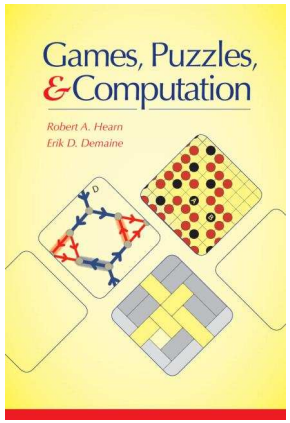
Game Complexity

Gadgets

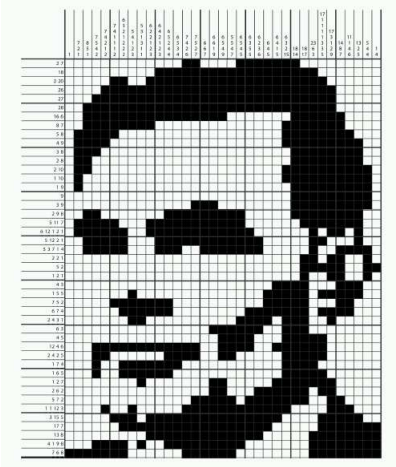


Walter Kosters, Universiteit Leiden

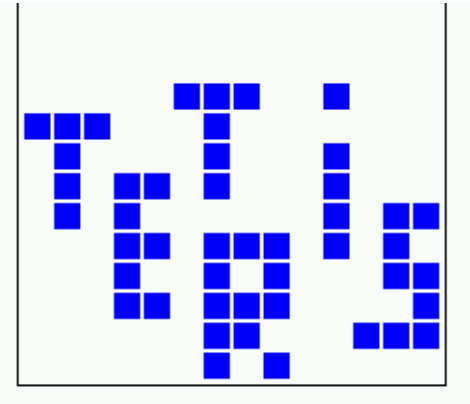
www.liacs.leidenuniv.nl/~kosterswa/

IPA, Eindhoven; Friday, July 8, 2016

link



link

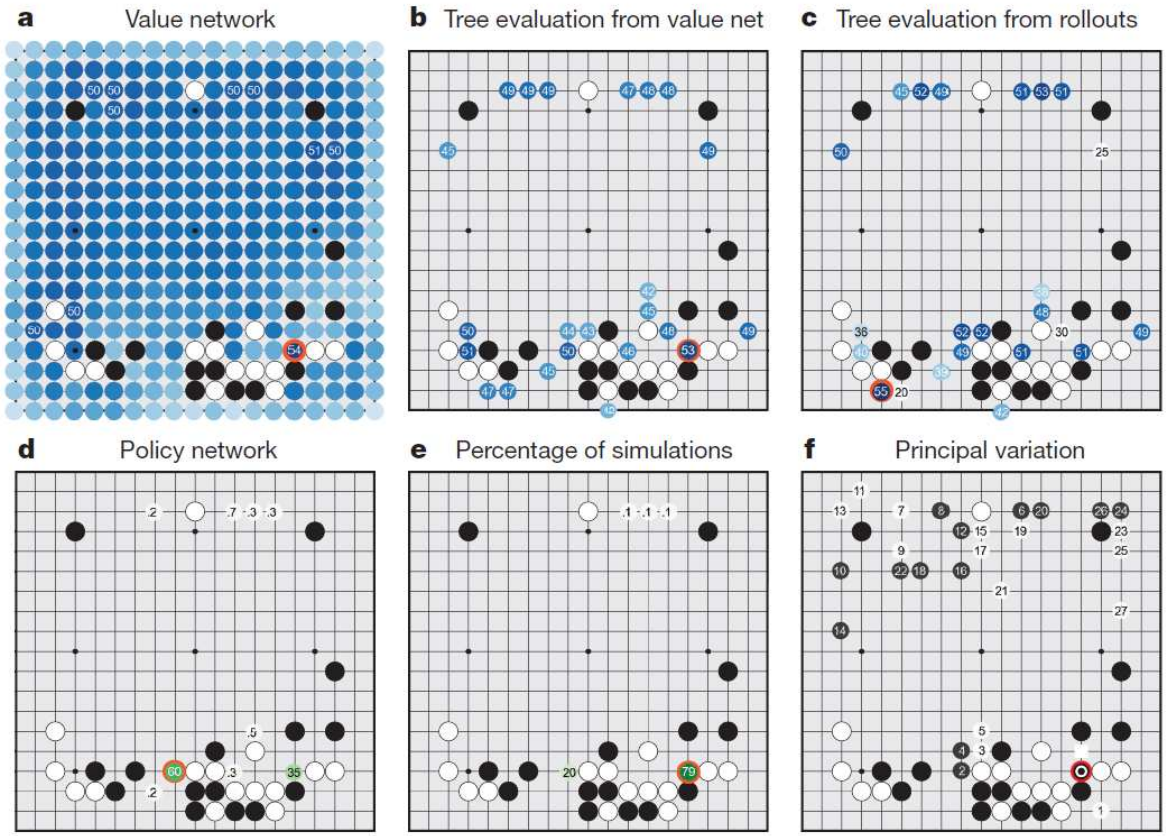


link

crime novels,

tomography

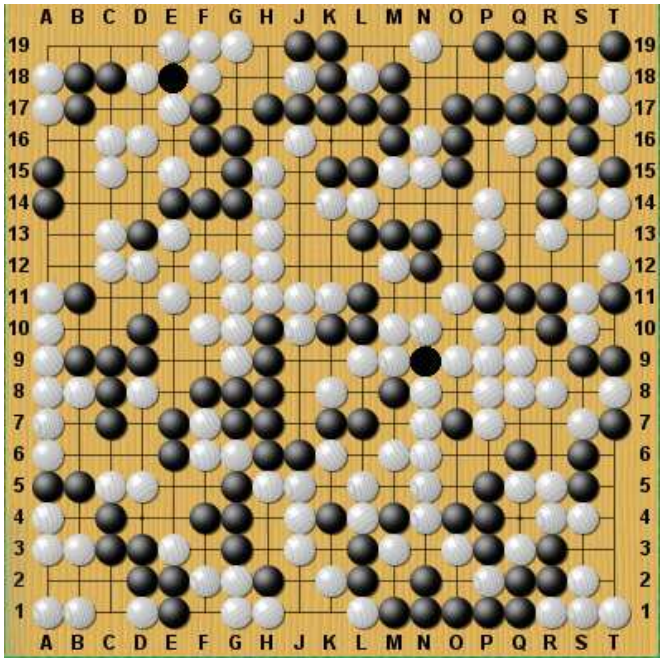
and Tetris



3.2016: AlphaGo(-ogle) defeats top Go player Lee Sedol

In 2016 John Tromp showed at CG2016 that there are

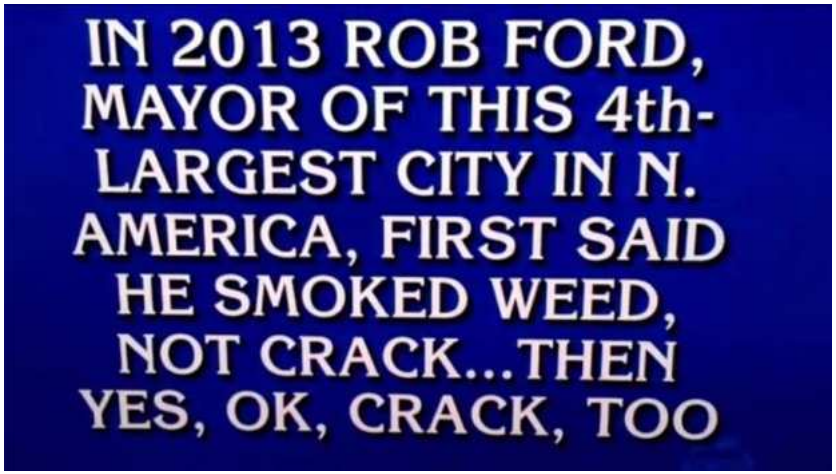
208168199381979846
994786333448627028
652245388453054842 5
639456820927419612 7
380153785256484516 9
851964390725991601 5
628128546089888314 4
271297153193175577 3
662039724706484093 5



$\approx 2 \cdot 10^{170}$ legal positions in 19×19 Go, using DP & CPUs.

https://tromp.github.io/go/legal.html

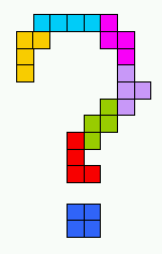
Jeopardy!



2011

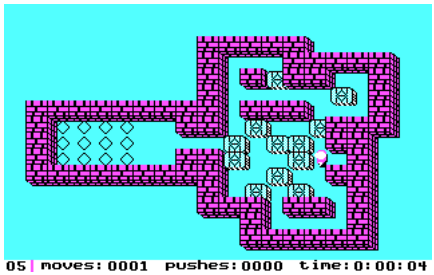
We study the complexity of games (puzzles, . . .). We want to make statements like

Tetris is NP-complete.



In order to do so, we examine reductions between appropriate games, with the help of gadgets.

Games studied include TipOver,
Plank puzzles, Sokoban→,
Rush Hour, Mahjongg, . . .



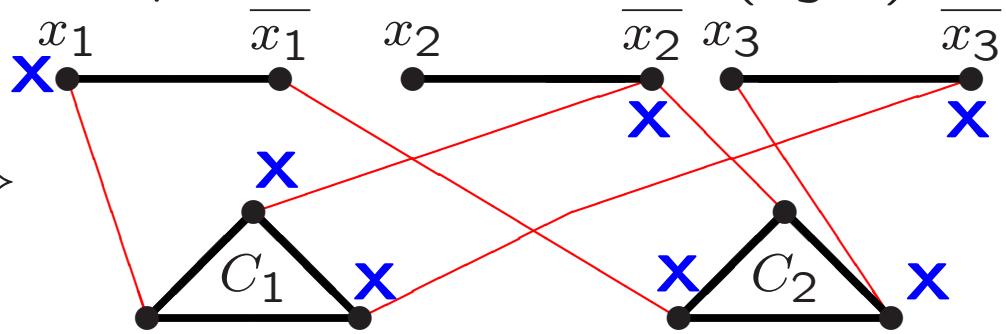
We want to reduce a known problem to a new one, for example, 3SAT to VC (so **V**ertex**C**over is NP-hard).

For every Boolean variable x_i we make a variable gadget (left) and for every clause C_j a clause gadget (right):

We connect these gadgets in the intuitive way; now satisfying assignments (left) correspond to vertex covers (right):

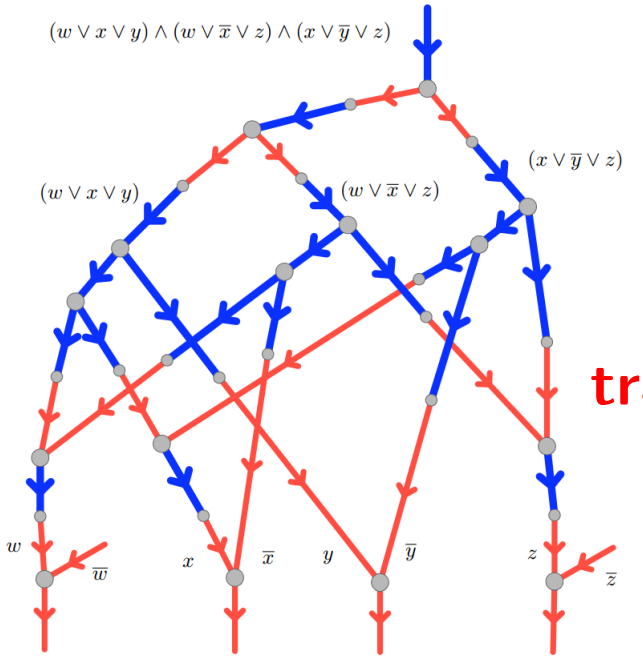
$$\overbrace{(x_1 \vee \overline{x_2} \vee \overline{x_3})}^{C_1} \wedge \underbrace{(\overline{x_1} \vee \overline{x_2} \vee x_3)}_{C_2}$$

$\Rightarrow$

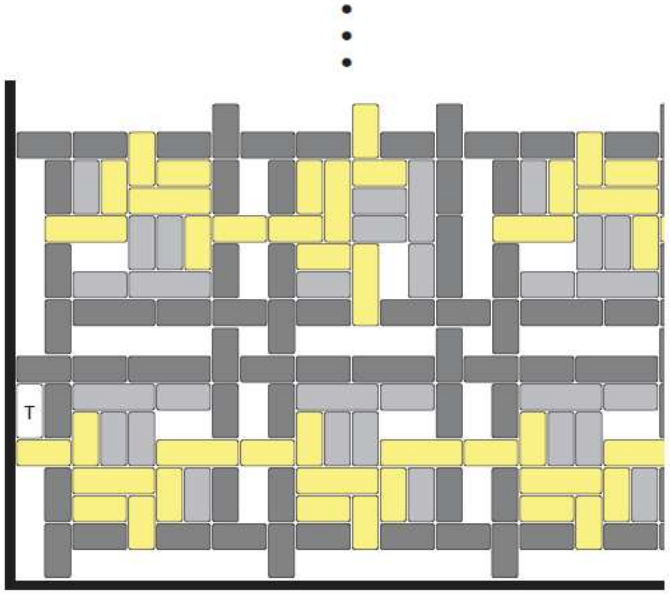


Satisfying assignment $x_1 = \texttt{true}$, $x_2 = x_3 = \texttt{false}$ gives a VC **X** of size $3 + 2 \cdot 2 = 7$, for 3 literals and 2 clauses.

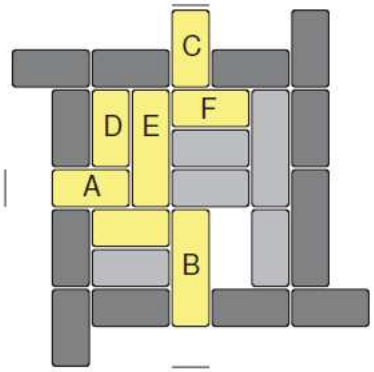
Basic idea: gadgets



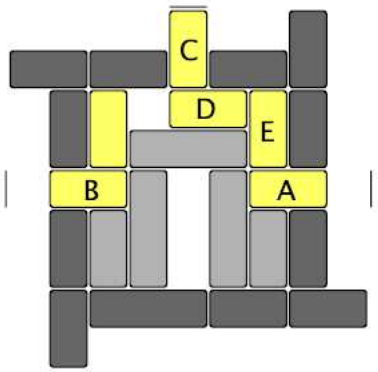
translate into



(a) Layout



(b) AND



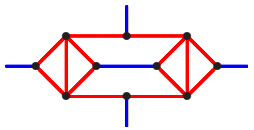
(c) Protected OR

Suppose we want to show a game to be NP/PSPACE-hard (formally: some related (y/n)-decision problem Π).

For this purpose we produce a reduction from a known well-chosen *graph* game (formally: some related (y/n)-decision problem Π', hopefully with planar graphs) to Π.

The less complicated Π' is, the better. If we are lucky, we only have to show how certain basic constructs are "emulated" by means of gadgets. Plus many details . . .

We also have gadgets to emulate certain (sub)graph behaviour in the graphs themselves.

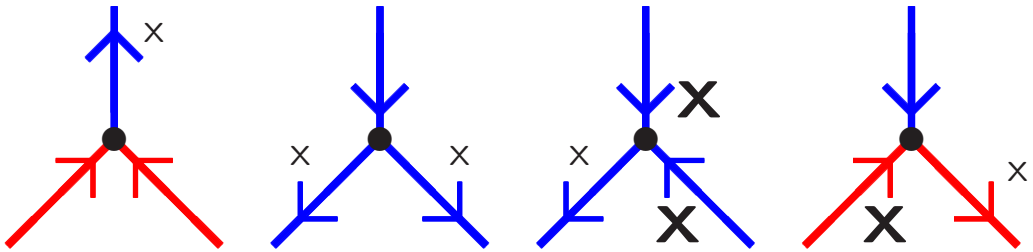


Constraint graphs

Constraint graphs consist of AND- and OR-nodes:

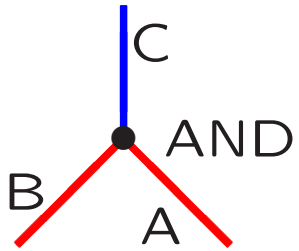
Edges are always directed such that every node = vertex receives a total input ≥ 2, where incoming blue edges contribute 2 and incoming red edges 1.

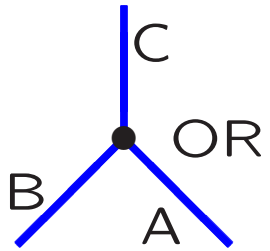
Examples:



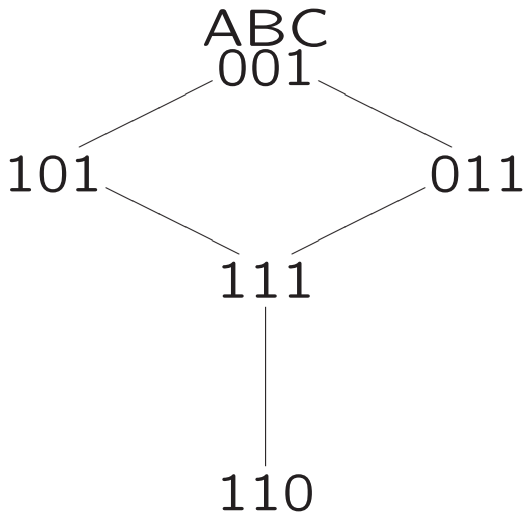
An edge can be reversed if all total inputs remain ≥ 2 (**X**).

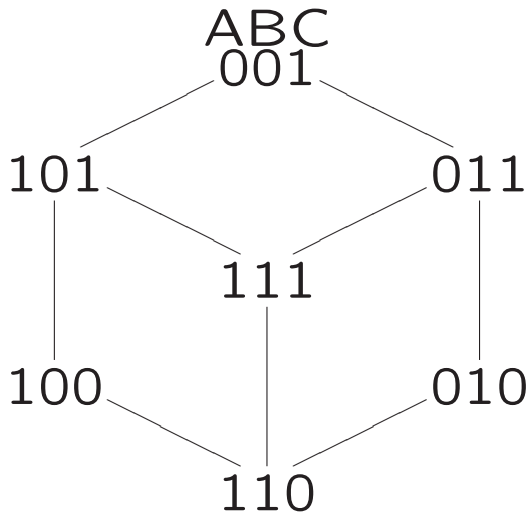
AND- and OR-node





External behavior of these gadgets can be described by the statespaces below (where 1: points in; 0: points out):

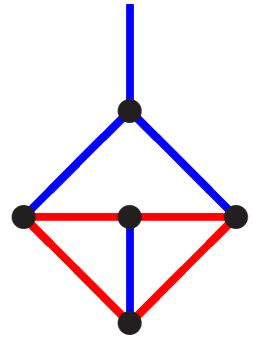




Simple gadgets

We have several simple gadgets available:

- free blue-edge terminator (FBET)
- *constrained blue-edge terminator* (CBET):

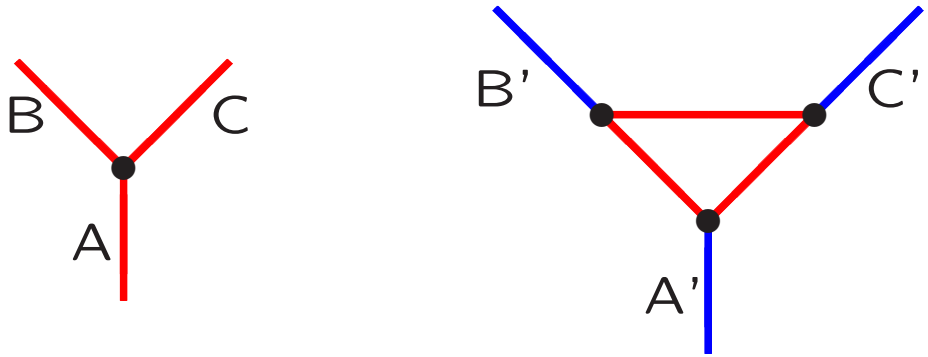


- free red-edge terminator (do we need this?)

Exercise: Explain the CBET (arrows? statespace?).

Exercise: Develop a FBET.

The *CHOICE-vertex* (left) can be emulated by the gadget on the right:



Exercise: Show that the emulation works.

Don't worry about the fact that A, B and C are all red or blue. What matters now is whether they point in or out.

And in reality edges are always directed (have arrows)!

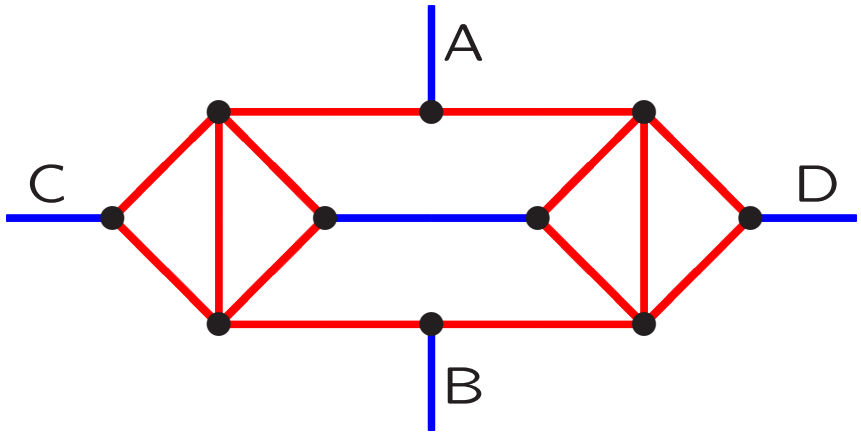
Edge crossings

In many graphs we have (unavoidable) *edge crossings*.

We now want a gadget that can replace such a crossing. So assume that we have two crossing blue edges. (There is no node where the edges cross.)

If we have such a gadget, we need only emulate *planar graphs* in our reductions to specific games — and these are often planar (flat)!

Crossover gadget



Exercise: Show that A and B may not both point out.

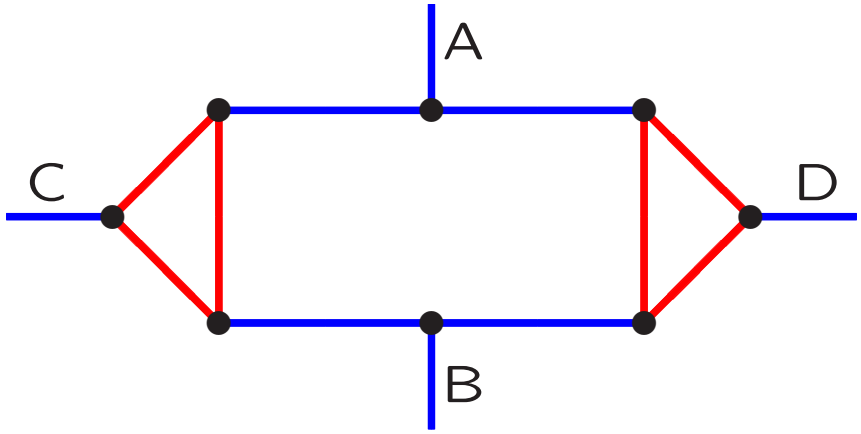
Exercise: Show there are $\leq 2^4 - 7 = 9$ states for ABCD.

Exercise: Show that this emulates two crossing edges.

Exercise: And if each edge may be reversed at most once?

Half-crossover gadget

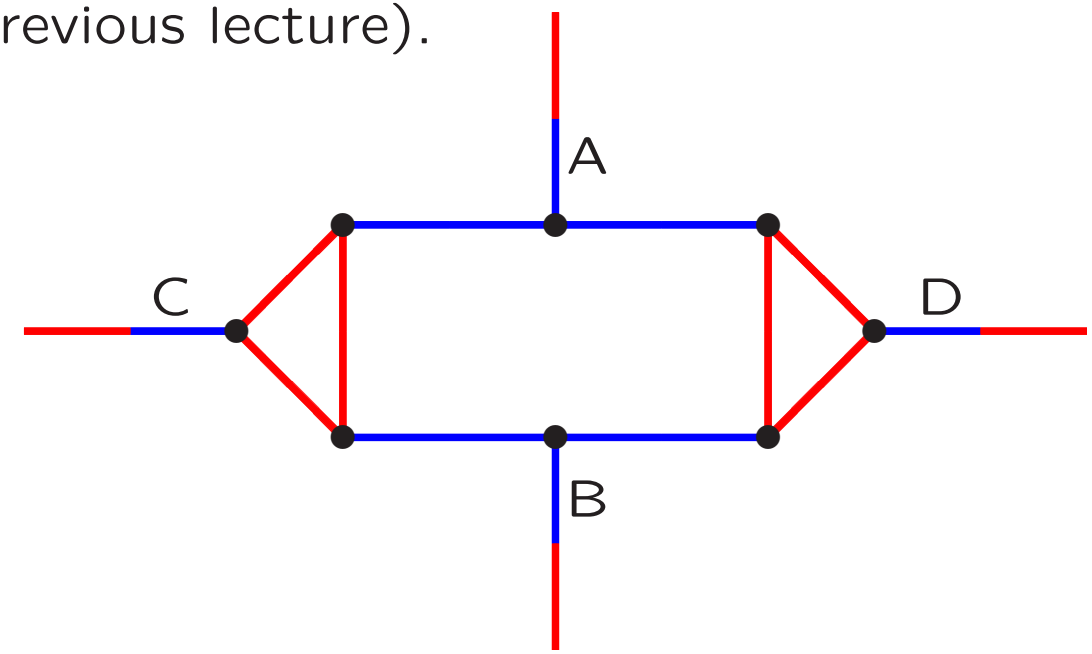
Wait a minute: did we just use "4-red-nodes"!?
The gadget below needs any 2 from A/B/C/D to go in:



Exercise: Show that this can replace a "4-reds-node".

Some issues

Remember that edges can "change color" on their way. The color is where the edges meet the vertices, so to speak (see previous lecture).

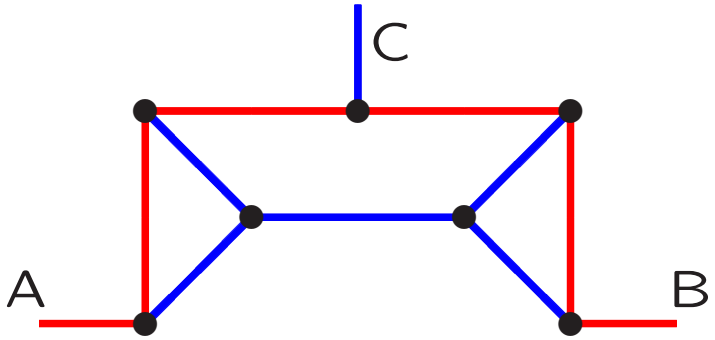


Exercise: Still OK if each edge may be reversed at most once?

In that case we (unfortunately) need a "race condition".

Protected-OR

For a *protected-OR-vertex* two of the three incident edges are special: they are *not both* allowed to be directed inward (by some outside force).



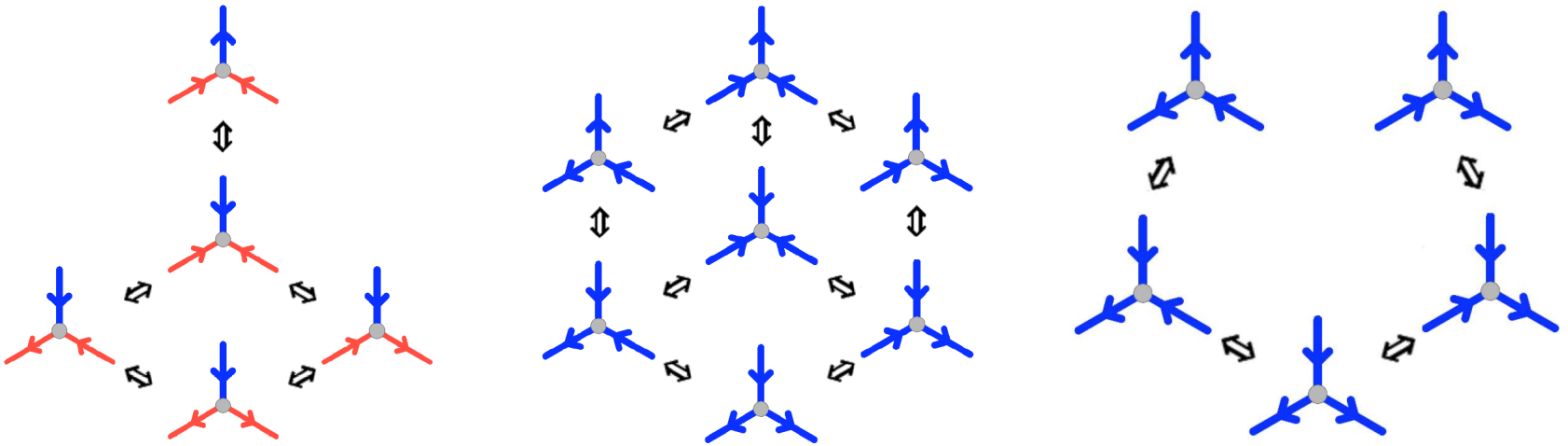
Exercise: Show that this emulates an OR-node.

(Remember again that A and B can "change to blue".)

Exercise: Where are the protected-OR-nodes in the gadgets?

Exercise: Describe the statespace of a protected-OR-node.

The statespaces for AND, OR and protected-OR:



Reductions between problems concerning games are often based on simple gadgets . . . and a lot of technique . . . and peculiarities.

Thanks:
Erik Demaine & Bob Hearn (book) and Jan van Rijn.