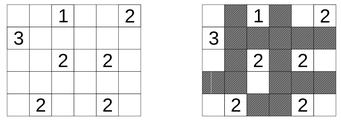
Nurikabe is NP-Complete

Brandon McPhail and James D. Fix
Reed College
{mcphailb,jimfix}@reed.edu

How to Play Nurikabe

A Nurikabe puzzle and its solution:



The Goal:
Shade the cells to form connected white regions called "islands".

The Rules:

I. Don't shade in numbered cells.
II. Each island contains exactly one number, its size.
III. No 2-by-2 blocks of shaded cells.
IV. All shaded cells are connected.

The Decision Problem

Nurikabe puzzles seem hard to solve. We characterize the **decision problem** for Nurikabe:

Given a Nurikabe puzzle, does a solution exist?

P: decision problems we can answer in reasonable (*polynomial*) time
NP: decision problems whose solution we can verify in reasonable (*polynomial*) time

Nurikabe $\in$ NP? $\rightarrow$ *Yes.*
Nurikabe $\in$ P? $\rightarrow$ *We don't know!*

A problem is **NP-hard** if *any* other problem in NP can be **reduced** to it. A problem is **NP-complete** if it is in NP and is NP-hard.

Theorem: Nurikabe is NP-hard

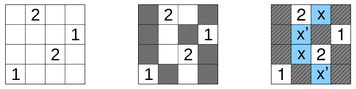
1. Any problem in NP can be reduced to Boolean circuits.
2. Boolean circuits can be reduced to Nurikabe.

Proof By Construction

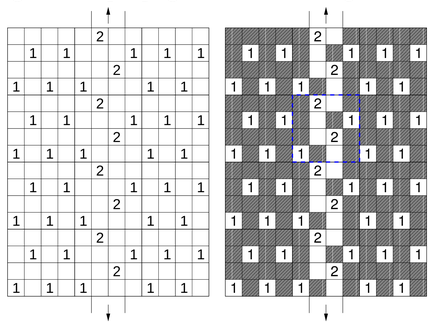
As a Boolean circuit consists of distinct components, so too will our Nurikabe puzzles be divided into separate parts. We refer to each part or component as a *tile*. Nurikabe puzzles initially have no shaded cells. For illustrative purposes, in some figures we shade those cells whose shading is easily and immediately determined.

Wire Construction

All of our circuits will be constructed on a simple tiling of 4 by 4 cells consistent with the rules of Nurikabe.



The **wire tile** has one of two possible states. If the x cells are shaded, we say the state is **false**. If the x' cells are shaded, we say the state is **true**. We can string these tiles together to propagate this *truth assignment*.



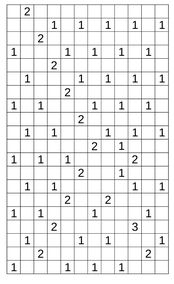
Assigning Values

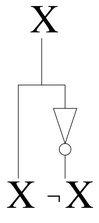
A Boolean circuit is satisfiable if and only if our Nurikabe puzzle has a solution.

To ensure this, we fix the final output wire to **true**. The input wires may take on values, and if we find any shading for them that results in a solution, this will correspond to a satisfying truth assignment for our circuit.

Splitting Wires

We'd like to split our wires to allow the output of one component to form the input for multiple components.

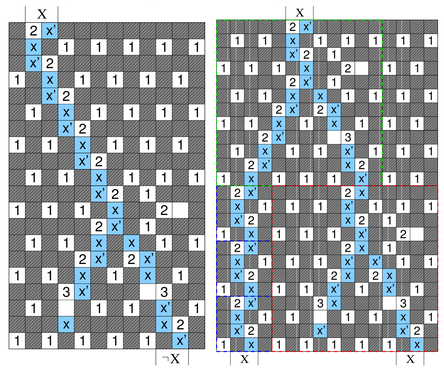




We'll call this our **branch/NOT** tile.

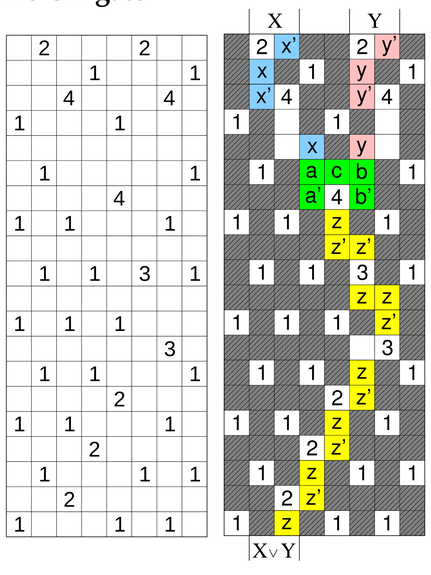
Branch and NOT Gates

If we cap the left wire of the **branch/NOT** tile, we are left with just a **NOT gate**. If we instead compose the branch/NOT tile with a NOT gate, we get a **branch gate**.



Note that we can bend our wires to align them correctly with neighboring tiles.

The OR gate



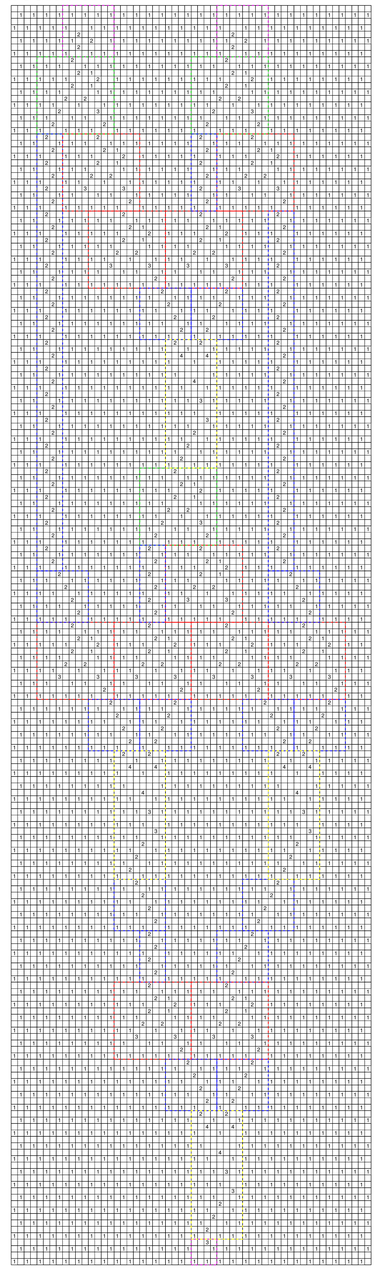
The **OR gate** takes two inputs and yields a truth assignment of *false*, if and only if both inputs were *false*. We can compose the OR gate with NOT gates to construct all combinations of AND, NAND, and NOR gates.

An Example Construction

A satisfying assignment can be found for the Boolean expression

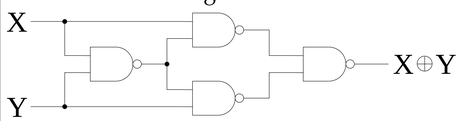
$$x \oplus y = (x \vee y) \wedge \neg(x \wedge y)$$

if and only if the following Nurikabe puzzle has a solution.

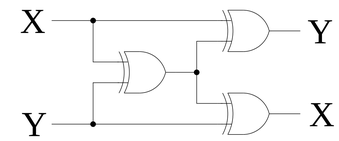


Important Details

Our circuit lies in a plane, so we need to either describe explicitly how wires should cross without interacting or show that wire crossing aren't necessary. As it turns out, we already have sufficient componentry to build a wire crossing.

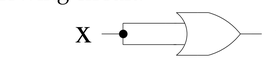


We can use 3 branch and 4 NAND gates to construct an **XOR gate**, which proves useful.

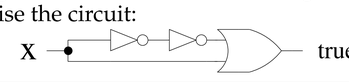


We can then use 3 branch and 3 XOR gates to allow two wires to cross.

Another concern of ours is that we may violate Nurikabe rule IV. Consider the following circuit:



There is no solution, since assigning **false** to the input wires results in a disjoint area of shaded cells between them. We require a length of wire to include two not gates. We revise the circuit:



Other puzzles shown to be NP-complete include Minesweeper, Paint By Numbers/Nonogram, Pearls, Spiral Galaxies, and Clickomania.

References

[1] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, New York, New York, 1971. Association for Computing Machinery, ACM Press.

[2] H. de Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 1(10):41–51, 1990.

[3] Erich Friedman. Spiral Galaxies Puzzles are NP-complete. Technical report, Stetson University, 2000.

[4] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[5] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, New York, New York, 1972. Plenum Press.

[6] Richard Kaye. Minesweeper is NP-complete. *The Mathematical Intelligencer*, 22(2):9–15, 2000.

[7] Richard Kaye. Some Minesweeper Configurations. Technical report, The University of Birmingham, August 2000. http://for.mat.bham.ac.uk/R.W.Kaye.

[8] C. Moore and J.M. Robson. Hard Tiling Problems with Simple Tiles. *Discrete & Computational Geometry*, 26(4):573–590, 2000.

[9] Seta Takahiro. The Complexities of Puzzles, Cross Sum, and their Another Solution Problems (ASP). The University of Tokyo, 2001. Undergraduate Thesis.

[10] Nobuhisa Ueda and Tadaaki Nagao. NP-completeness Results for NONOGRAM via Parsimonious Reductions. Technical report, Tokyo Institute of Technology, 1996.

[11] Takayuki Yato. On the NP-completeness of the Slither Link Puzzle. In *IPSJ SIGNotes ALgorithms*, pages 25–32, 2000.

[12] Takayuki Yato. Complexity and Completeness of Finding Another Solution and its Application to Puzzles. Master's thesis, The University of Tokyo, 2003.