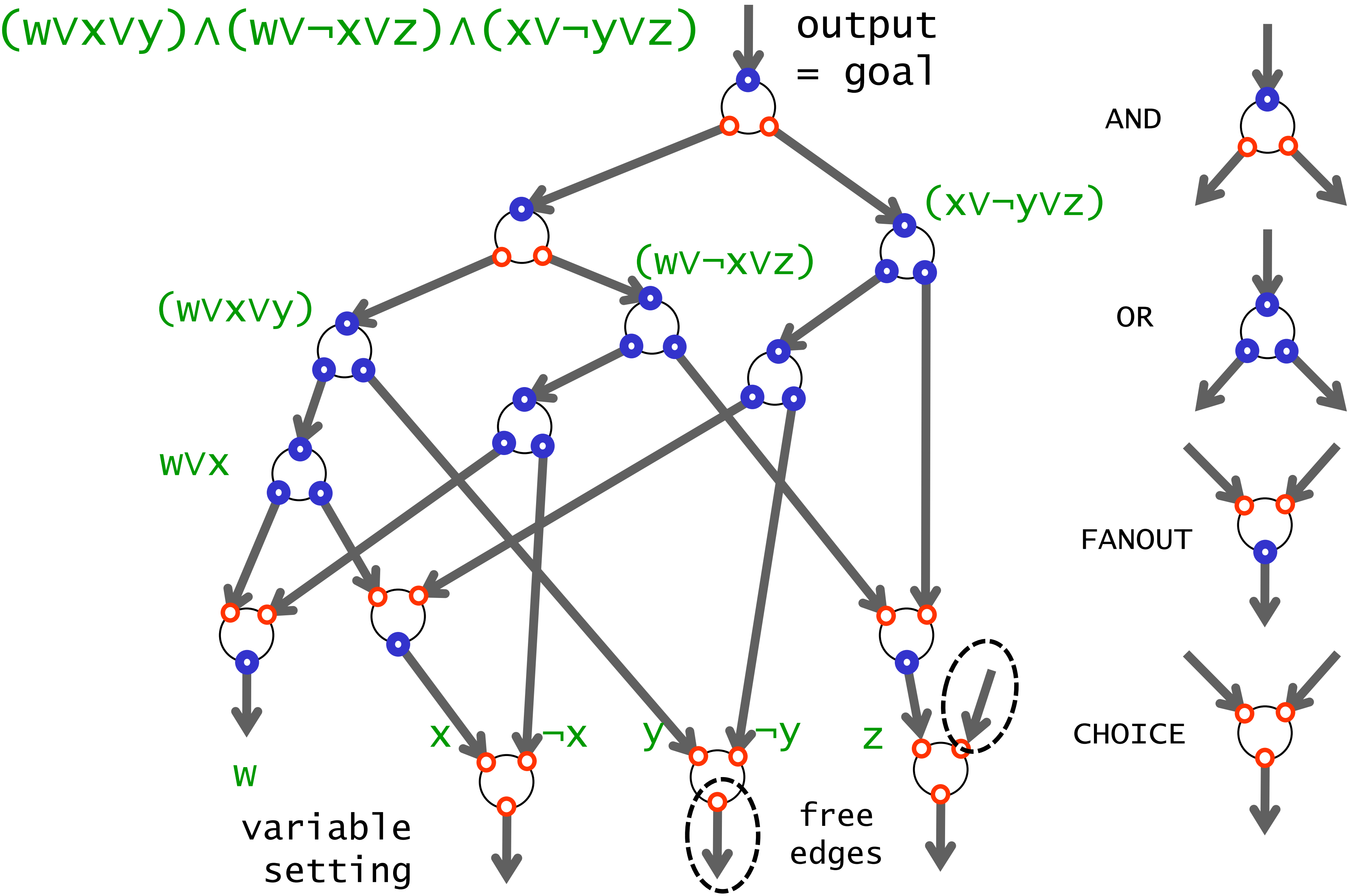
goal:
flip a given edge *iff* formula satisfiable

$$(w \vee x \vee y) \wedge (w \vee \neg x \vee z) \wedge (x \vee \neg y \vee z)$$

components

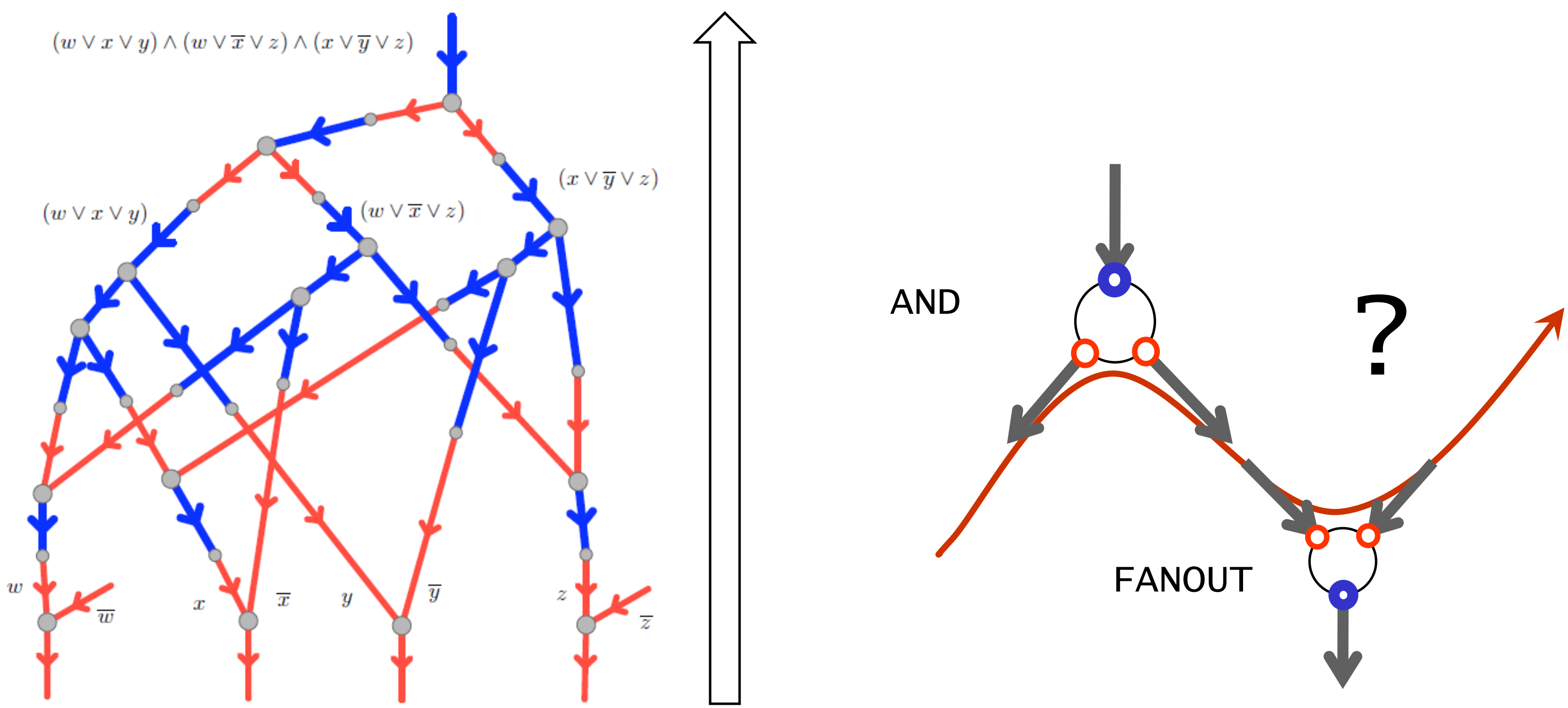


formula constraint graph



questions

- ‘can’ : not obliged to reverse edges upwards
  ie, we do not always set variable
- can we reverse the ‘wrong way’?
- do we need restriction to reverse edge once?



game categories

game categories and their natural complexities

(polynomial)
TM resources

Rush Hour
River Crossing

unbounded **SPACE**	**PSPACE**	**PSPACE** NPSPACE	**EXPTIME** APSPACE	undecid
bounded **TIME**	**P**	**NP**	**PSPACE** AP	NEXPTIME
#	**zero** *simulation* ***determ.***	**one** *puzzle* ***nondeterm.***	**two** *game* ***alternat.***	team *imperfect informat.*

TipOver

$NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXPTIME \subseteq NEXPTIME$

= NPSPACE = AP

formula games – complete problems

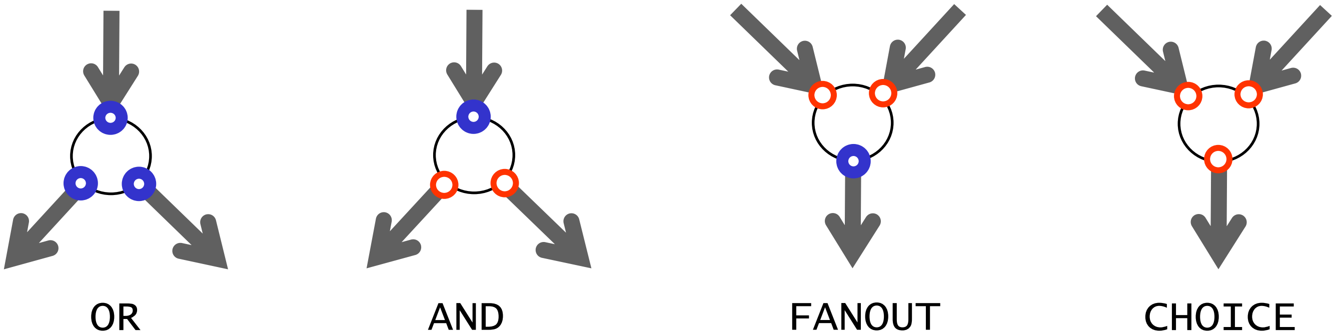
NP **SAT** satisfiablity

$\exists x_1 \ \exists x_3 \ \exists x_5 \ (x_1 \vee x_3 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_3) \wedge (x_5 \vee x_1)$

(N)PSPACE **QBF** aka QSAT

$\exists x_1 \ \forall x_3 \ \exists x_5 \ (x_1 \vee x_3 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_3) \wedge (x_5 \vee x_1)$

conclusion (B-NCL)



BOUNDED NCL - nondet constraint logic

instance: constraint graph G, edge e

question: sequence which reverses *each edge at most once*, ending with e

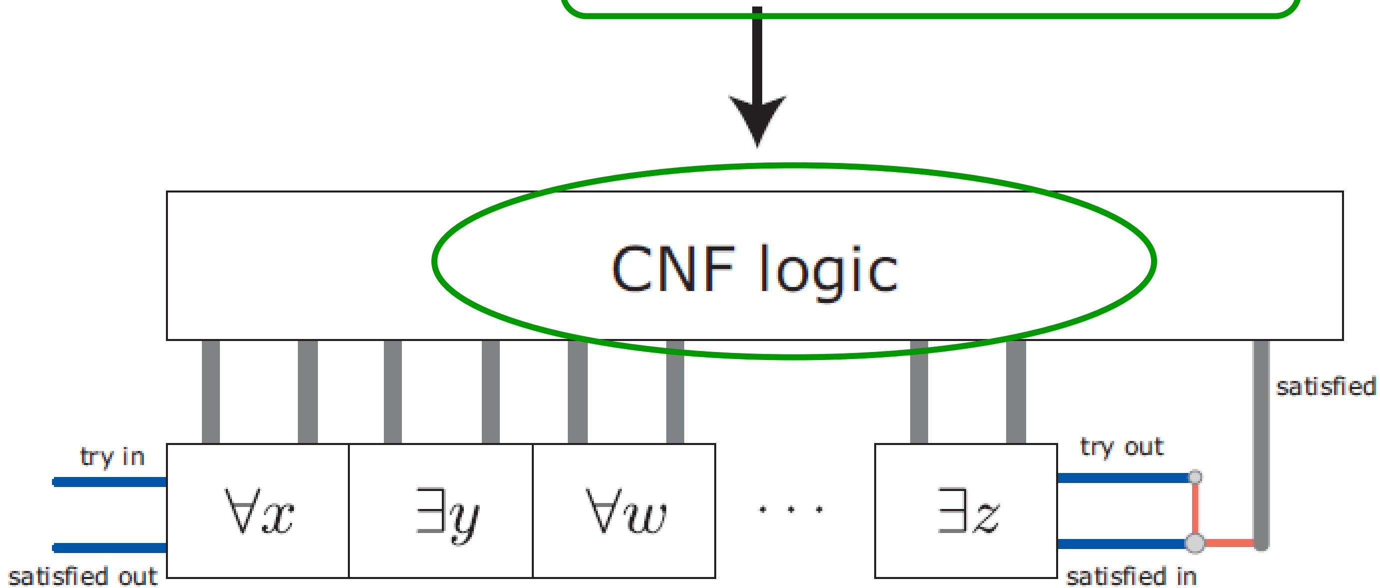
- reduction from 3SAT into Bounded NCL
- Bounded NCL is in NP

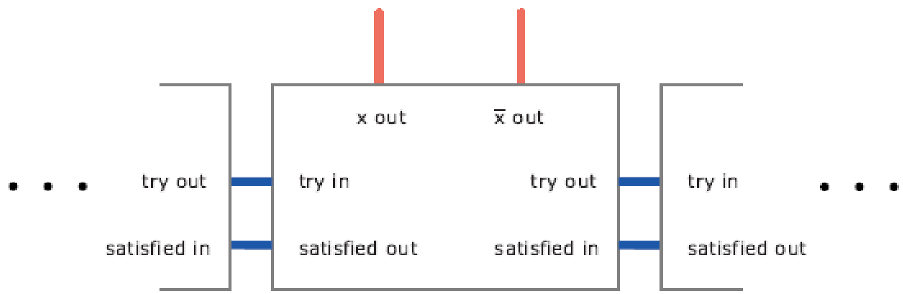
thm. Bounded NCL is NP-complete

however: topling domino's cannot cross

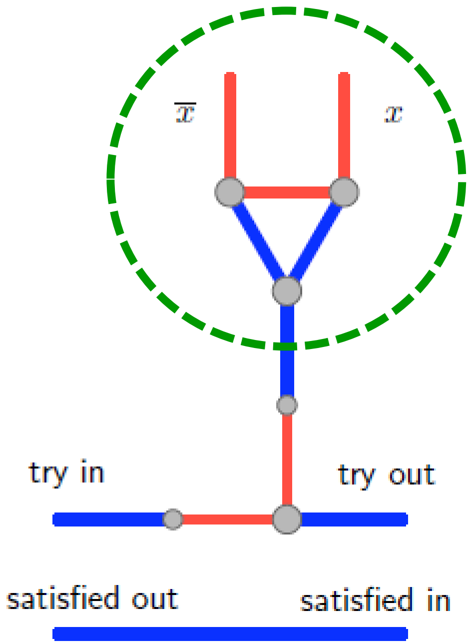
quantification

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

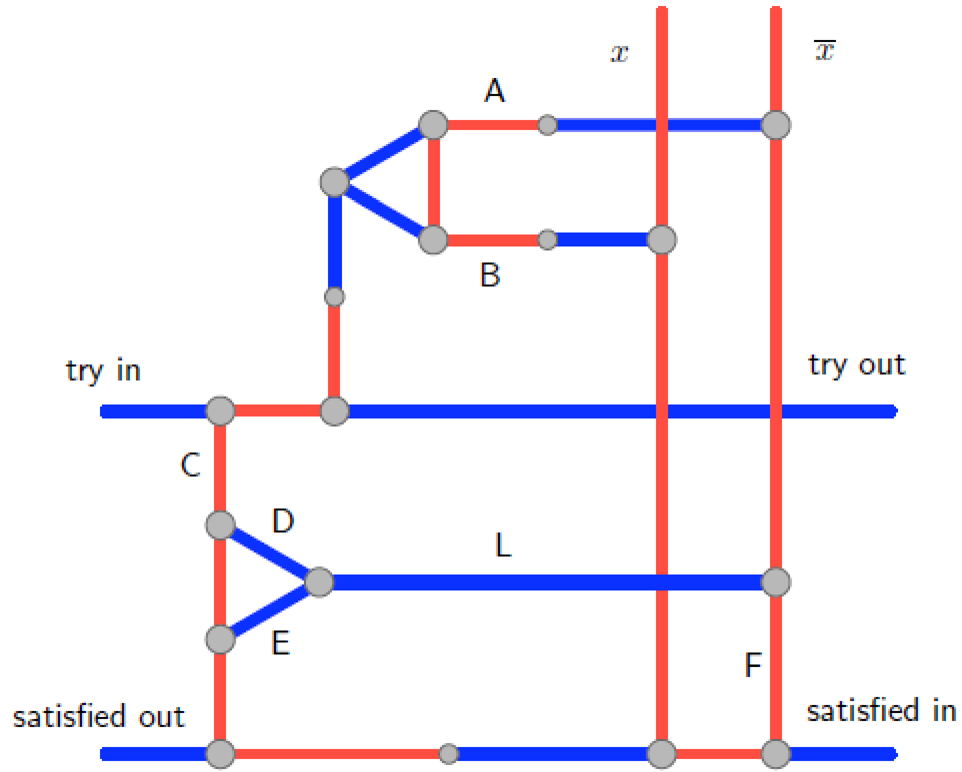




quantifier gadgets

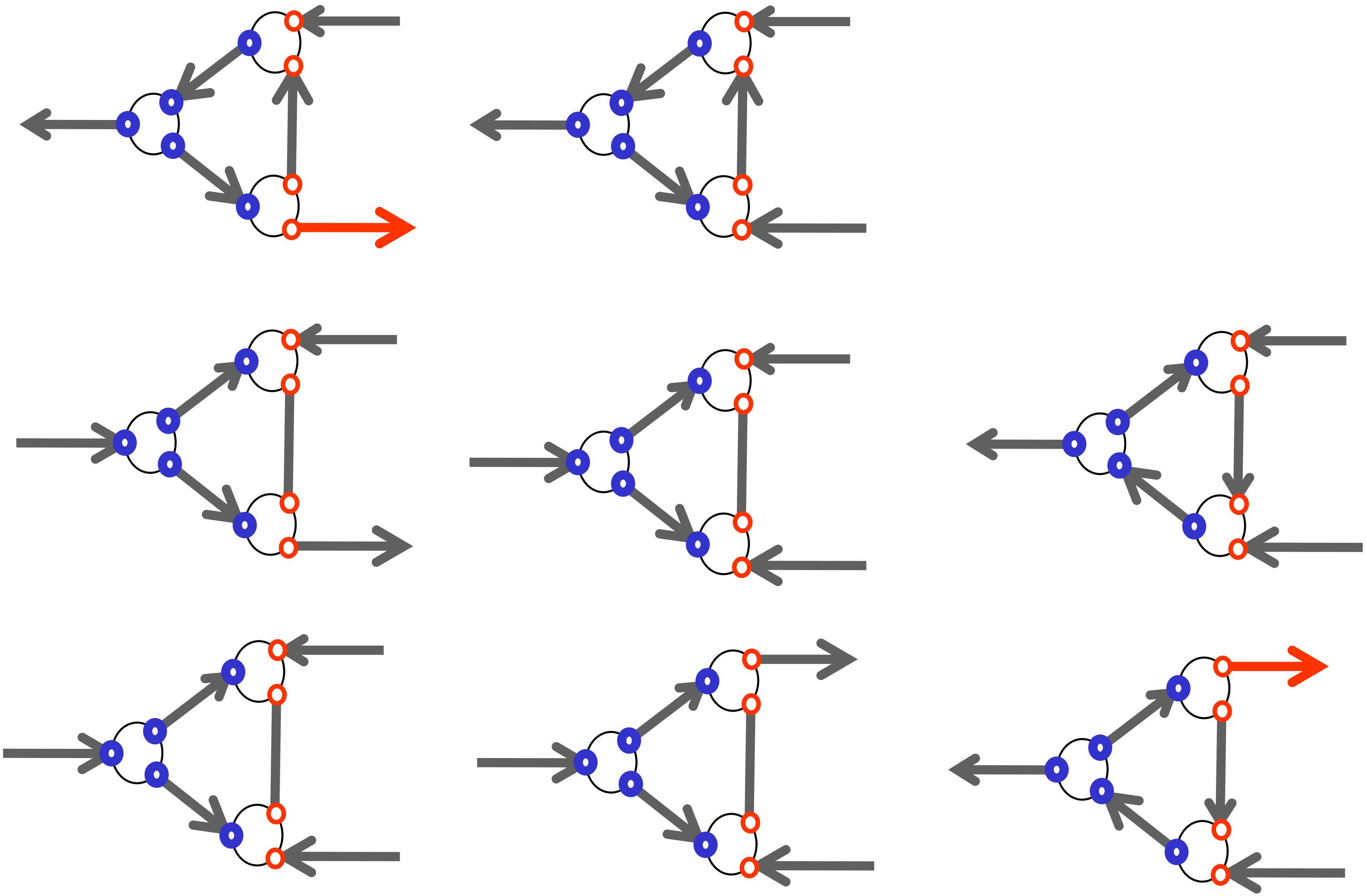


(a) Existential quantifier

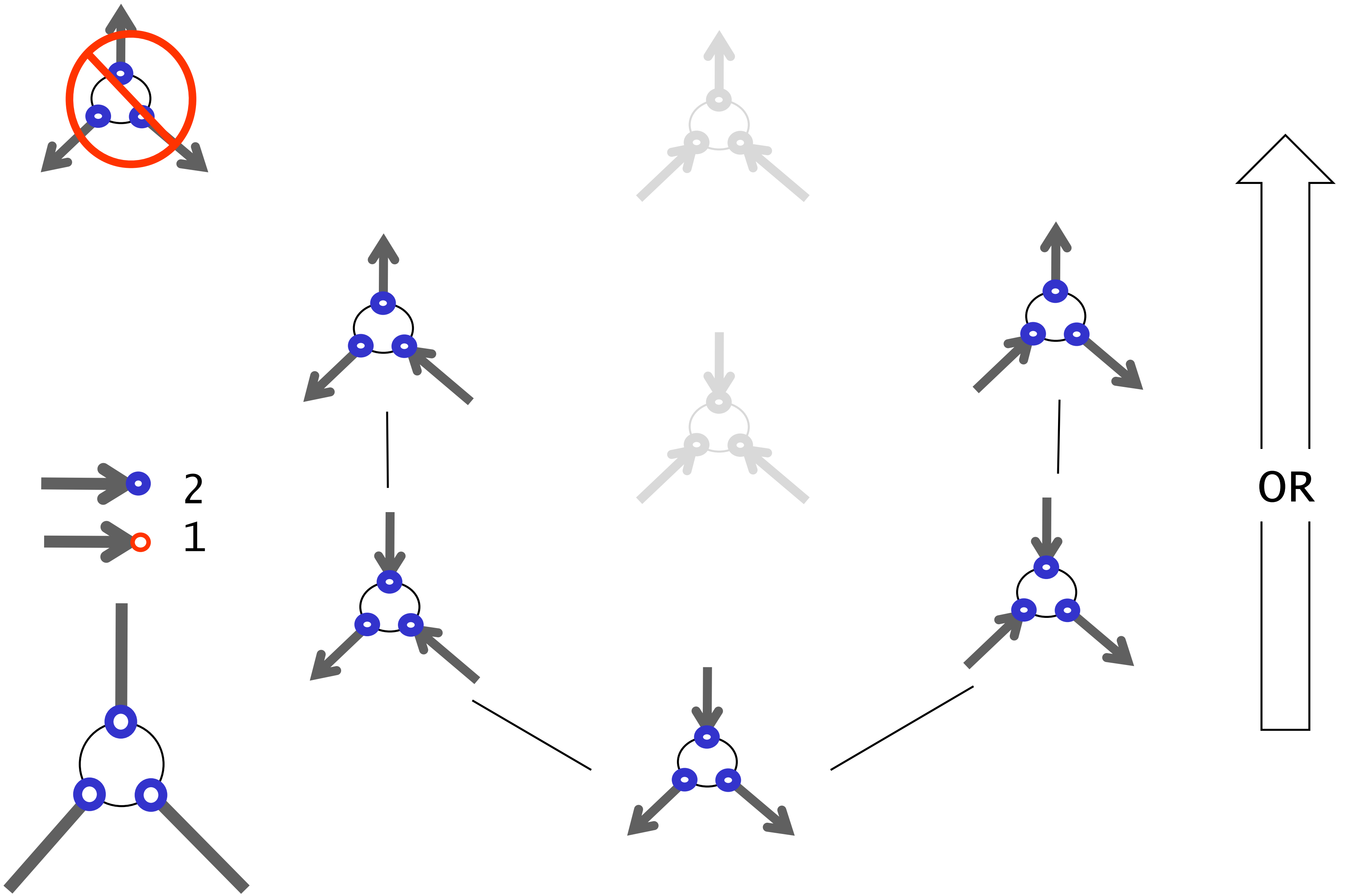


(b) Universal quantifier

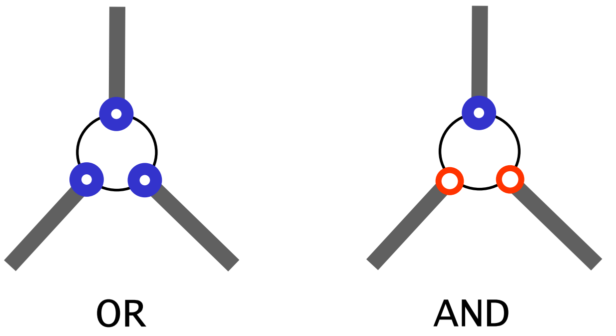
latch behaviour



protected OR



conclusion (NCL)



initial orientation
arrows not specified

NCL - nondet constraint logic
instance: constraint graph G, edge e
question: sequence which reverses e

thm. NCL is PSPACE-complete

next: concrete games

bounded: NP

unbounded: PSPACE

games and complexity

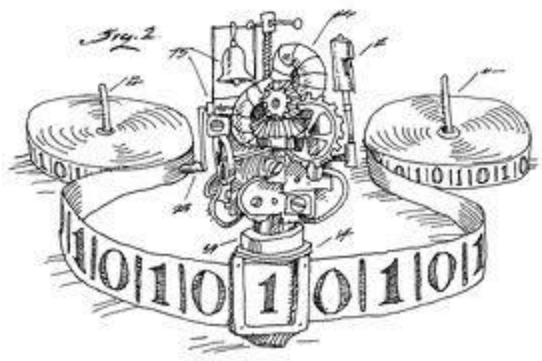
Met NCL en BNCL als tussenstap kunnen we concrete spellen NP-compleet dan wel PSPACE-compleet bewijzen.

Het volstaat om ‘gadgets’ te bouwen die de diverse knopen kunnen simuleren.

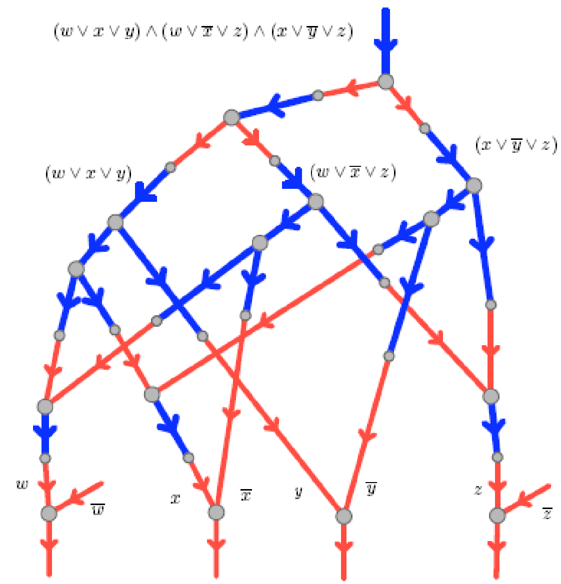
Probleem: de grafen van NCL laten kruisende takken toe. Dat is in spellen vaak niet toegestaan. Oplossing een gadget dat signalen laat kruisen.

Tipover is een bounded spel. Als een krat gevallen is blijft deze liggen
Rush-hour is unbounded. Auto’s mogen weer terug geschoven worden.

TipOver is NP-Complete



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



NP

3SAT

part I

constraint logic

'graph games'

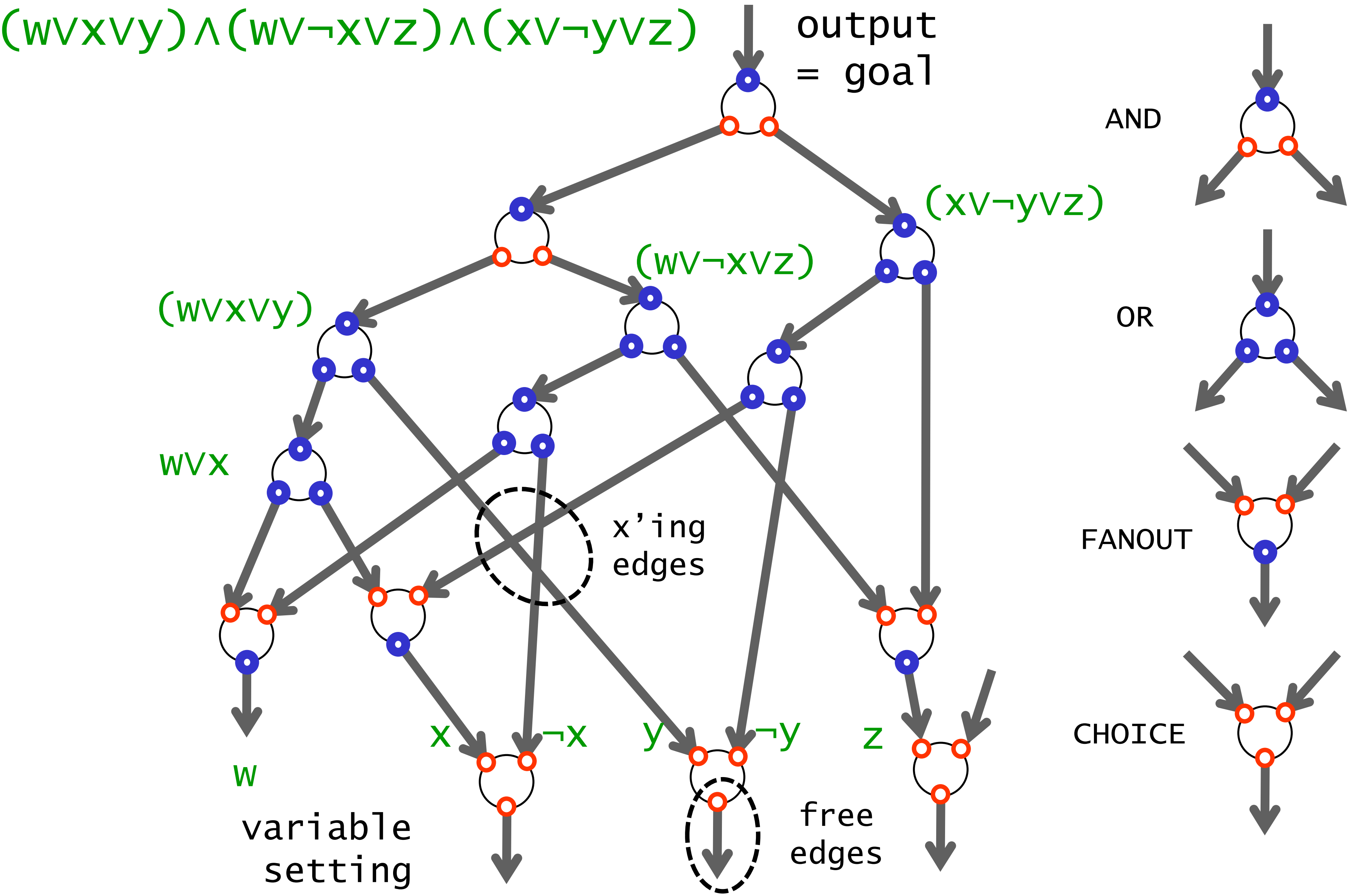
Bounded NCL

part II

games in particular

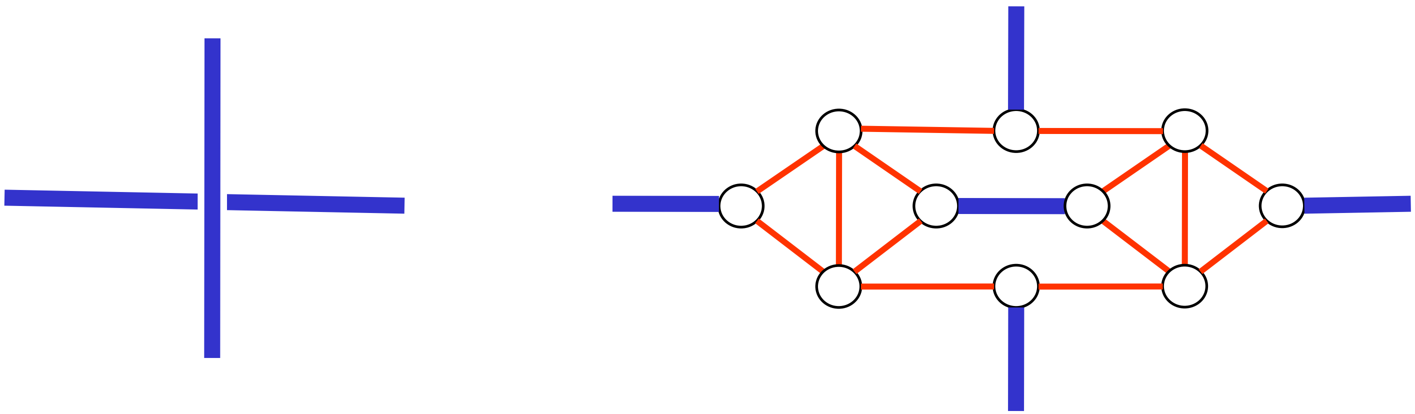
TipOver

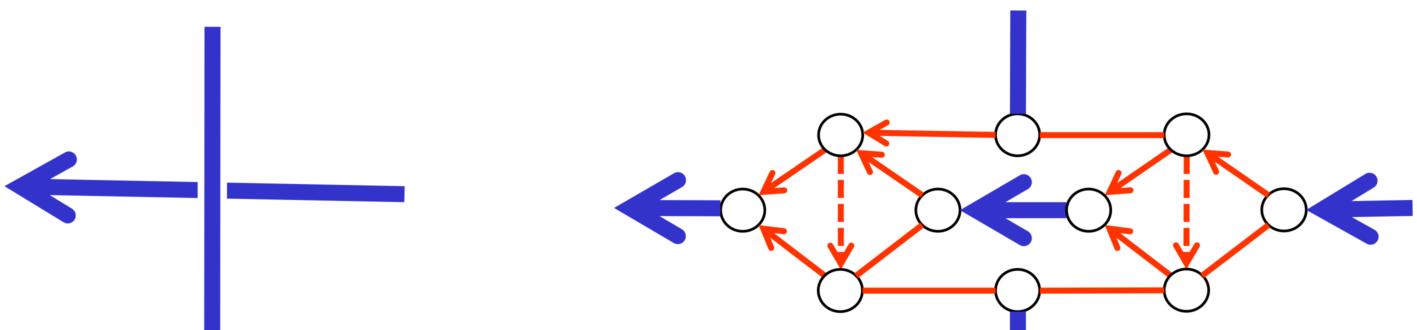
formula constraint graph



planar crossover gadget

formal proof Lemma 5.10

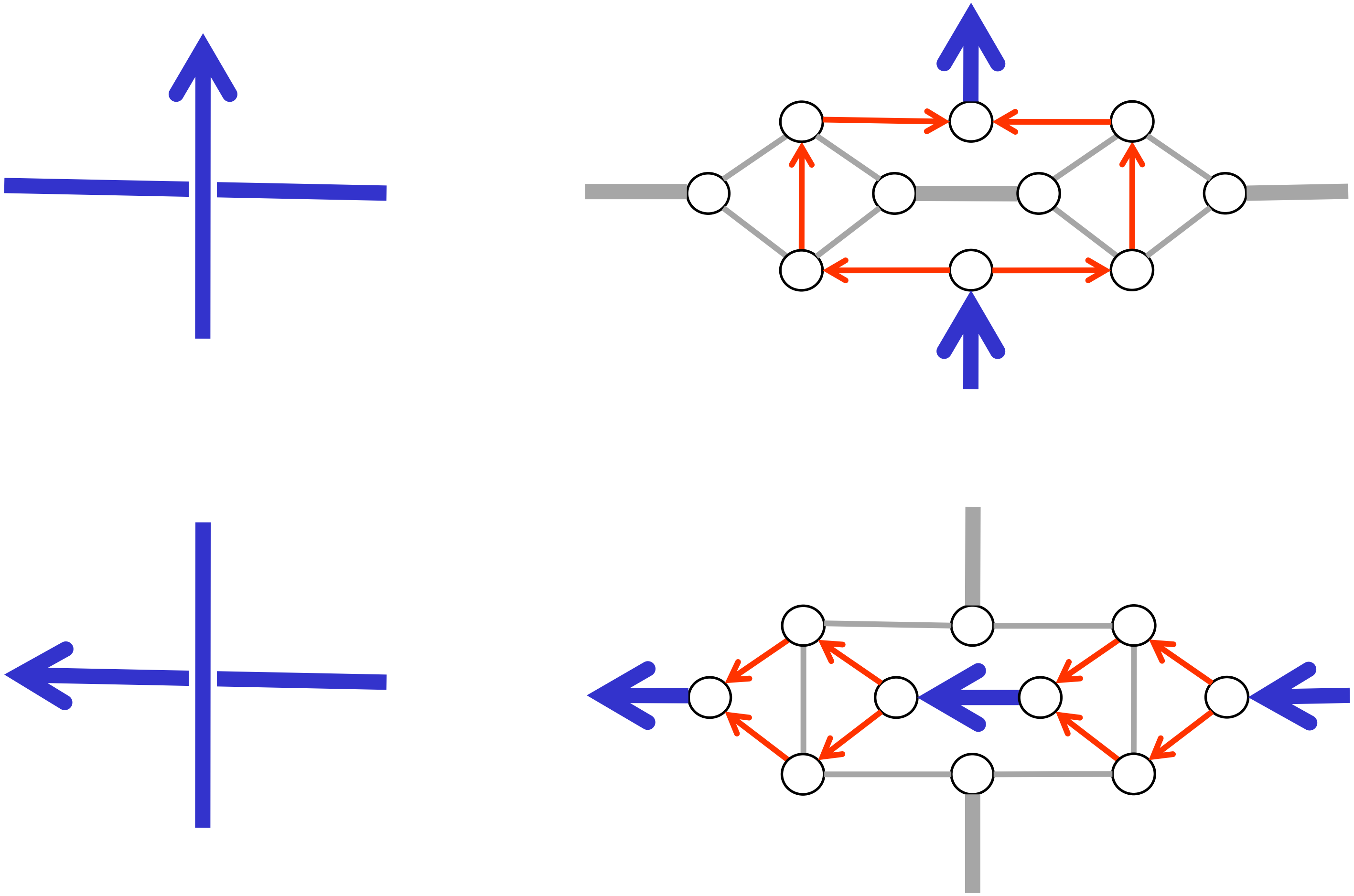




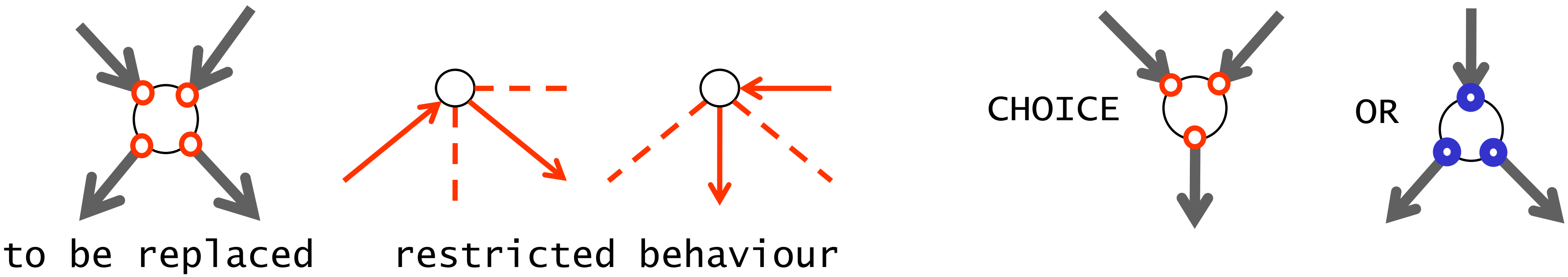
if leftward …

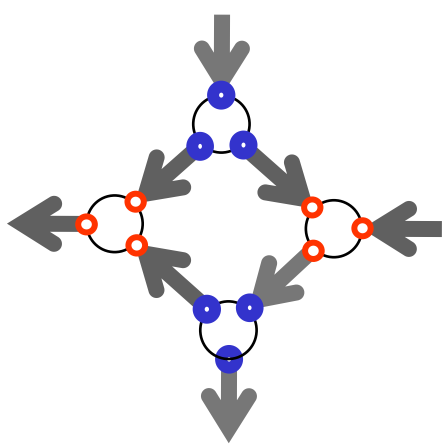
then leftward

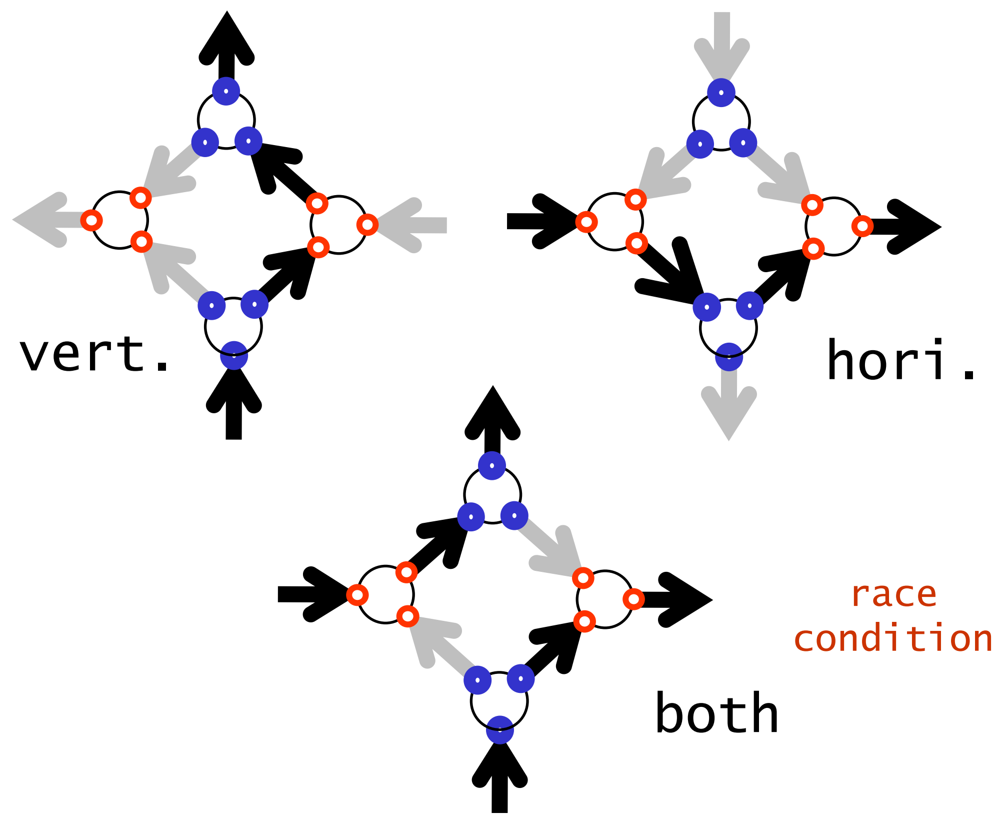
formal proof Lemma 5.10

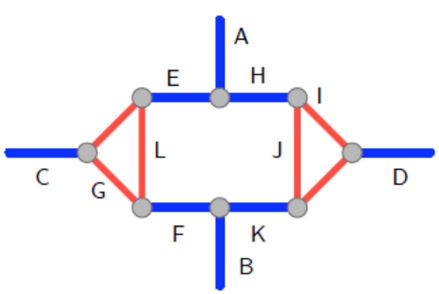


bounded NCL half-crossover





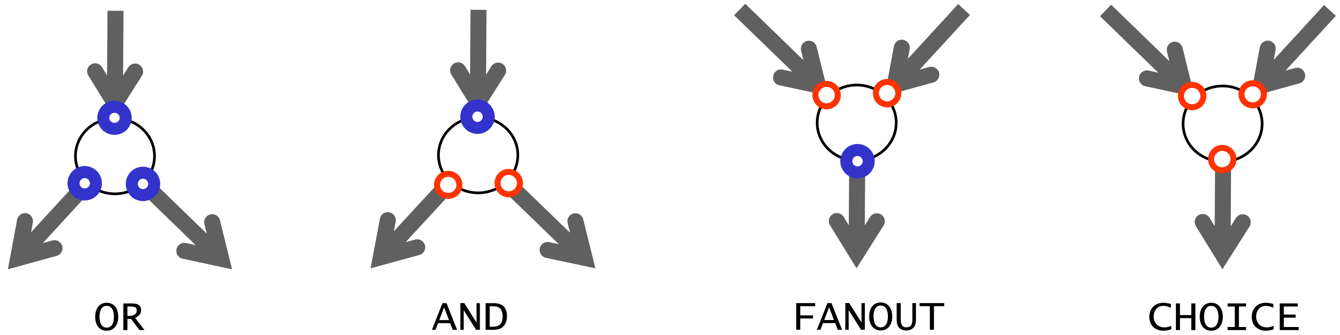




(b) Half-crossover

half-crossover *bounded* ncl type

conclusion (planar BNCL)



BOUNDED NCL - nondet constraint logic

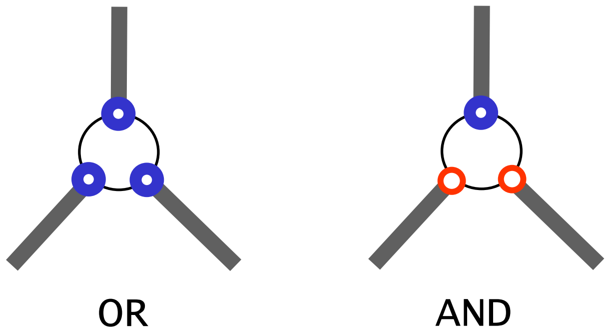
instance: constraint graph G, edge e

question: sequence which reverses *each edge at most once*, ending with e

Bounded NCL is NP-complete,

even for planar graphs,
with restricted vertices

conclusion (planar NCL)



NCL - nondet constraint logic

instance: constraint graph G, edge e

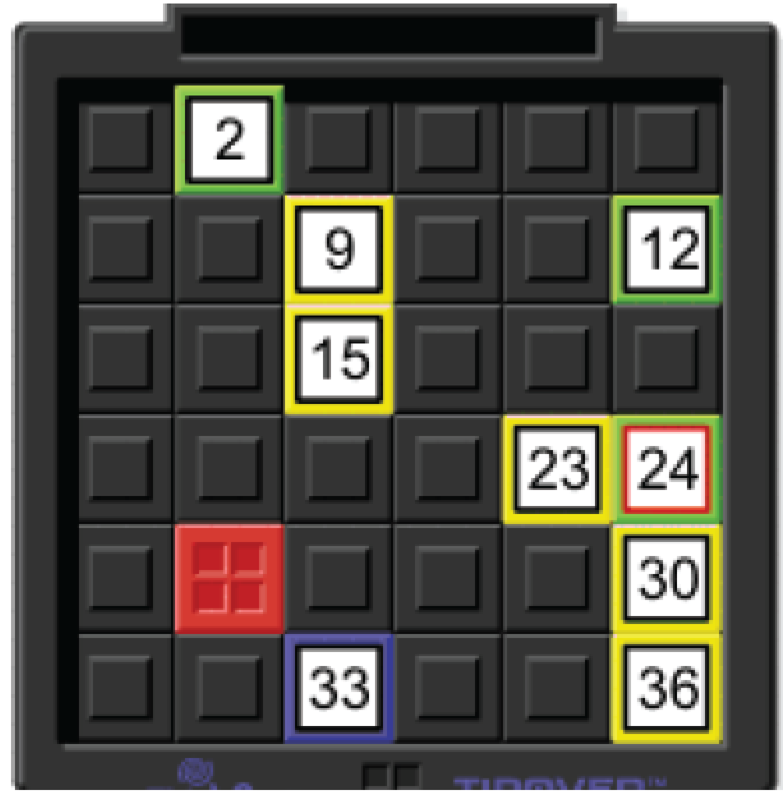
question: sequence which reverses e

NCL is PSPACE-complete,

even for planar graphs,

with restricted vertices

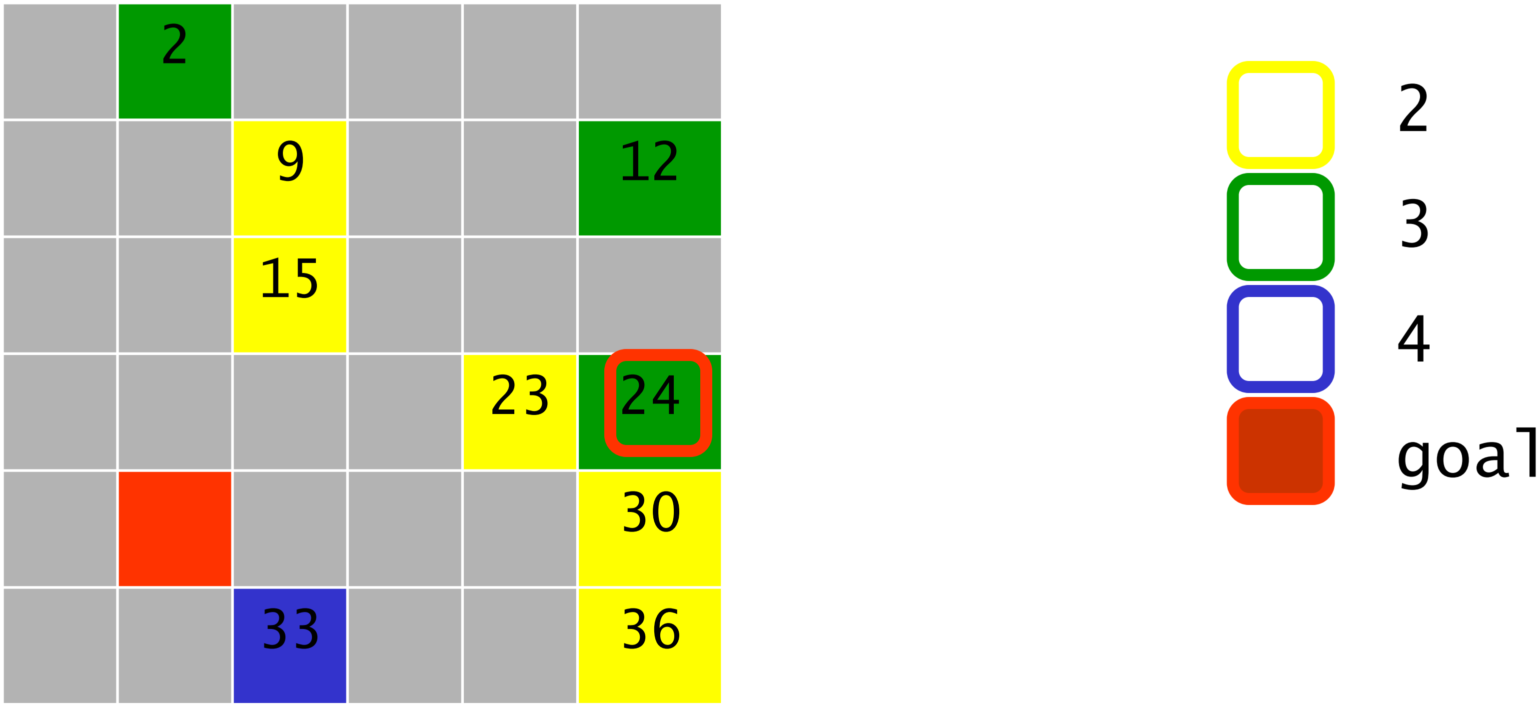
application: TipOver



- yellow: 2
- green: 3
- blue: 4
- red: goal

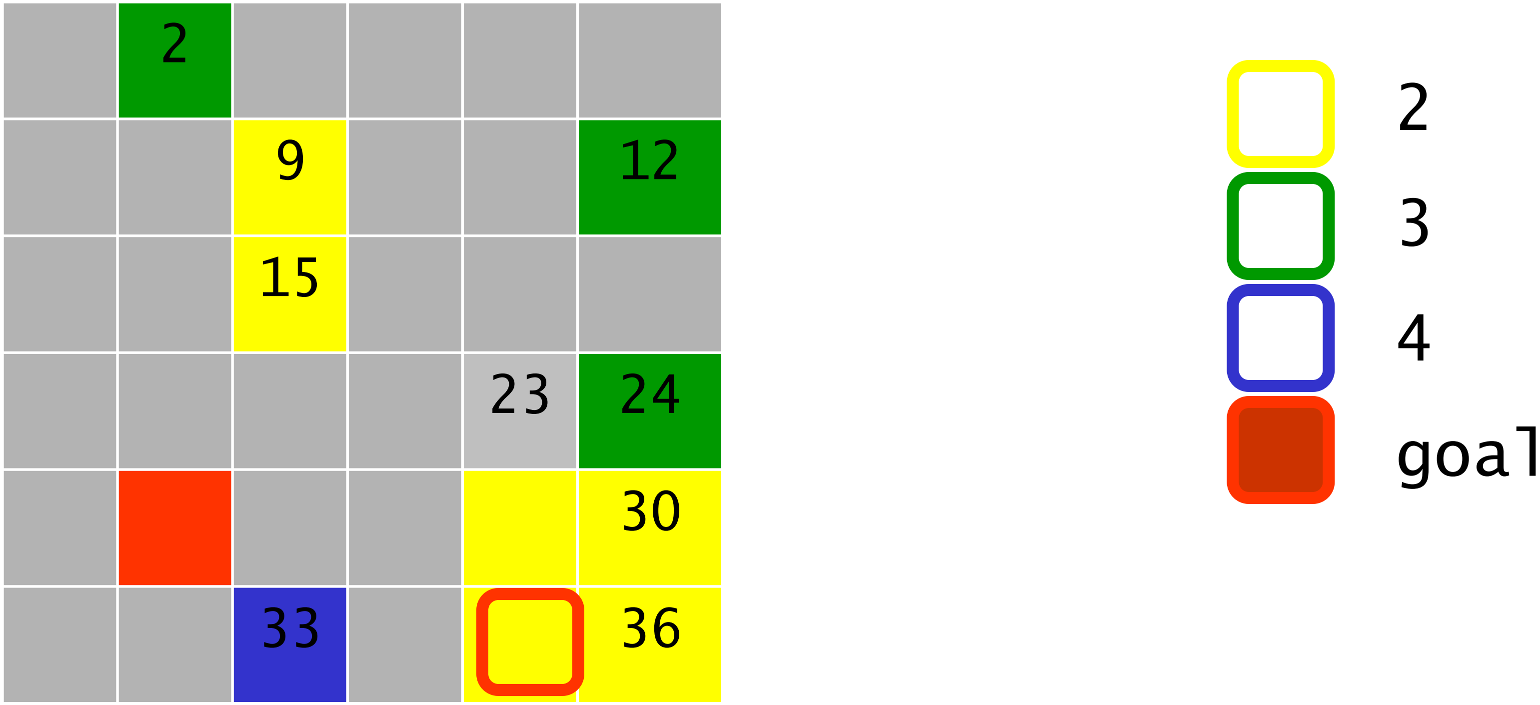
24 initial position

solution advanced



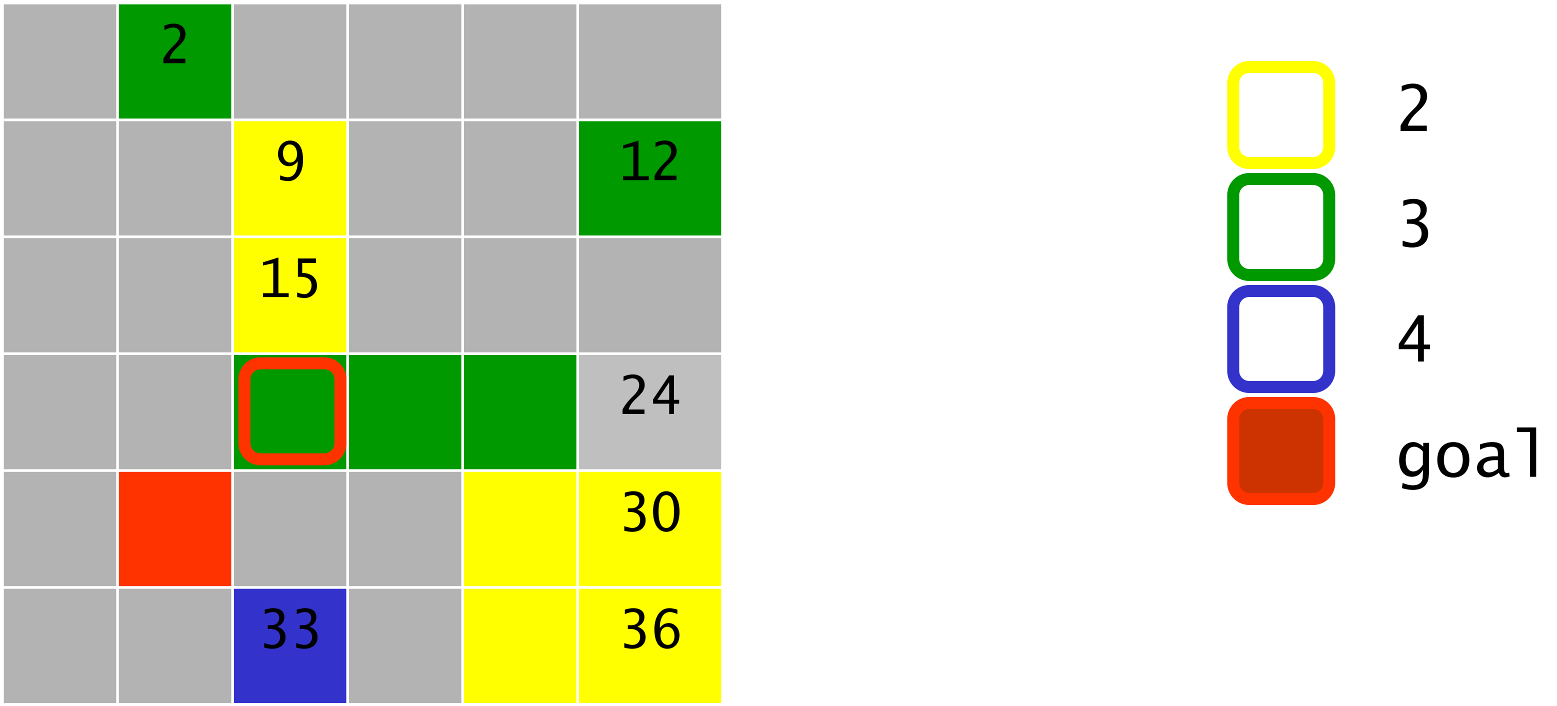
23D, 24L, 30U, 9R, 15U, 2D

solution advanced



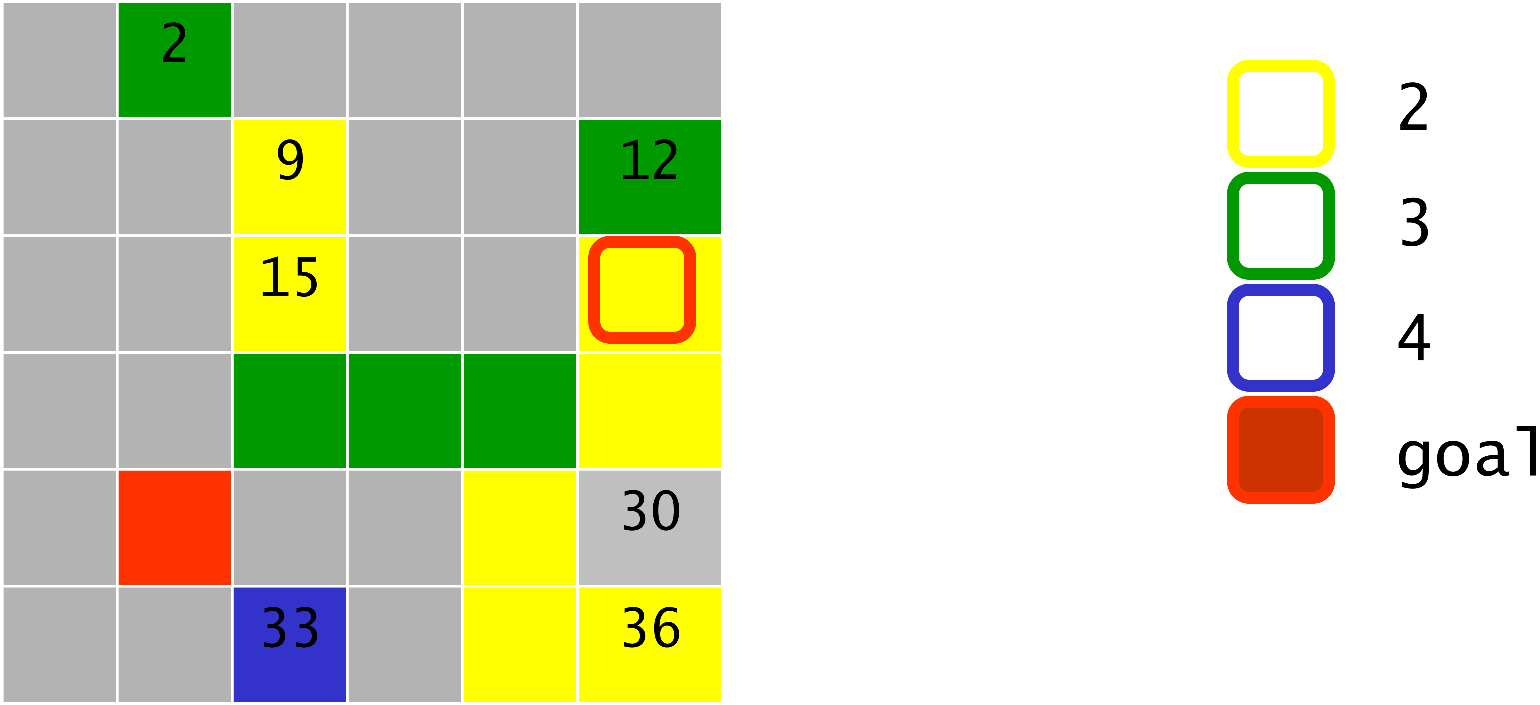
23D, 24L, 30U, 9R, 15U, 2D

solution advanced



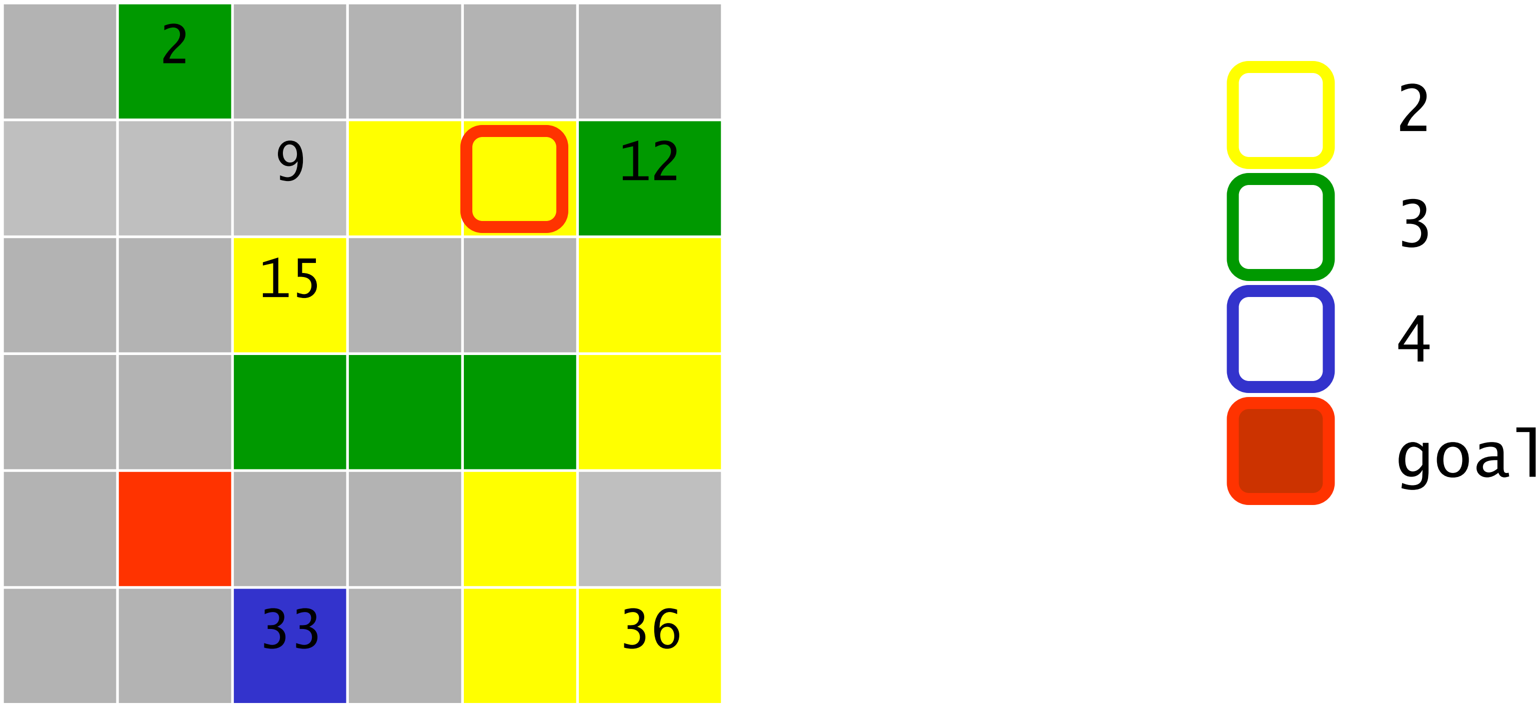
23D, 24L, 30U, 9R, 15U, 2D

solution advanced



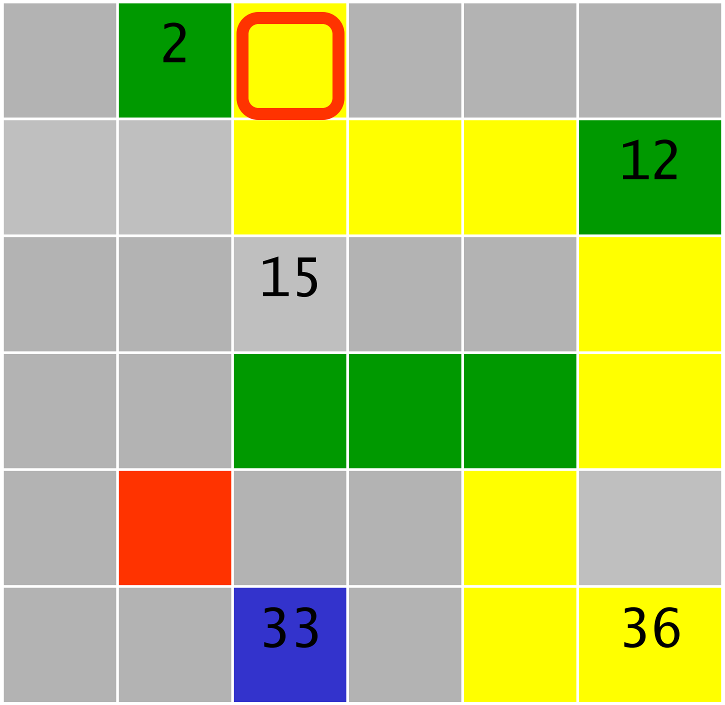
23D, 24L, 30U, 9R, 15U, 2D

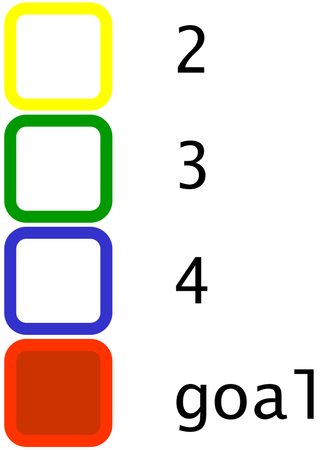
solution advanced



23D, 24L, 30U, 9R, 15U, 2D

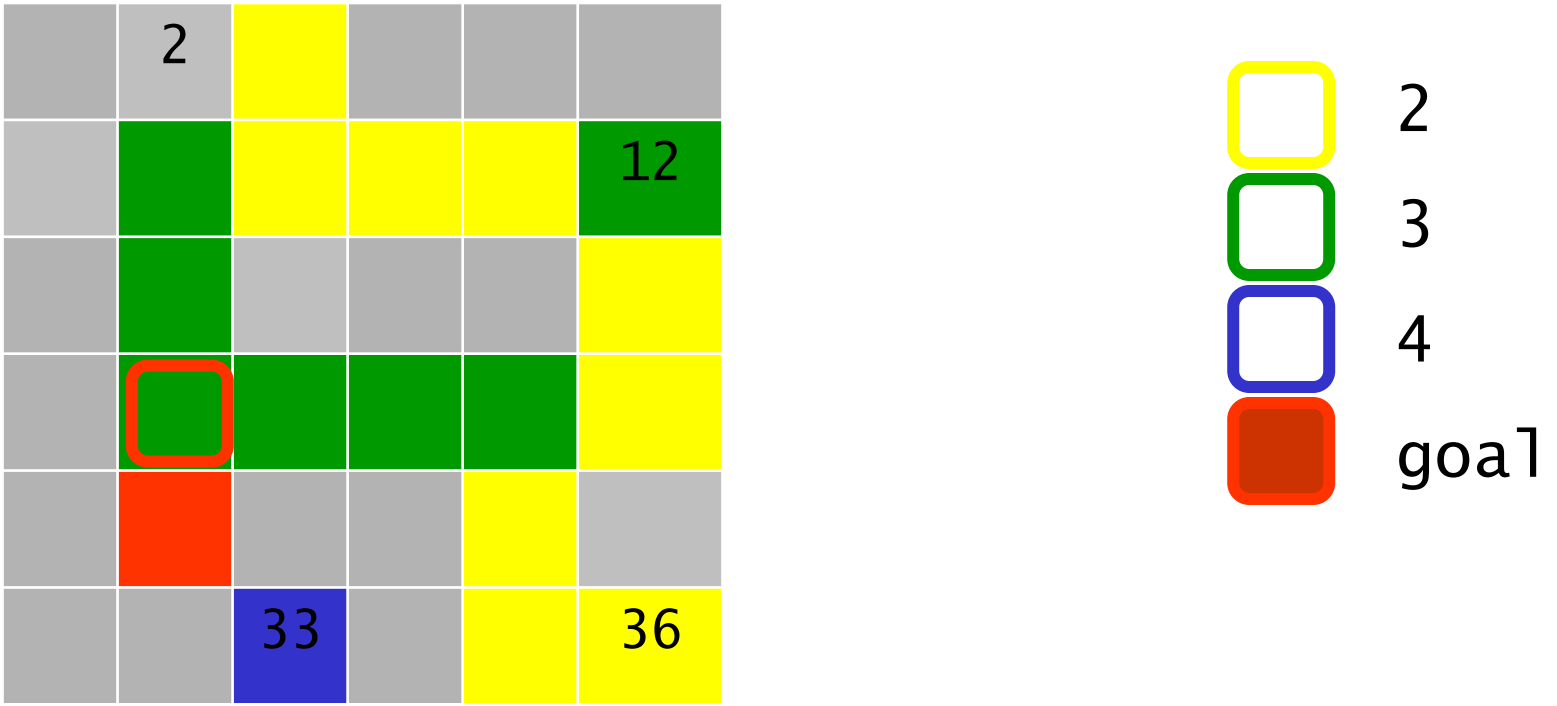
solution advanced





23D, 24L, 30U, 9R, 15U, 2D

solution advanced



23D, 24L, 30U, 9R, 15U, 2D

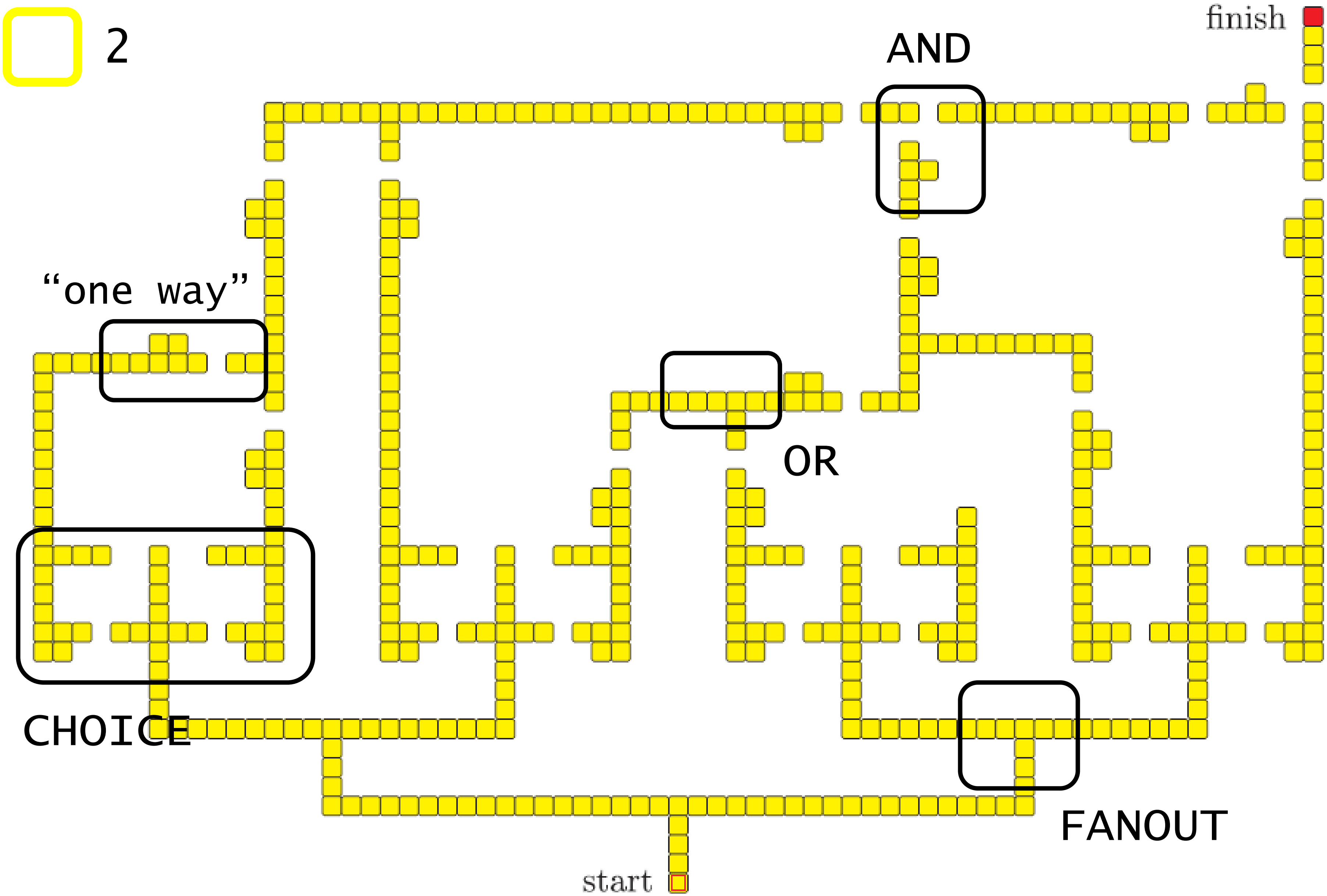


Figure 9-7: TipOver puzzle for a simple constraint graph.

gadgets: “one way”, OR

invariant:

- can be reached ⇔ can be inverted
- all visited positions remain connected

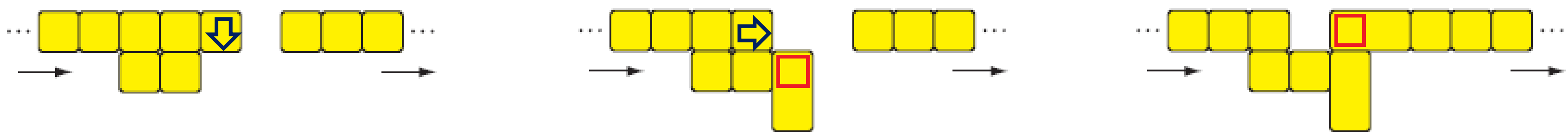
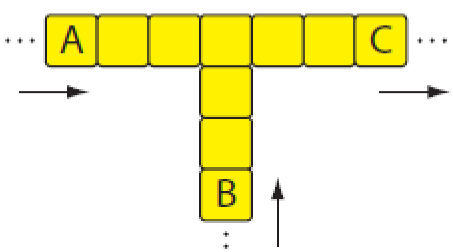
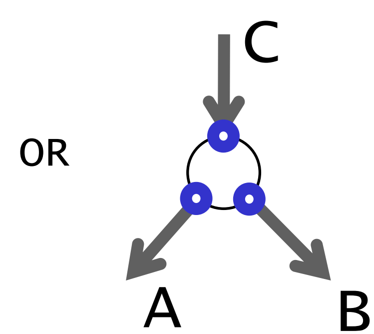


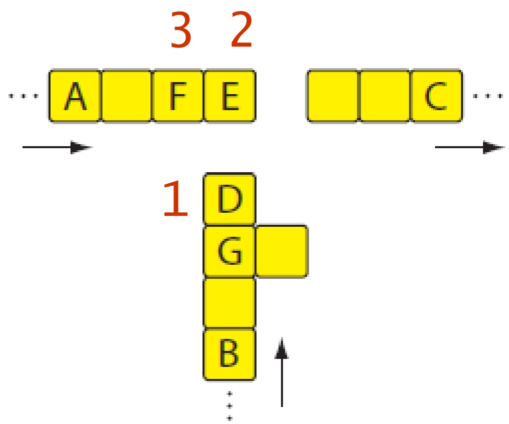
Figure 9-3: A wire that must be initially traversed from left to right. All crates are height two.



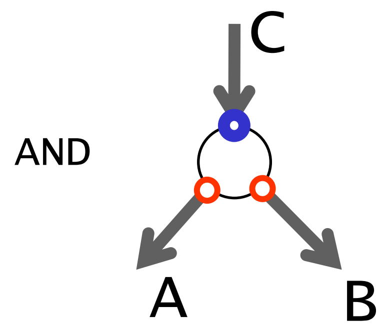
(a) OR gadget. If the tipper can reach either A or B, then it can reach C.



gadgets: AND



(b) AND gadget. If the tipper can reach both A and B, then it can reach C.



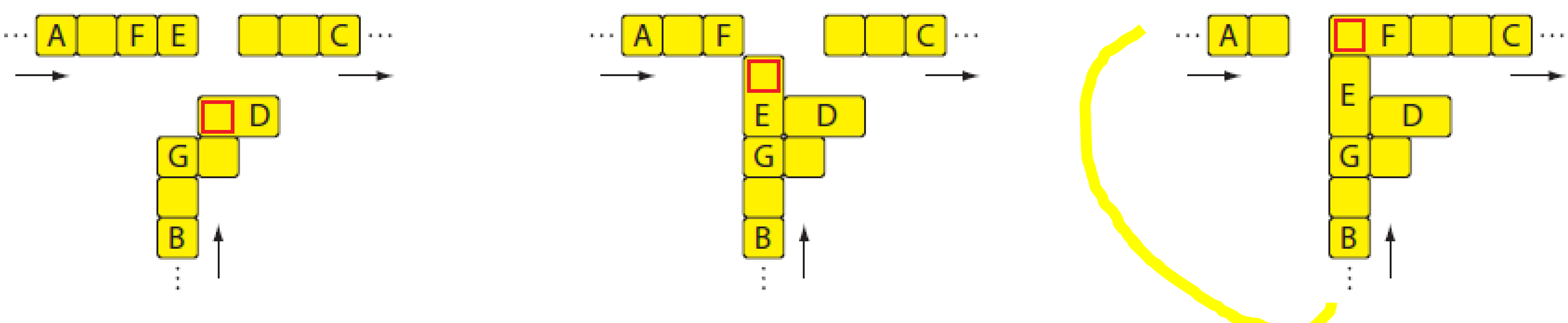
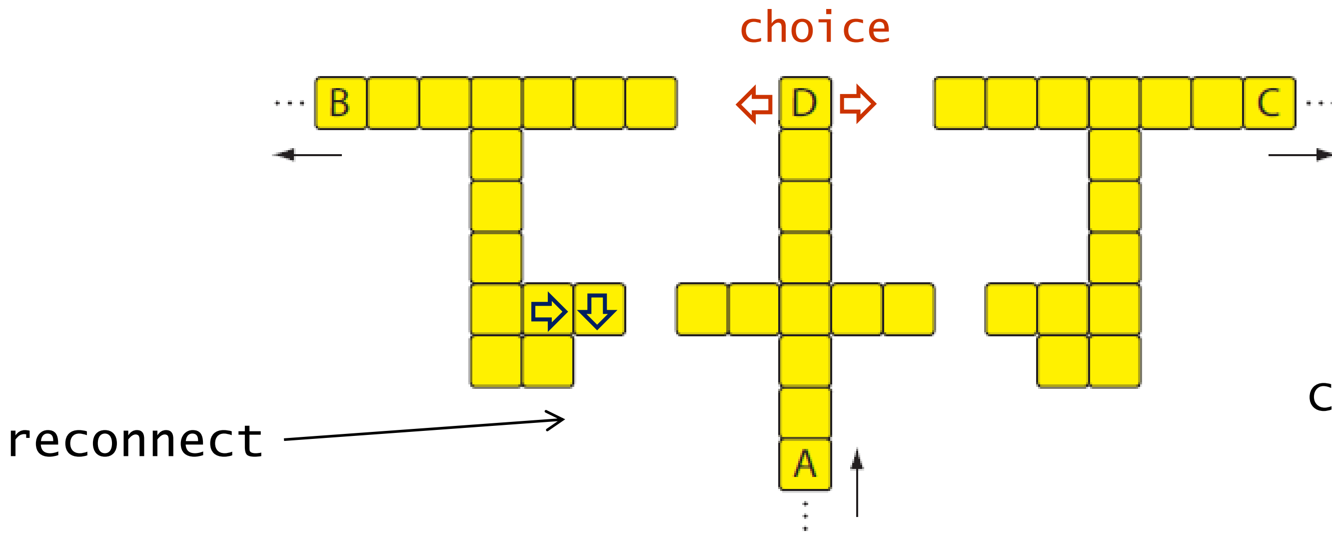


Figure 9-5: How to use the AND gadget.

remains connected

gadgets: CHOICE, FANOUT



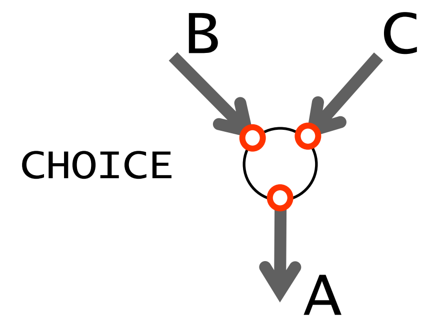
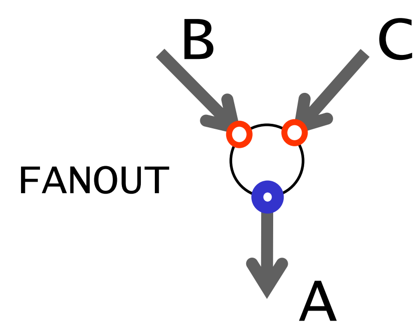


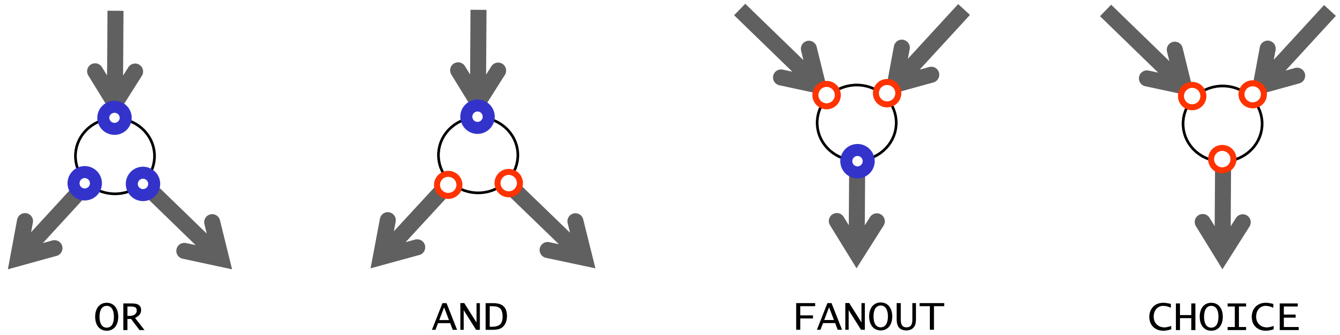
Figure 9-6: TipOver CHOICE gadget. If the tipper can reach A, then it can reach B or C, but not both.



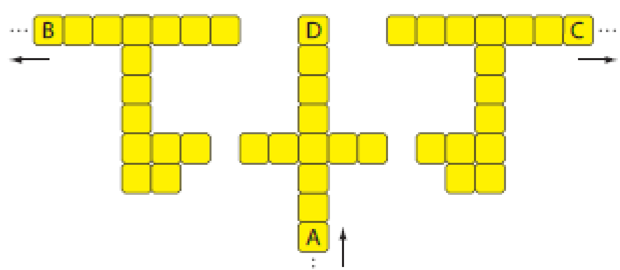
use one-way gadgets at B and C

(control information flow)

conclusion



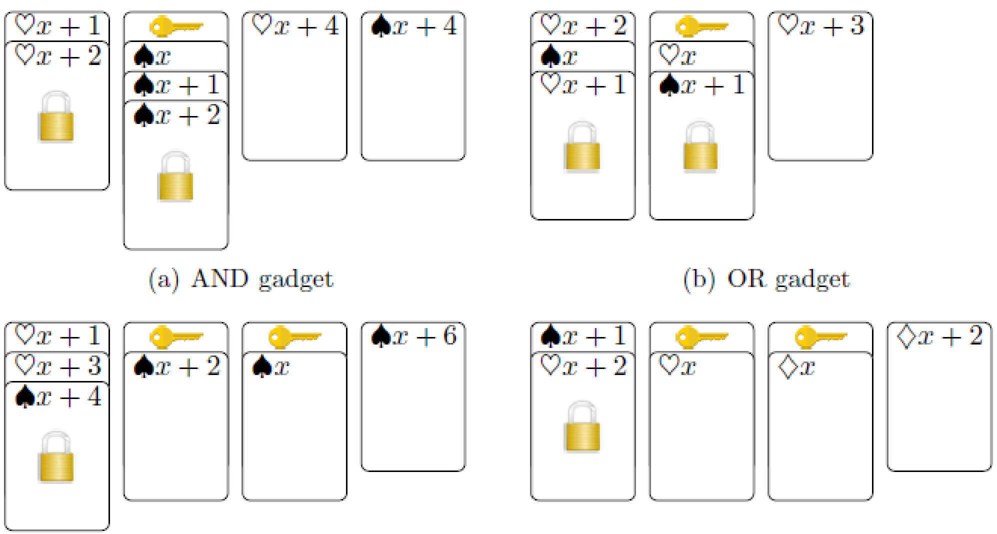
Bounded NCL is NP-complete,
even for planar graphs,
with above restricted vertices



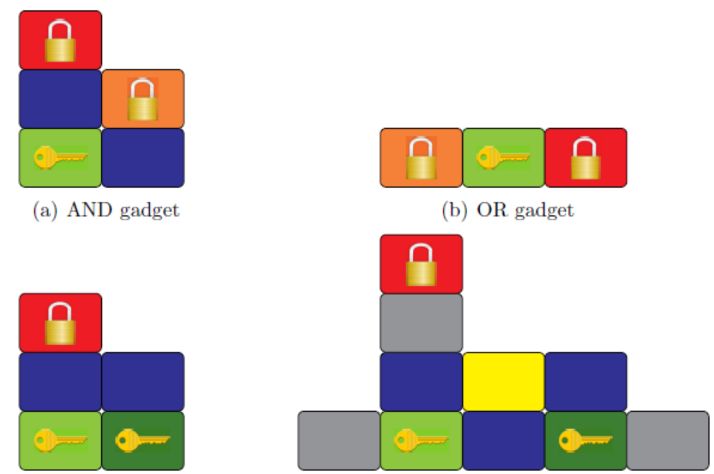
thm. TipOver is NP-complete

NP complete bounded games

Jan van Rijn: Playing Games:
The complexity of Klondike, Mahjong, Nonograms
and Animal Chess
(Master Thesis, 2013, Leiden)

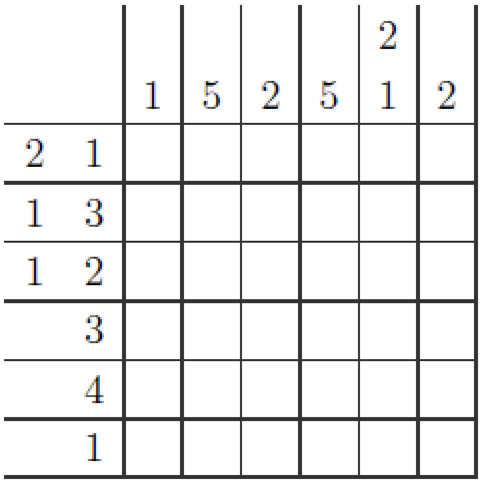


(a) AND gadget (b) OR gadget (c) FANOUT gadget (d) CHOICE gadget

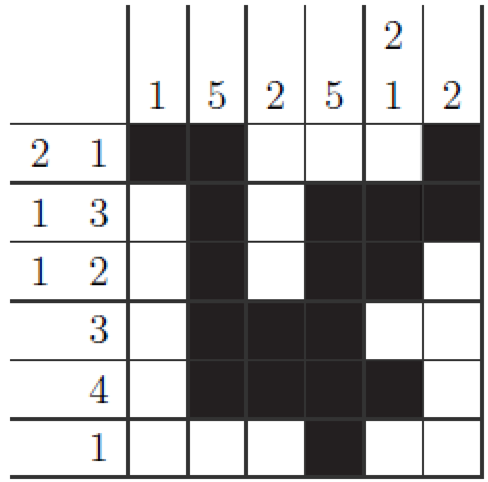


(a) AND gadget (b) OR gadget (c) FANOUT gadget (d) CHOICE gadget

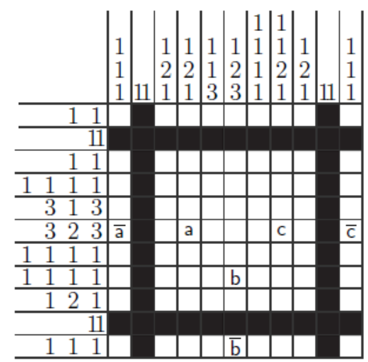
(ctd.) nonograms



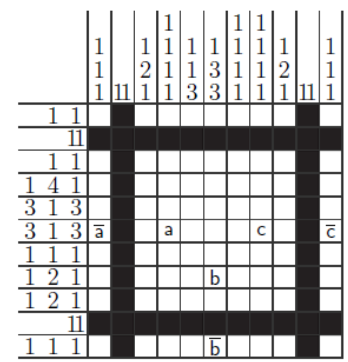
(a) 6 × 6 Nonogram



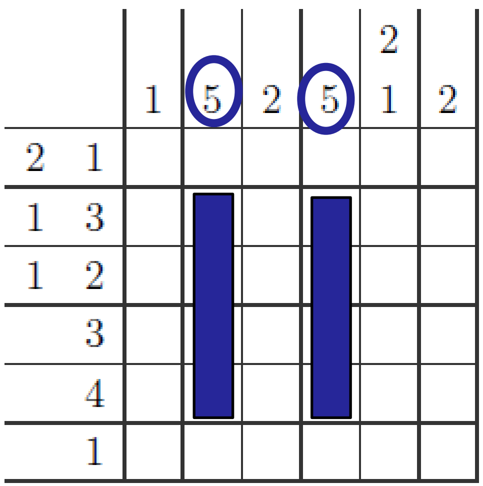
(b) Solved Nonogram

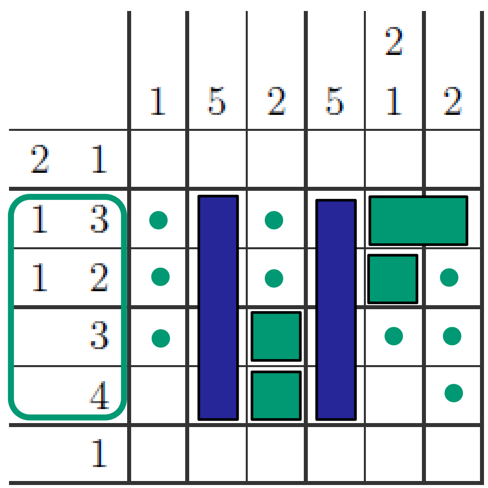


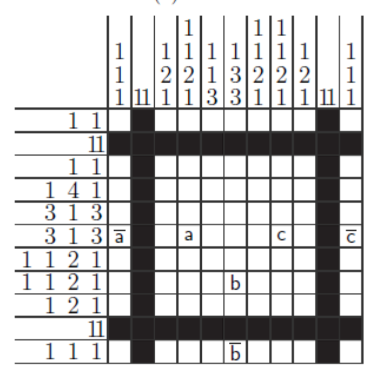
(a) AND



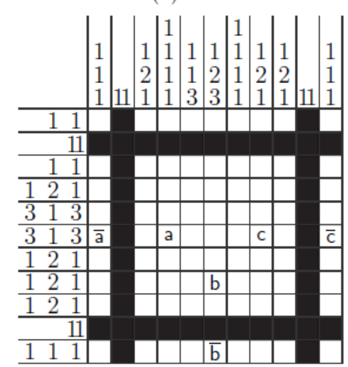
(b) OR







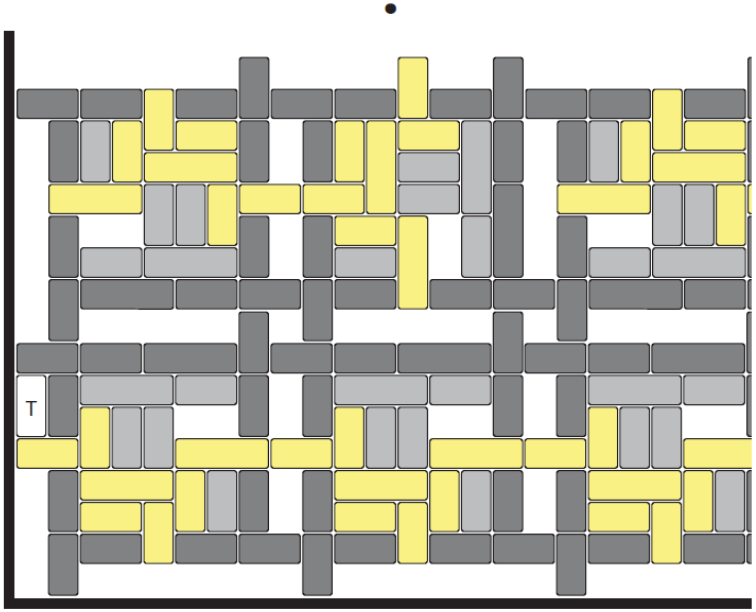
(c) FANOUT

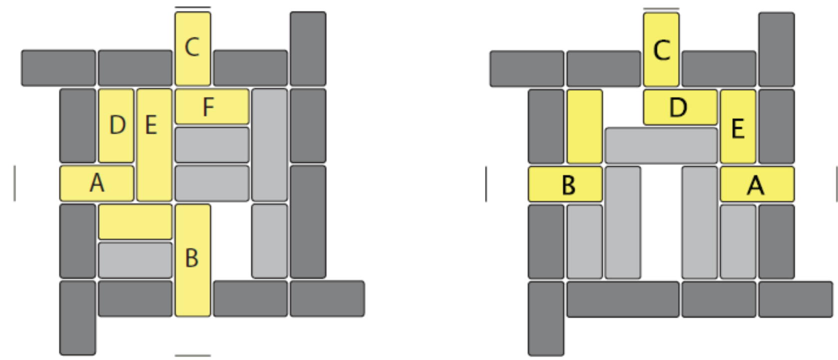


(d) CHOICE

Rush Hour and Plank puzzle are PSPACE-complete

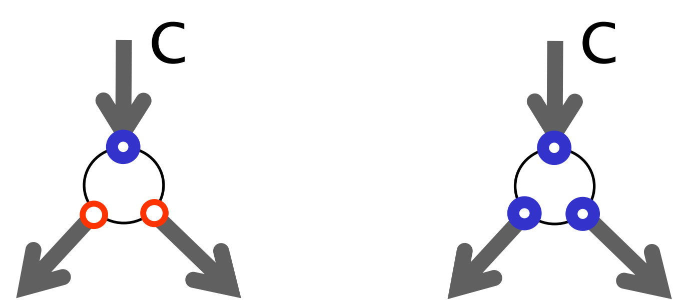
rush hour





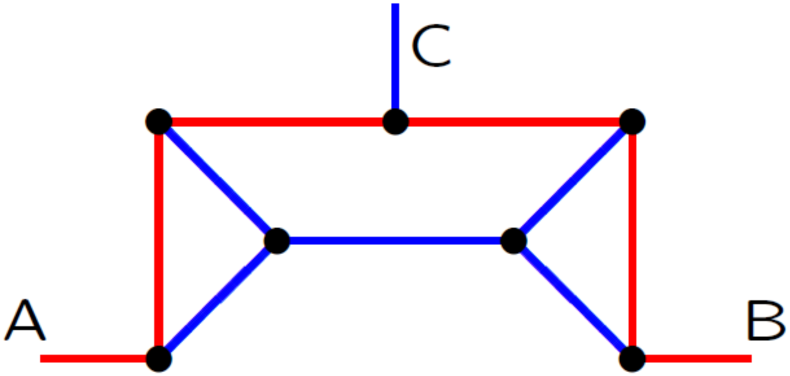
(b) AND (c) Protected OR

car in ⇔ edge out

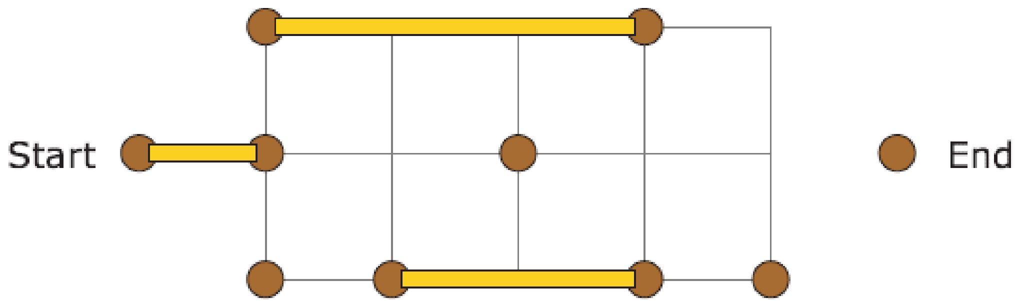


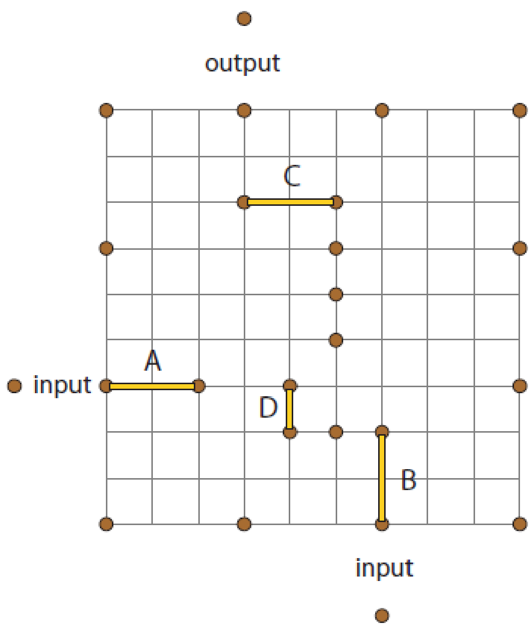
AND OR

latch / protected OR

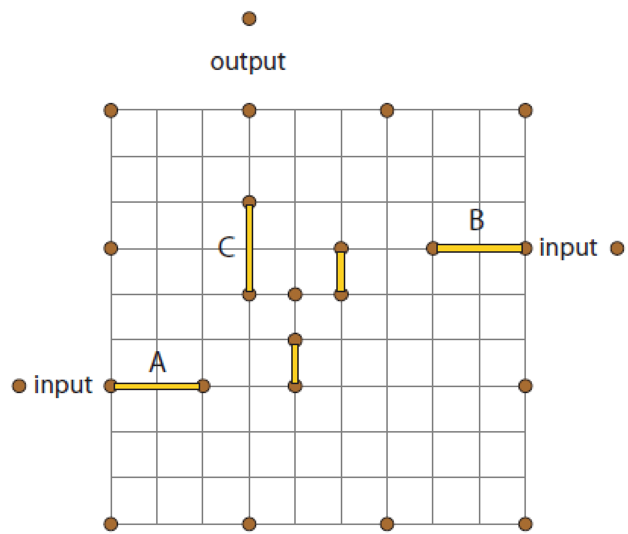


plank puzzle

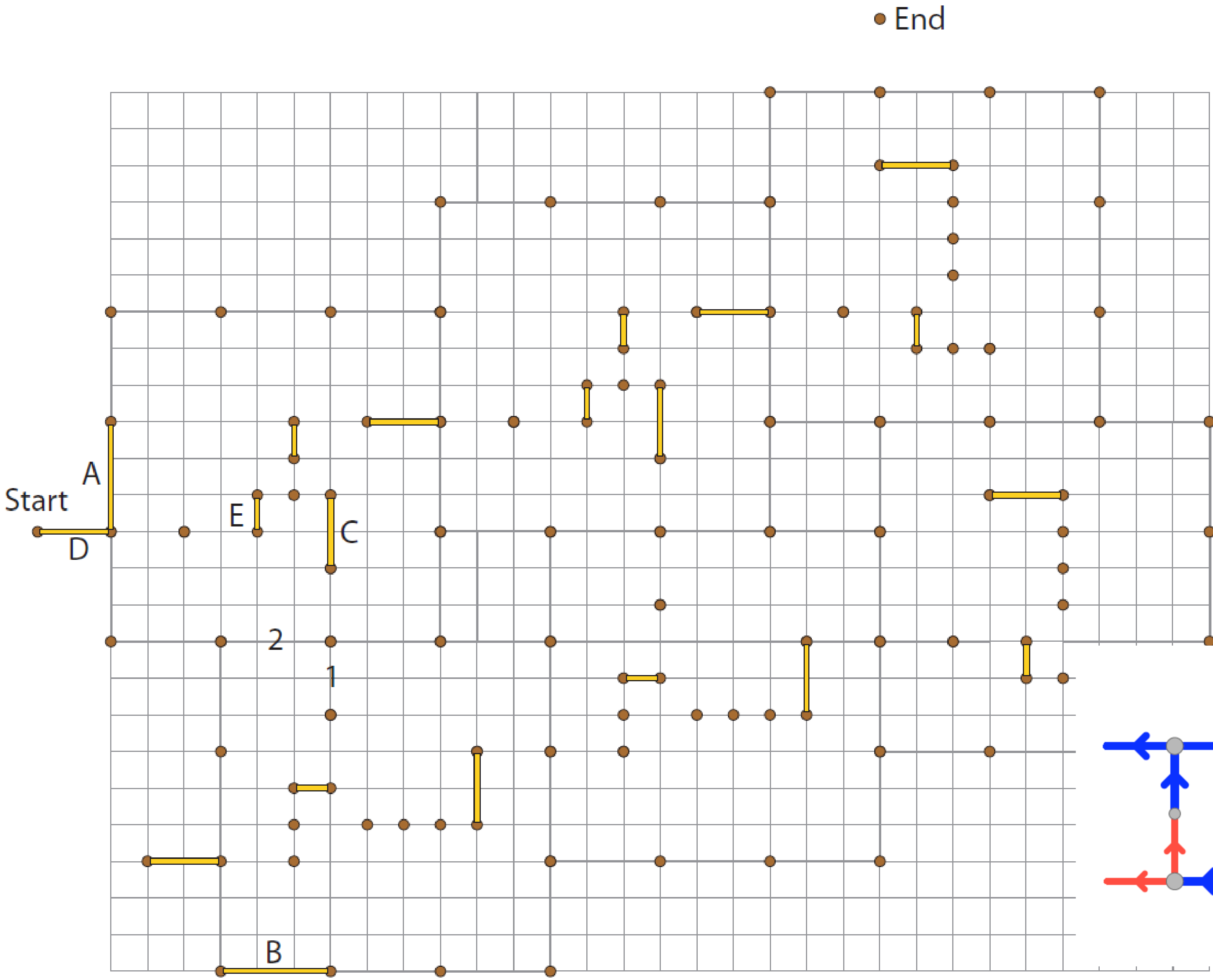


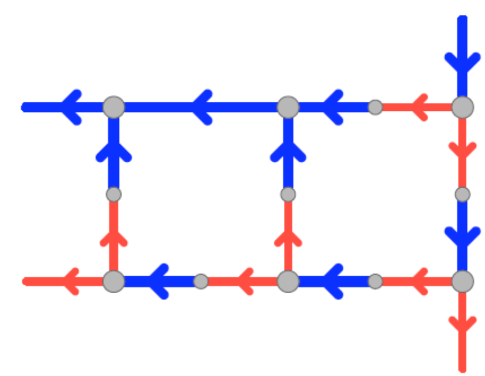


(a) AND



(b) OR


End
Start
A
D
E
C
2
1
B



Wrap Up

conclusion

conclusion: nice uniform family of graph games, suitable for the various game classes

not in this presentation:

deterministic classes are hard to prove complete: timing constraints

bounded det. ncl has no known planar normal form

2pers. games need two types of edges (apart from colours), for each of the players

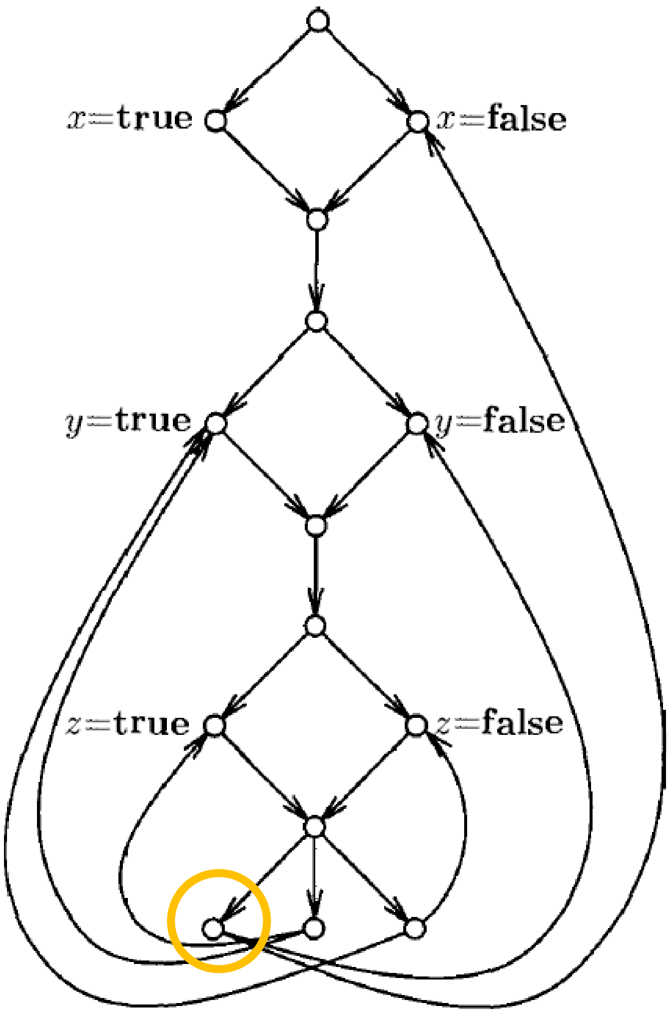
for teams one needs hidden info, otherwise equivalent to 2p games

roots can be found in the literature
(see Geography)

take care: game of life (what is the 'goal'?)
is PSPACE, it also is undecidable ☺
(on infinite grid)

example of P-complete:
the domino topling simulation

geography



[Schäfer, J.CSS, 1978]

THM. Geography is PSPACE-complete

two players on directed graph alternately pick next vertex, without repetition

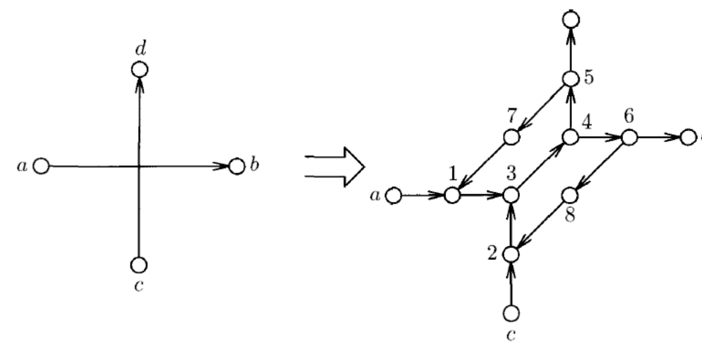


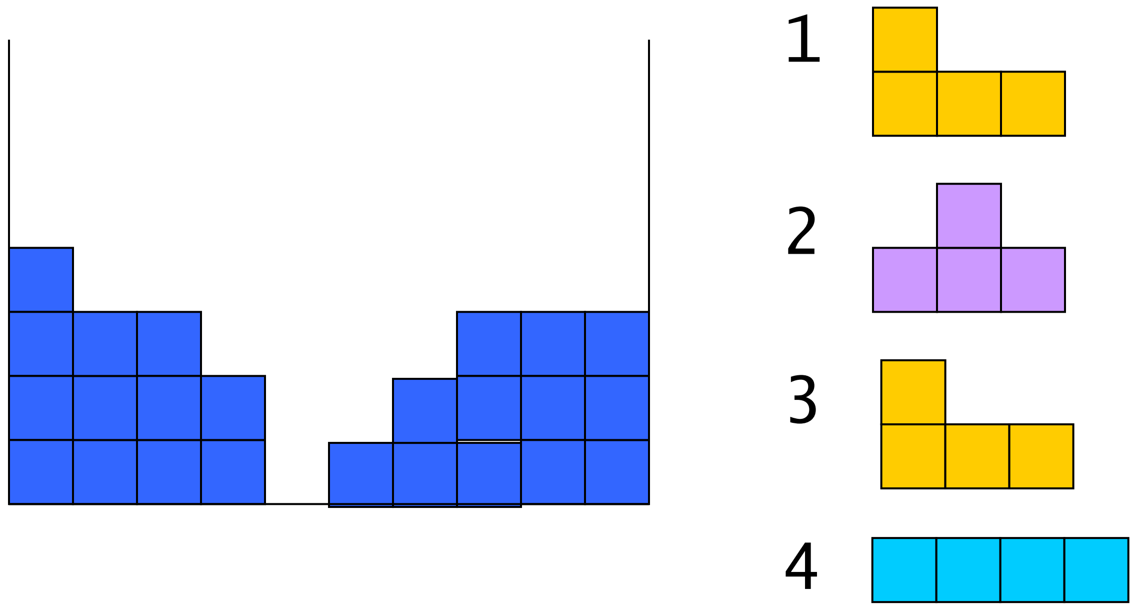
Figure 19-4. Crossing edges in GEOGRAPHY.

application to GO
[Lichtenstein&Sipser, J.ACM, 1980]

$$\exists x \forall y \exists z [(\neg x \vee \neg y) \wedge (y \vee z) \wedge (y \vee \neg z)].$$

Tetris is NP complete

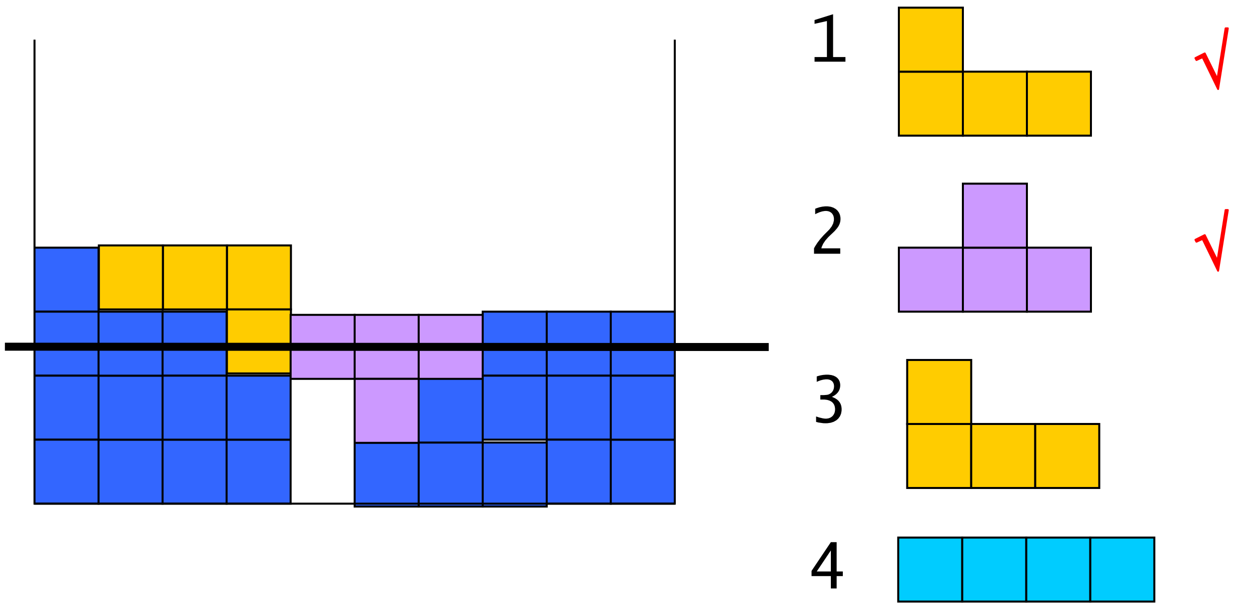
"Given an initial game board and a sequence of pieces, can the board be cleared?"



Breukelaar, Demaine, Hohenberger, Hoogeboom, Kosters, Liben-Nowell. Tetris is Hard, Even to Approximate. Selected Papers from the Ninth Int. Computing and Combinatorics Conf. (COCOON 2003). *Int. J. of Computational Geometry and Applications* 14 (2004) 41-68.

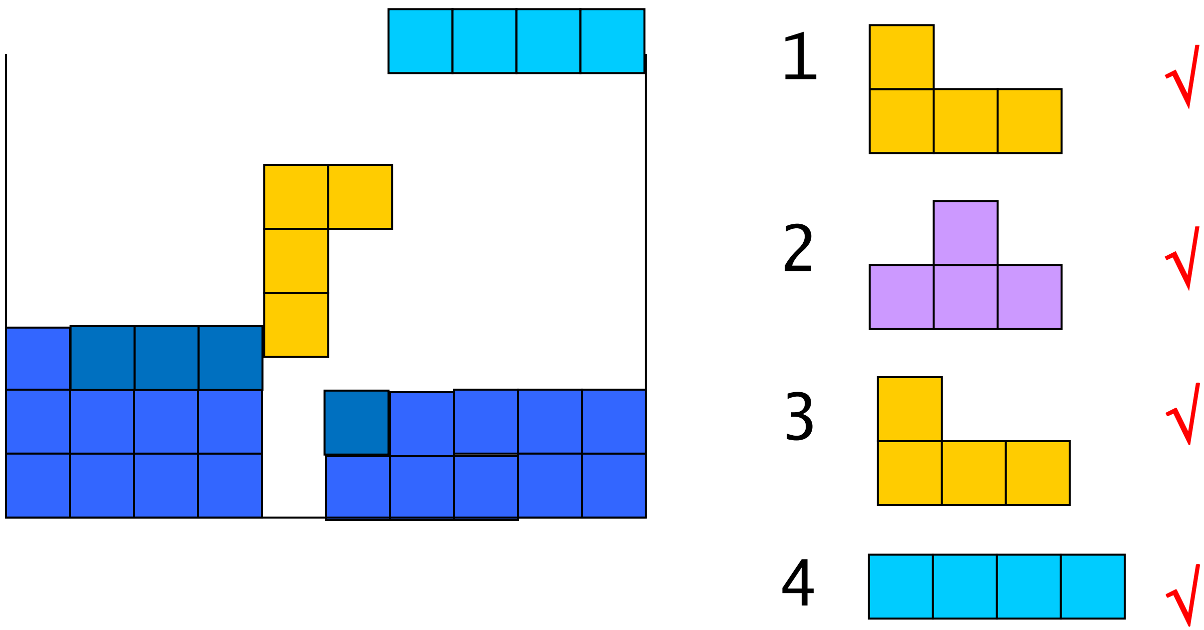
Tetris is NP complete

"Given an initial game board and a sequence of pieces, can the board be cleared?"



Tetris is NP complete

"Given an initial game board and a sequence of pieces, can the board be cleared?"



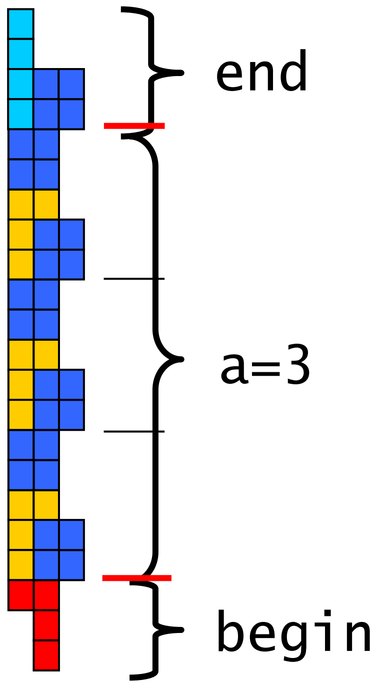
yes!

Tetris is NP complete

reduction from 3-partitioning problem
(can we divide set of numbers into triples?)

OPEN: directly with Bounded NCL ?

find OR, AND, FANOUT, CHOICE



Game Complexity

IPA Advanced Course on Algorithmics and Complexity

Eindhoven, 25 Jan 2019

Walter Kosters
Hendrik Jan Hoogeboom

LIACS, Universiteit Leiden

Cook, Stephen (1971). "The complexity of theorem proving procedures". *Proceedings of the Third Annual ACM Symposium on Theory of Computing*. pp. 151–158. http://portal.acm.org/citation.cfm?coll=GUIDE&dl=GUIDE&id=805047.

Provably difficult combinatorial games, L. J. Stockmeyer and A. K. Chandra, *SIAM J. Computing* 8 (1979), pp. 151-174.

N.D. Jones, W.T. Laaser, Complete Problems for Deterministic Polynomial Time, TCS 3: 105-117, 1977.

Chandra, Ashok K.; Kozen, Dexter C.; Stockmeyer, Larry J. (1981). "Alternation". Journal of the ACM. 28 (1): 114-133