The Game of Cubic is NP-complete

Erich Friedman
Stetson University
efriedma@stetson.edu

Introduction

In the puzzle solitaire game of Cubic, there are variously colored unit blocks in some configuration. The player is allowed to drag blocks to horizontally adjacent and unoccupied positions. Gravity pulls unsupported blocks down into available spaces. When more than one block of the same color come into contact, they disappear. The goal is to remove all the colored blocks.

An example of a puzzle is shown in Figure 1. We use differently numbered blocks instead of colored blocks. The game of Cubic can be played on-line at http://www.agon.com/doodle/four.html.

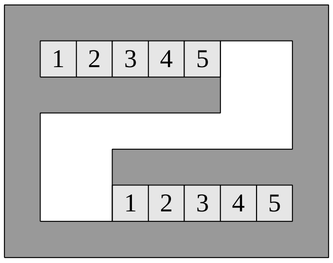

Figure 1. A small Cubic puzzle

We will show that the question of whether or not a given Cubic position is solvable is NP-complete. To do so, we simulate Boolean circuits where "wires" carry truth values, and junctions in these wires simulate logical gates. The question of whether a solution exists corresponds to the canonical NP-complete problem Satisfiability [3], which asks whether a set of truth values for the inputs exists that makes the output true. Similar approaches are taken in [1, 2, 4-8].

The Construction

We will refer to a block with label *n* as an *n*-block. Our wires will be passageways for blocks to move in. A wire carries the value TRUE is there is a 1-block moving through it, and the value FALSE otherwise.

To create our inputs and outputs, we need to have variables which can have either truth value, and be able to end wires in a way that requires them to have a specific truth value. We will then present NOT and AND gates (which can be combined to create an OR gate). Then we present configurations to split wires and allow wires to cross.

Our variable is shown in Figure 2. The left 1-block can either exit or remove itself by moving right.

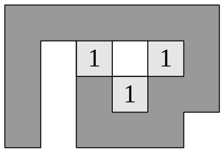

Figure 2. A variable

To make sure a wire is TRUE (or FALSE), we simply end the wire with a 1-block (or no block) as in Figure 3.

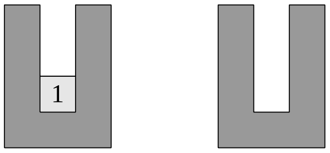

Figure 3. How to make wires TRUE (left) or FALSE (right)

Our NOT gate is shown in Figure 4. A 1-block can exit if and only if one does not enter.

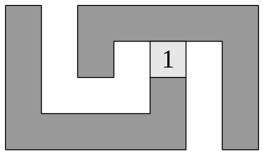

Figure 4. The NOT gate

Our AND gate is shown in Figure 5. In order to have a block exit, blocks must enter both input wires. A 1-block must enter the left input wire, allowing the 2-block to slide to the right, allowing a 1-block from the right input wire to exit. The 1-blocks at the top give an optional 1-block to remove any 1-blocks that enter but do not exit. The eight blocks at the bottom prevent two entering 1-blocks from cancelling themselves out.

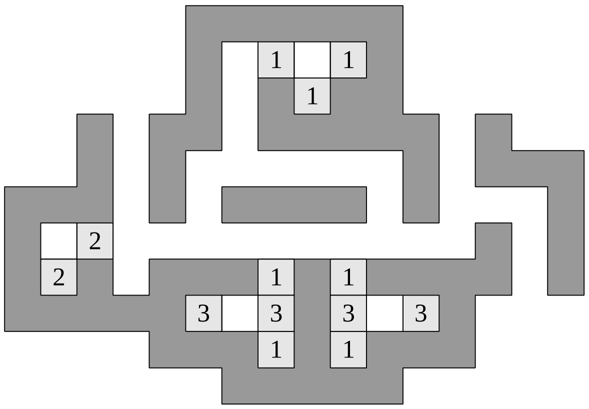

Figure 5. The AND gate

We now present a configuration to switch the color of the signal carrying block. We will need this construction to split a signal or allow signals to cross. This configuration, which we call SWTICH, is shown in Figure 6.

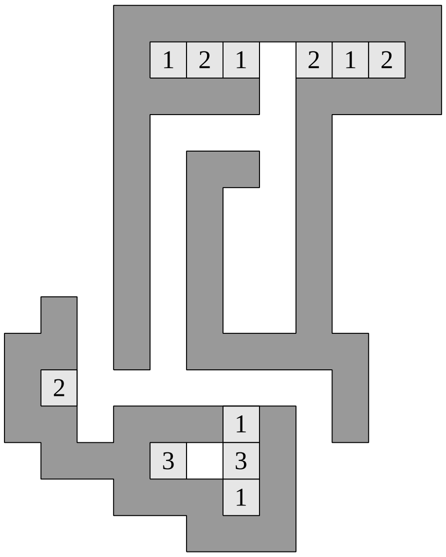

Figure 6. The SWITCH configuration

If no 1-block enters, a 2-block can leave the top part to cancel with the 2-block in the lower left. If a 1-block enters, the 2-block can exit, and a 1-block from above can be used to cancel the remaining 1-block. The four blocks at the bottom prevent a 1-block from exiting. There is no way that the truth value of the signal can change.

The configuration that allows two signals to cross is shown in Figure 7. The key is to switch the color of one signal before and after the crossing. The blocks at the bottom prevent a 1-block from exiting left and a 2-block from exiting right.

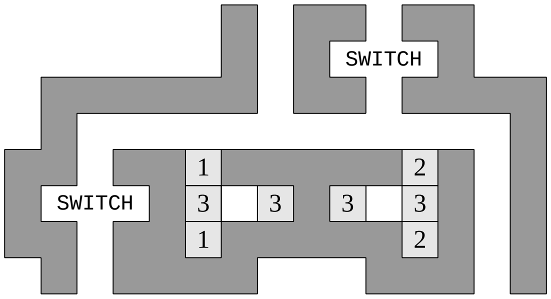

Figure 7. The configuration to allow crossing of wires

The configuration that duplicates a signal is shown in Figure 8. If a 1-block enters, a 2-block must help it exit, but a 2-block can not exit unless a 1-block has entered and helps it exit.

From these building blocks, it is clear that for any given Boolean circuit, we can construct a Cubic position that is solvable if and only if the circuit is satisfiable. The mapping from circuits to Cubic is bounded, so any polynomial time verification of the Satisfiability problem extends to a polynomial time verification for Cubic. Thus Cubic is NP-complete.

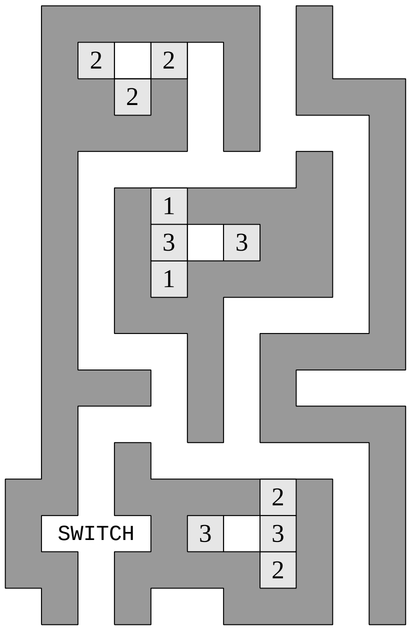

Figure 8. The configuration to duplicate signals

References

[1] D. Beauquier, M. Nivat, E. Rémila, and J.M. Robson, *Computational Geometry* **5** (1995) 1-25.

[2] J. Culberson, "Sokoban is PSPACE-complete." preprint.

[3] M.R. Garey and D.S. Johnson, *Computers and Intractibility: A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.

[4] E. Goles and I. Rapaport, "Complexity of tile rotation problems." *Theoretical Computer Science* **188** (1997) 129-159.

[5] E. Goles and I. Rapaport, "Tiling allowing rotations only." *Theoretical Computer Science* **218** (1999) 285-295.

[6] R. Kaye, "Minesweeper is NP-complete." *Mathematical Intelligencer,* to appear.

[7] K. Lindgren, C. Moore and M.G. Nordahl, "Complexity of two-dimensional patterns." *Journal of Statistical Physics* **91** (1998) 909-951.

[8] C. Moore and J.M. Robson, "Hard Tiling Problems with Simple Tiles." preprint.

[9] E. Rémila, "Tilings with bars and satisfaction of Boolean formulas." *Europ. J. Combinatorics* **17** (1996) 485-491.