Games and Complexity Theory

Vedant Valluri

November 2024

1 Background

Combinatorial game theory helps us analyze the outcomes of games with optimal play. This begs the question of how efficiently we can compute the winner, or find a winning strategy for one of these games, inviting the use of complexity theory. This allows us to classify problems into complexity classes making it easier to understand if we can find the solutions to certain problems as well as their relation to other problems.

Definition 1.1. $O(f(n))$ is the set of functions that grow asymptotically no faster than $f(n)$. Formally, a function $f(n)$ is in $O(g(n))$ if

$$\text{there exist } c, n_0 > 0 \text{ such that } f(n) \leq c \cdot g(n) \text{ for all } n \geq n_0.$$

Definition 1.2. The complexity class P consists of all languages decided by a deterministic Turing machine in $O(p(x))$ for a polynomial $p(x)$. The complexity class NP consists of all languages for which an input string can be verified by a deterministic Turing machine in polynomial time.

Definition 1.3. A *polynomial-time reduction* from a problem A to a problem B is a function f that transforms instances of A into instances of B such that:

- $x \in A$ if and only if $f(x) \in B$, and
- f can be computed in polynomial time.

Definition 1.4. A problem is NP-hard if every problem in NP can be reduced to it in polynomial time. A problem is NP-complete if it is both in NP and NP-hard.

Definition 1.5. The *space complexity* of a computation is the maximum number of cells used by a Turing machine as a function of the input size. The class PSPACE consists of all decision problems that can be solved by a deterministic Turing machine using polynomial space.

Definition 1.6. A problem is PSPACE-complete if:

- It is in PSPACE.
- Every problem in PSPACE can be reduced to it in polynomial space.

Definition 1.7. The *Quantified Boolean Formula* (QBF) problem is that given a boolean expression and universal ($\forall$) and existential ($\exists$) quantifiers for each variable in the expression, we can

determine whether the expression can be satisfied. This problem has been proven to be PSPACE-complete.

2 Redwood Furniture in Hackenbush

DEFINITION 2.1. A piece of *redwood furniture* in HACKENBUSH is a position in which

- No red edges touch the ground.
- All blue edges have one vertex on the ground (**foot**) and the other touching a unique red edge (**leg**).

THEOREM 2.1. Every connected piece of redwood furniture has value $\frac{1}{2^n}$ for some n

Proof. Let's take some position of connected redwood furniture G. We claim that for every move that Left makes (G^L) there is a corresponding move that Right can make (G^{LR}) such that $G^{LR} \leq G$. Let's say Left takes some foot. Since no red edge touches the ground the leg attatched to the foot must still be standing. Right can take this edge. Clearly by taking away a foot, the position has gotten worse for Left so it must be true that $G^{LR} \leq G$. This means that all options G^L are reversible and can be replaced all G^{LRL}. Since making moves in redwood furniture still leaves the position satisfying the conditions, we can recursively apply reversible options until Left only has a move to 0 remaining. This means that $G = \{0 \mid G^R\}$ where G^R also satisfies the same condition. By induction on the number of edges, G must be equal to $\frac{1}{2^n}$ for some integer n. □

DEFINITION 2.2. Say that a *redwood tree* is a position of redwood furniture such that removing any red edge would leave it disconnected. We call a position of redwood furniure such that all non-leg edges have only one end at the top of a leg a *redwood bed*.

Say that a *worthwhile* move for Right is one that leaves a piece of redwood furniture connected and the value will now be at most $\frac{1}{2^{n-1}}$. This follows from the *Don't-Break-It-Up Theorem* from *Winning Ways*, which, as the name suggests, advises to leave pieces connected.

LEMMA 2.1.1. Any redwood bed A such that m moves can be made to some redwood tree T satisfies $A = \frac{1}{2^{m+1}}$.

Proof. If there is a worthwhile move to a position B from a position A then we have $A = \{0 \mid B\}$. We can continue to apply this until reaching the redwood tree T. This means that $A = \frac{1}{2^m}T$. We claim that such redwood trees have the value $\frac{1}{2}$. We can prove this by induction. If a move is made by Right in a tree, it will break up either into two smaller trees or one tree and a HACKENBUSH stalk BRR. In either case T will evaluate to $\{0 \mid \frac{1}{2} + \frac{1}{2}\} = \frac{1}{2}$ or $\{0 \mid \frac{1}{2} + \frac{1}{4}\} = \frac{1}{2}$. This means that $T = \frac{1}{2}$ and $A = \frac{1}{2^{m+1}}$. □

From this we have that finding the value of a redwood bed is the same is finding the smallest redwood tree which contains all of the legs.

Given an undirected graph with non-negative edge weights and a subset of vertices called terminals, the *Steiner Tree Problem* is to find a tree of minimum weight such that it contains all terminals. This problem has been proved to be NP-hard.

THEOREM 2.2. Determining the value of a redwood bed is NP-hard

Proof. Finding the smallest redwood tree containing all of the legs is equivalent to finding the Steiner tree of the redwoood furniture position where the terminals are the vertices atop the legs. This means that we have a reduction between the two problems and determining the value of a redwood bed is NP-hard. □

3 Complexity of Puzzles

3.1 Constraint Logic

When analyzing puzzles, something helpful is the *nondeterministic constraint logic* (NCL) model of computation. With this we define something called a *constraint graph* and valid moves within it.

DEFINITION 3.1. A *constraint graph* is an directed graph with weighted edges and an in-flow constraint. Red edges have weight 1 while blue edges have weight 2. All vertices satisfy that the sum of the weights of the edges pointed inwards as 2 at a minimum. A valid move on this graph is the reversal of the direction of some edge such that the in-flow constraint is still satisfied.

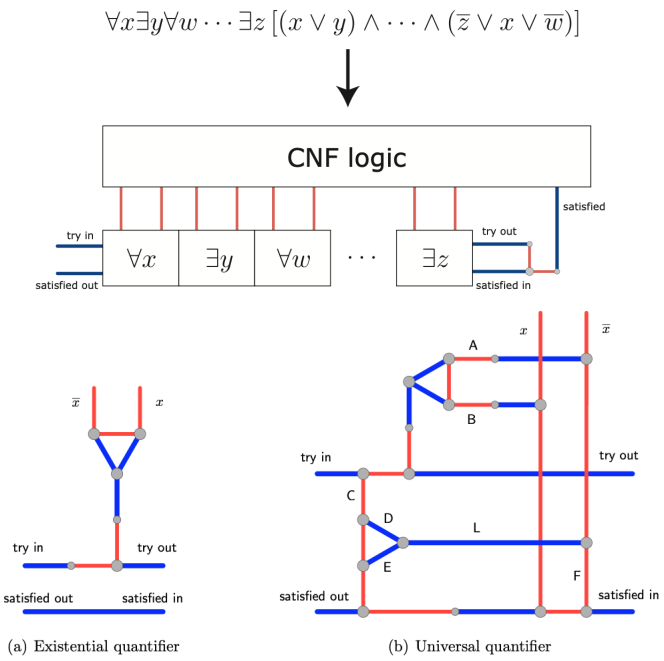
From this, the decision problem that arises is whether, given a constraint graph, a sequence of moves can lead to the reversal of some specific edge.

The pictures above show vertices that depict AND and OR gates within a constraint graph. In the AND gate, only if both red edges point inward can the blue edge point outward. Similarly, for the OR gate, only one of the two blue edges point inward (two input edges being chosen) must be pointing inward for the output edge to put out. The graph is *nondeterministic* because at any time it is up to the player whether to actually perform a legal reversal.

From these basic vertices, we can also construct a red-blue conversion gadget, a free edge terminator, and a constrained edge terminator. The red-blue conversion gadget allows for the connection of two differently colored edges while the terminators allow for a degree-1 vertex. When looking at RUSH HOUR later, we will need that no two edges intersect. Fortunately, we can construct a crossover gadget, which is able to convert any constraint graph into a planar constraint graph (no intersecting edges at non-vertices). Through these, we can allow for any boolean formula in CNF (conjunctive normal form) to be represented with AND and OR vertices in a planar constraint graph.

THEOREM 3.1. There is a reduction to determining whether an edge can be reversed given a constraint graph from the quantified boolean formula problem.

Proof. We can create the following graph to represent a QBF problem.



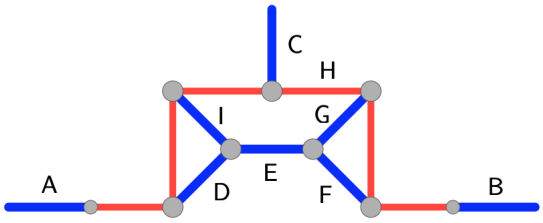
(a) Existential quantifier (b) Universal quantifier

We can construct special gadgets using just AND and OR vertices to work as universal and existential quantifiers and we have already seen that we can represent the boolean formula with such vertices as well. From here, finding whether the quantified boolean formula is satisfiable is the same as finding out whether the left satisfied out edge can be switched to directing outward. □

Since the state of the constraint graph can be encoded in linear space and the possible moves can be computed in polynomial time non-deterministically, NCL is in PSPACE. This means that Nondeterministic Constraint Logic is PSPACE-complete.

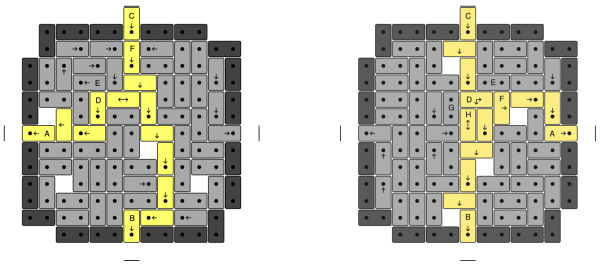
3.2 Rush Hour

To prove the PSPACE-competeness of RUSH HOUR we find AND and OR gadgets for a reduction from planar NCL to RUSH HOUR. The game being studied will be the standard ruleset of RUSH HOUR where there are 1×2 and 1×3 blocks and blocks may only move along the grid in the direction in which they are originally pointed.



As it turns out, there are some situations in the game of RUSH HOUR where if both blocks are moved in an OR gadget, the whole puzzle falls apart. This creates the need for a PROTECTED OR gadget

where at most one of the graph edges can be pointed inwards. This is not a problem as we can just add a new constraint to PROTECTED OR vertices preventing both input edges from being pointed in. A regular OR gadget can be constructed with ANDs and PROTECTED ORs.



Above are the gadgets created to represent the AND and the PROTECTED OR vertices in NCL. In the left figure, only if both block A and block B are moved left and down respectively can the corresponding edges point towards the vertex in the contraint graph, allowing the corresponding edge for C to reverse, meaning the C block can move inwards. Similarly, only if one of the blocks A or B, but not both, slide right or down respectively, the C block may move down. With additional wiring to connect the gadgets, we can create any arbitrary planar graph using them.

THEOREM 3.2. RUSH HOUR is PSPACE-complete.

Proof. As we can encode any planar graph in a RUSH HOUR position, there is a reduction from NCL to RUSH HOUR. As RUSH HOUR is clearly in PSPACE, RUSH HOUR must be PSPACE-complete. □

References

[1] Elwyn R. Berlekamp, John H. Conway, and Richard K. Guy. Winning ways for your mathematical plays. Vol. 1. A K Peters, Ltd., Natick, MA, second edition, 2001.

[2] Hearn, R. & Demaine, E. The Nondeterministic Constraint Logic Model of Computation: Reductions and Applications.