6.890	Lecture 17	Oct. 30, 2014

Constraint Logic: [Hearn & Demaine 2009]

Constraint graph (a model of computation)

= graph with red & blue edges → MACHINE
  - red: weight 1
  - blue: weight 2

+ orientation of edges → CONFIGURATION
  such that incoming weight ≥ 2 at each vertex
  → INFLOW CONSTRAINT

- move = reversal of one edge
  resulting in valid configuration
  (i.e. satisfying inflow constraint)
  ⇒ every move can be undone (immediately)
- asynchronous move: directed edge ↔ undirected
  - equivalent power [Viglietta – CCCG 2013]

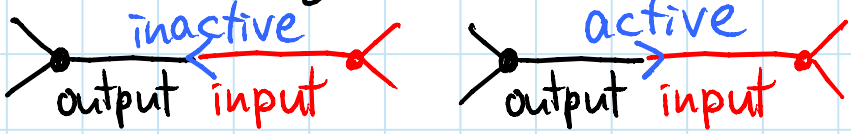
Nondeterministic Constraint Logic: (NCL)

- given constraint graph
- find a sequence of moves
- goal 1: reverse specified edge
- goal 2: reach specified configuration

- PSPACE-complete, even for just 2 vertex types: AND & OR

AND vertex = 2 incident red edges → inputs
+ 1 incident blue edge → output

- output can activate = be directed out only if both inputs active = directed in
  - edges are consistently (in)active from both ends:



- but there can be a delay between input activations & output activation

SPLIT vertex = 1 incident blue edge → input
+ 2 incident red edges → outputs

- outputs can activate only if input active
- alternative view of AND vertex

OR vertex = 3 incident blue edges
→ 2 inputs + 1 output

- output can activate only if at least one of the inputs active

NOT vertex is impossible:

- goal: output can activate only if input is not activated
  or: output never activated when input is
- inflow constraint always happier to have activated inputs & de-activated outputs

CHOICE vertex = 3 incident red edges
→ 1 input + 2 outputs
- output can activate only if input active & other output not active
- gadget reduction to AND/OR

Red-blue conversion:
- needed for e.g. output of AND or OR (blue) → input of AND or CHOICE (red)
- gadget on pairs
- will force even #

CNF formulas (ANDs of ORs)
via dual-rail logic for x_i & $\overline{x_i}$
- force at most one true via SPLIT
- output can activate only if formula is satisfiable

Wire terminators: → degree-1 vertices
- unconstrained blue & red terminators
(why red, instead of red-blue conversion? to force equal # red-blue conversions)
- forced-inward blue terminator

Constraint Graph Satisfaction: ∃ configuration?
- NP-complete [Hearn]

NCL is PSPACE-complete:

- reduction from CNF QSAT
- latch gadget – one bit of memory
  - "unlock" input & two outputs A & B
  - when locked, state is fixed:
    can output A or can output B (never both)
  - when unlocked, state is free to flip
    (and can output both A & B)
- existential quantifier gadget
  - latch to make guess
  - lock before activating rest of formula
- universal quantifier gadget
  - upper latch to set & lock variable
  - lower latch set up initially, (try-in inactive)
    settable down if x=0 & satisfied-in (⇒try-out)
  - satisfied-out only if latch down & x=1 & sat-in
- final satisfied-out flippable ⇔ formula true
- attach latch, flip, unwind ⇒ config-to-config.

Planar NCL is PSPACE-complete

- crossover gadget:
  - B can point down ⇔ A can ⇔ I can
  - D can point right ⇔ C can ⇔ E can
- to cross red edges: convert to blue & back
- vertex with 4 red edges:
  ≥2 edges must face inward

Grid constraint graphs:

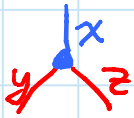
- 2×2 & 2×3 filler gadgets (all active)
- straight, turn, AND/OR gadgets

Protected OR: guaranteed ≤ 1 input activated

- can build OR
- use of red-blue conversion OK (forced config.)

Reconfiguration 3SAT:

- given 2 satisfying assignments to 3CNF formula
- move = flip one variable false ↔ true
- ∃ move sequence from one assignment to other?
  [Gopalan, Kolaitis, Maneva, Papadimitriou – SICOMP 2009]
- PSPACE-complete
- easy reduction from NCL: [Eisenstat 2014]
  - edge → variable (0/1 indicates orientation)
  - OR vertex 

→ (x in) ∨ (y in) ∨ (z in)
  - AND vertex 

→ (x out ⇒ y in) ∧ (x out ⇒ z in) (2CNF)
  - formula = AND of all these clauses
- NCL is essentially a special case of this problem
- other reconfiguration problems:
  [Ito, Demaine, Harvey, Papadimitriou, Sideri, Uehara, Uno – TCS 2011]

Sliding-block puzzles: (initial motivation)
- rectangular blocks in rectangular box
- move = noncolliding slide
- goal: move one block, e.g. out hole of box

- PSPACE-complete even for 1×2 blocks
[Hearn & Demaine 2002]

Sliding tokens = reconfig. Independent Set
- like 1×1 blocks on a graph but require no adjacent tokens

Rush Hour: [Flake & Baum 2002; Hearn & Demaine 2002]
- blocks can only slide in long direction
- PSPACE-complete for 1×2 & 1×3
- 1×2 PSPACE-complete } [Tromp & Cilibrasi 2008]
- 1×1 OPEN
- triangular PSPACE-complete

Hinged dissection: chain of blocks folding polygon A → polygon B
- always exist, avoiding collisions
[Abbott, Abel, Charlton, Demaine, Demaine, Kominers - DCG 2008]
- polyabolo font - collisions?
[Demaine & Demaine - J. Rec. Math. 2003]
- avoiding collisions is PSPACE-complete
[Hearn & Demaine]

Sokoban: [Culberson 1998; Hearn & Demaine 2002]
- PSPACE-complete
- most blocks where they need to be
- goal: satisfy formula, move 1 block, unwind
- can't wedge a block immovable
- AND/OR gadgets
- parity fix via stretching
- tunnels to reach all areas
- turn gadget

Push-2F: [Demaine, Hearn, Hoffmann - CCCG 2002]
- lock gadget } enough for Viglietta framework
- crossover
- NCL AND/OR out of that

Rolling block mazes: [Holzer & Jacobi - FUN 2012]
- 1×1×2 blocks, which can "roll" onto clear space
- rectangular frame

Plank puzzles / River Crossing [Hearn 2004]
- player can traverse, pick up, drop planks
- can hold only one at a time
- planks must end on posts
- global traversal of gadgets via length-3 planks

Dynamic map labeling: [Buchin & Gerrits – ISAAC 2013]

- want to reconfigure labels (squares) next to points while adding / panning / zooming map

Partial searchlight scheduling: [Viglietta – CCCG 2003]

- searchlight = rotable ray around a point
- intruder can move super fast but not through a light ray
- want to guarantee a region within a polygon is intruder-free

6.890 Algorithmic Lower Bounds: Fun with Hardness Proofs

Fall 2014