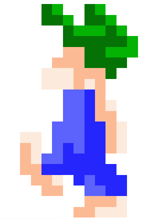
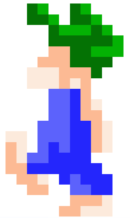


How hard are computer games?

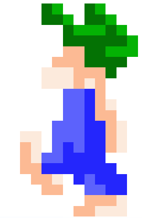
Graham Cormode, DIMACS
graham@dimacs.rutgers.edu

Introduction

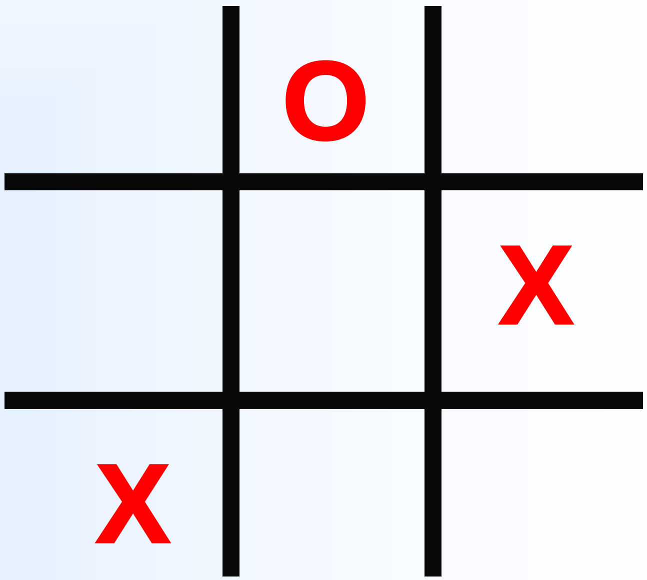


- Computer scientists have been playing computer games for a long time
- Think of a game as a sequence of Levels, where each level has some objective that must be carried out to complete the level
- Given a Game and a Level, what is the complexity of finding a solution to the level?
- Depends on : How many players? Predictable or random? How much information does player have?

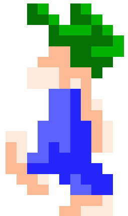
2 Player Game Puzzle



- Just a puzzle...
- "Toe-Tac-Tic": the first player to make three in a row **loses**.
- Is this game a win for the first player, a draw, a win for the second player?

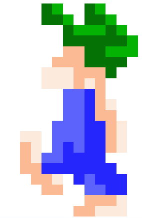


A different 2 player game



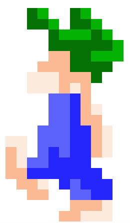
- Start with S = {1,2,3,4,5,6,7,8,9}
- Players take turns to remove an item from S and add it to their set.
- Winner is first who has a subset of size 3 that adds to 15
- Is this game a win for the first player, a draw, a win for the second player?
- Eg. A takes 5, B takes 6, A 8, B 2, A 4, B 7
  B wins: 6 + 2 + 7 = 15

Related Areas



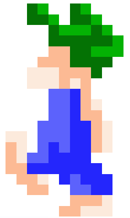
- 2 or more players, complete information, no randomness -- Game Theory, Nim games, Min-max theorem, Nash equilibrium etc...
- Very many players, partial information, some randomness -- Economics! Insider trading, Auction theory, mechanism design, internet protocols...
- 1 Player, complete information, no randomness -- Computer Games (today's topic).

Outline



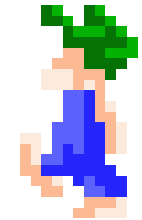
- Some prior work on computer games
- Some work in progress: Lemmings
- Hardness of Lemmings
- Hardness under restrictions
- Under what restictions is Lemmings not hard?

1 Player Computer Games



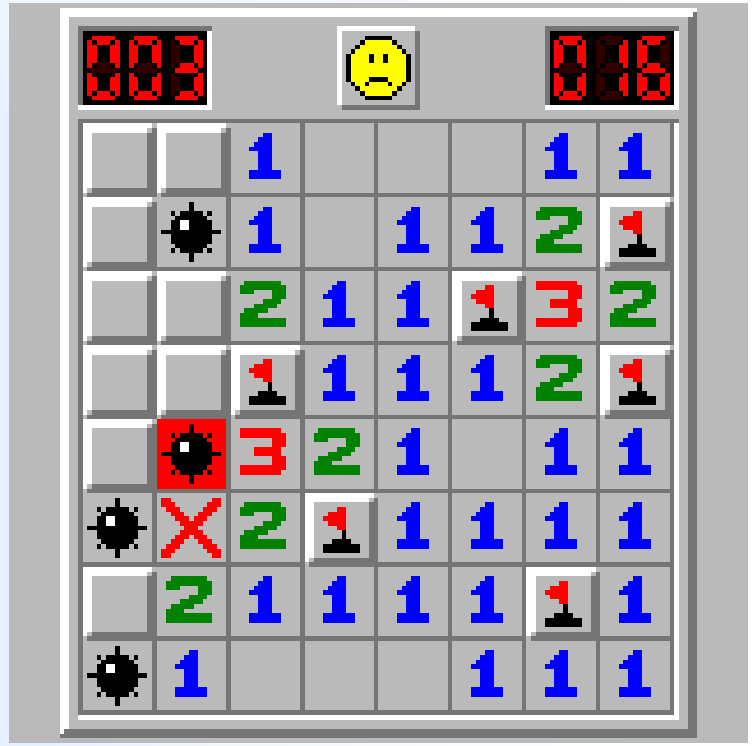
- Mostly, these are puzzle games: each level is a configuration of pieces that the player can manipulate or interact with, in order to reach some solution.
- The computer enforces the rules, but is not a player: no monsters to shoot
- Maybe there is a time limit

Example 1: Minesweeper



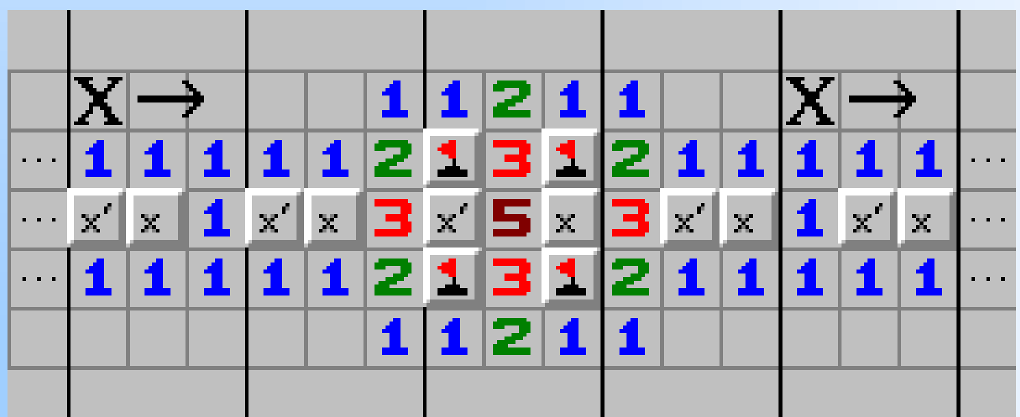
- A board with mines hidden — locate the mines but don't click on one!

- Question: Given a Minesweeper configuration (board with labels and counts) is it *consistent*? That is, is there some arrangement of mines that would give rise to that configuration?



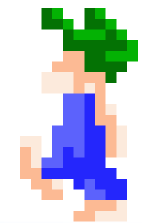
Minesweeper

- The problem is NP-hard [Kaye, 2000]
- Easy to check if a proposed layout of mines is consistent with the input
- Can encode Satisfiability problems by connecting up 'gadgets' (logic gates) made out of mines.



Wires with phase change

Example 2: Tetris

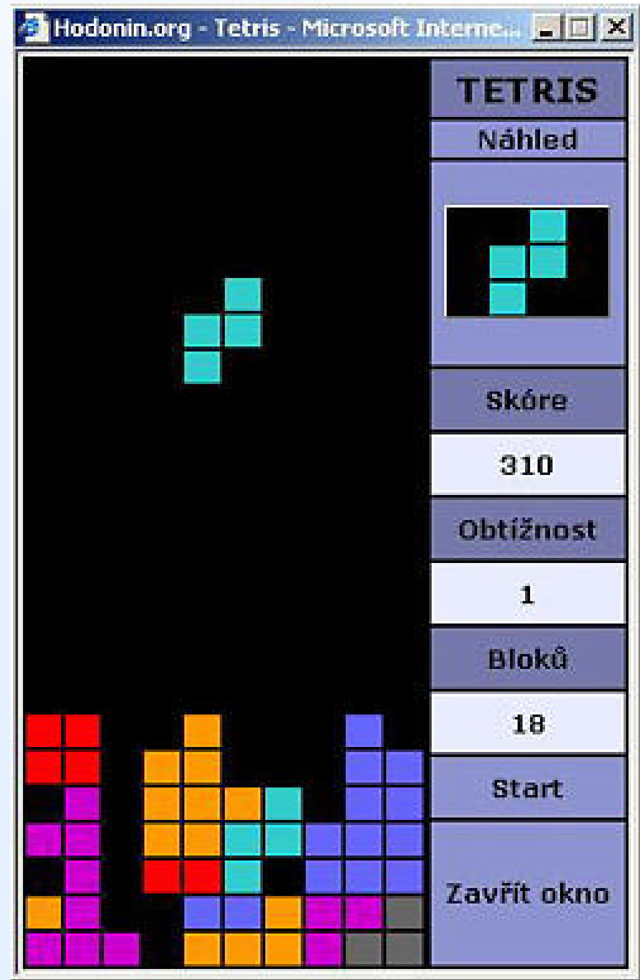


How to formalize Tetris?

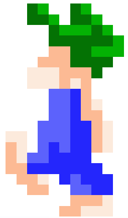
- Problem instance
  Complete information
  (1) Current board configuration
  (2) List of all future pieces.

- Decision problem: can all blocks be cleared?

- Generalization of the game:
  We must allow arbitrary sized playing area.

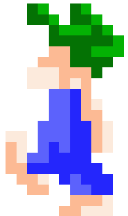


Tetris is Hard

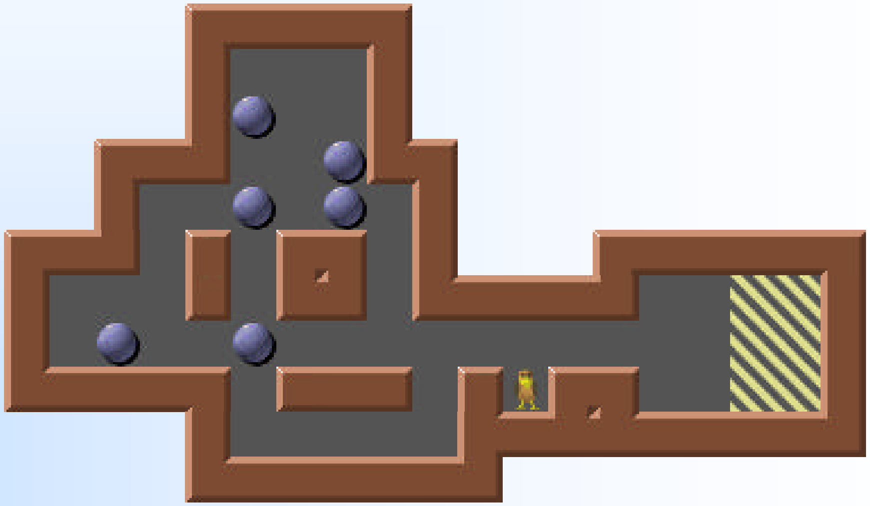


- Again, the problem is NP-Hard [Demaine, Hohenberger, Lieben-Nowell, 2003]
- This time, transform from a bin packing problem: initial configuration represents a set of bins, the game pieces in order encode a set of integers in unary.
- Show that the game board can be cleared if and only if there is a solution to the bin packing problem.

Example 3: Sokoban

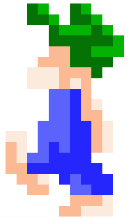


- Push blocks into storage locations
- Decision problem: Is there a strategy that stores all blocks?



- In NP: can check the proposed solution (assumes solution is polynomial in the level size)
- Not only is Sokoban NP-Hard, it is P-Space complete: can emulate a finite tape Turing Machine

Emacs is NP-Hard

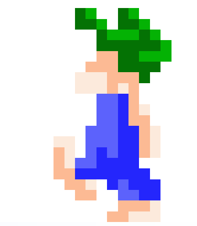


- All three of these games are in Emacs:
  M-x tetris
  M-x sokoban
  M-x xmine

- Therefore, we conclude that Emacs is NP-hard.

- Since many students use Emacs to write their theses in, we must conclude that this is also a hard task, as proved by the students who spend most of their time playing computer games...

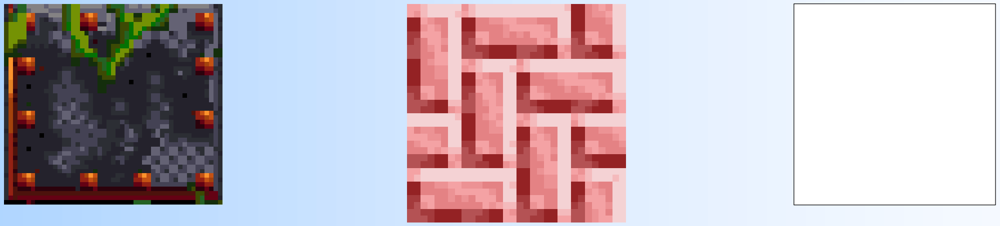
New Stuff



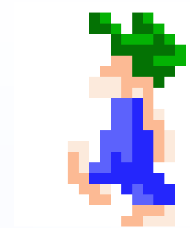
- I've been looking at the computer game 'Lemmings'

- Lemmings is quite complicated to describe to someone who hasn't played before, will attempt to give a cut-down description.

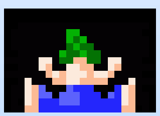
- The world is made up with of three kinds of stuff: steel, earth, and air.



Lemmings

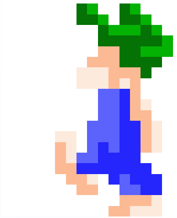


- Lemmings are stupid creatures... they keep walking in one direction until they hit a wall and turn round or fall down a hole...

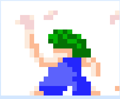


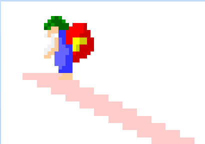
- Lemmings die if they fall too far, else they keep going
- The player can give certain skills to individual Lemmings that change how they proceed.
- The skills are permanent (stay with lemming forever), temporary (stop under certain conditions), plus two that don't fit into either category...

Skills



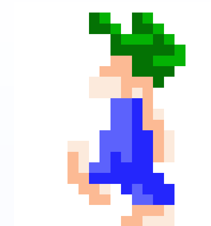
The Lemming Skills are:





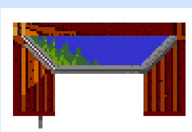
- Floater (permanent): Lemming can fall any distance
- Climber (permanent): Lemming can scale walls
- Digger (temp): Lemming digs down through earth
- Miner (temp): Lemming digs diagonally
- Basher (temp): Lemming digs horizontally
- Builder (temp): Lemming builds a small bridge
- Blocker (other): Lemming stops & blocks others
- Bomber (other): Lemming explodes & damages earth

The Lemmings Problem

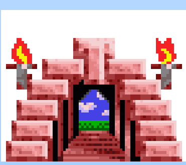


Formalize: L (a level of lemmings) is a tuple with these entries:

- *limit*: the time limit
- *save*: the number of lemmings to save
- *lems*: the number of lemmings at the start



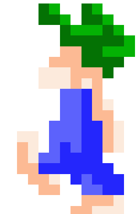
- *start*: initial position of the lemmings
- *width, height*: size of the level
- *grid*: description of the game board



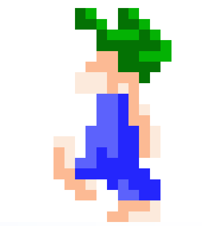
- *exit*: location of the exit
- *skills*: 8-vector listing available quantity of each skill

Problem: given L, is there a strategy that gets at least *save* lemmings to the *exit*?

Example Level

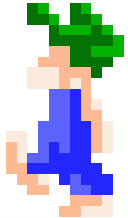


Outline of Hardness Proof



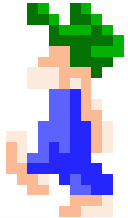
- Show that Lemmings is hard by encoding instance of 3-Sat (m clauses, n variables).
- Will show that the level is solvable iff the instance is satisfiable
- First need to show that the problem is in NP

Lemmings is in NP



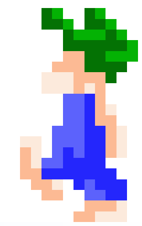
- Informally, the computer game shows Lemmings is in NP: the player provides the "certificate", and computer checks it.
- Formally, write down a strategy (step-by-step description of what to do) then check this certificate in poly-time: each move is valid, enough are saved.
- Detail: want the strategy to be poly in input size.
- Fix by insisting that time limit is bounded by poly in grid size — then check each step in poly time.

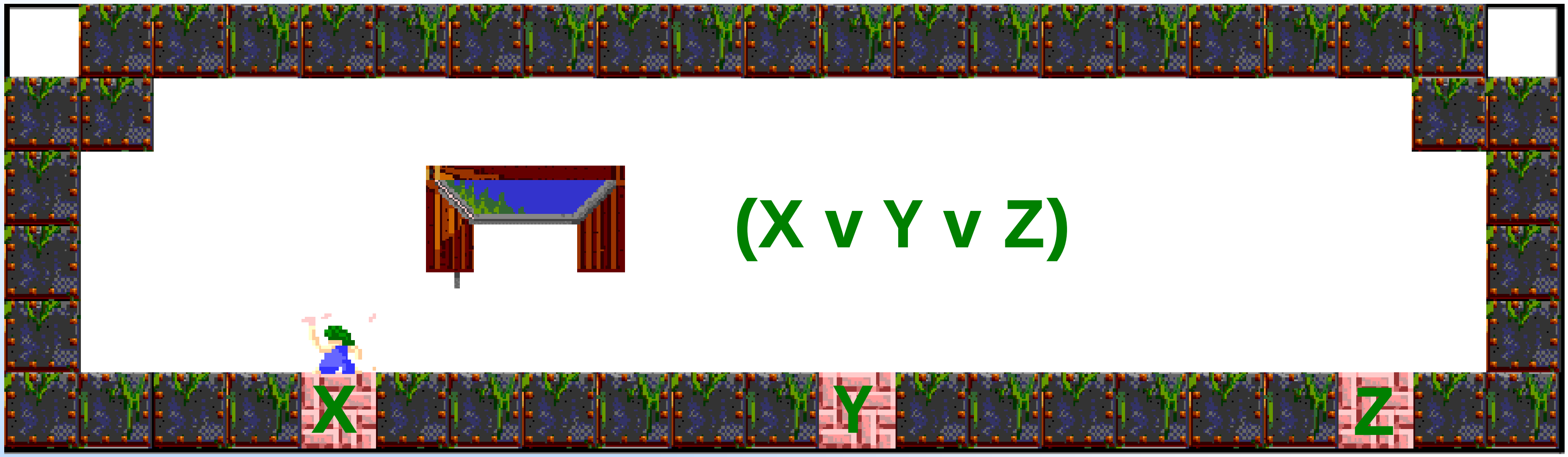
Encoding 3Sat



- Use a bunch of gadgets, then 'wire' these together to make the encoding.
- Use one lemming to represent each clause, and another lemming for each variable.
- Clause lemming chooses one of the literals in the clause. Only reaches exit if that literal is satisfied.
- Variable lemming sets its variable to true or false.

Clause Gadget



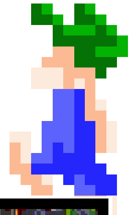


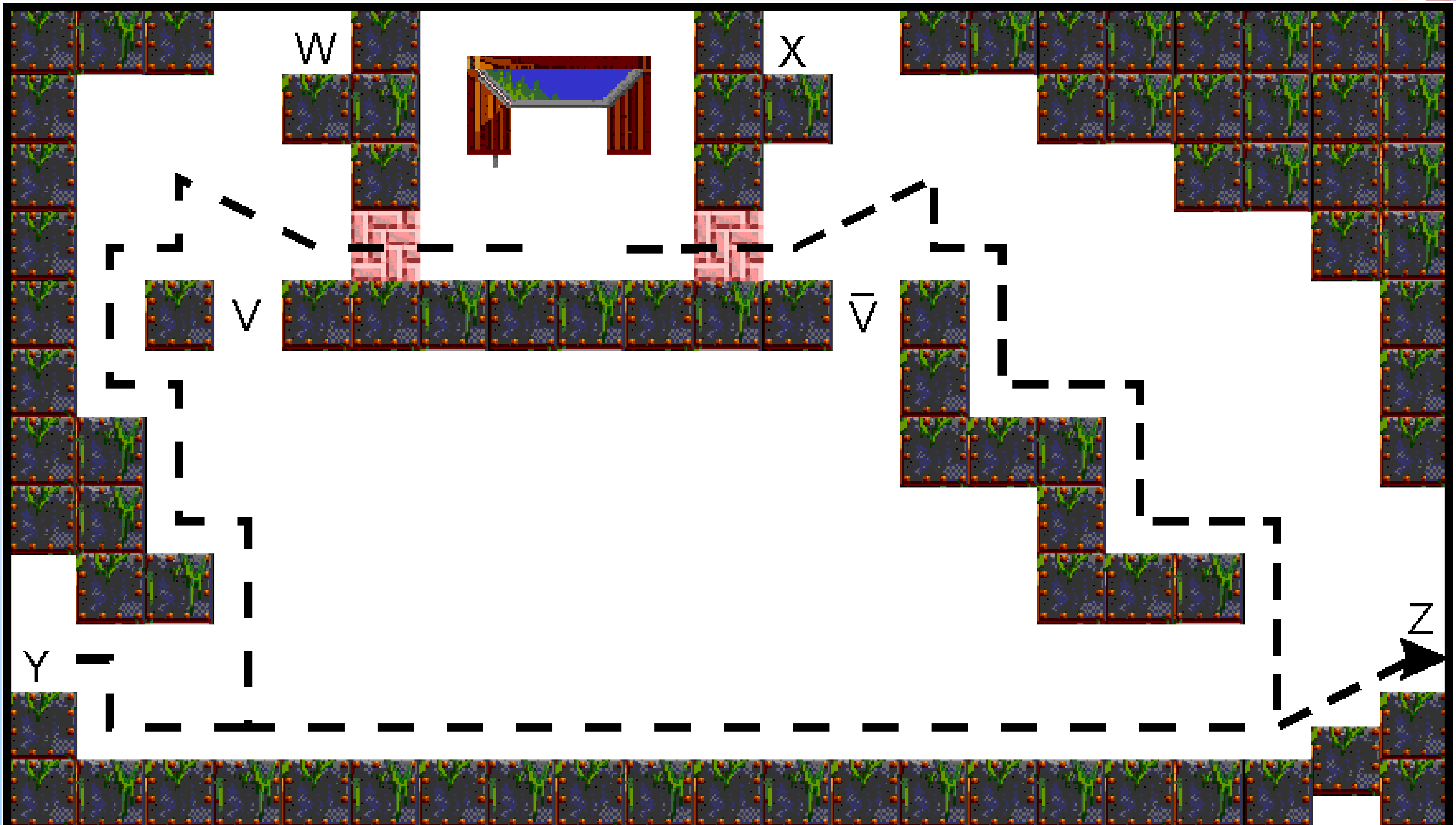
- Three ways out, one for each literal in the clause
- Only way out is for the Lemming inside to dig out.

Variable Gadget

- Only way out is for Lemming to bash one door, build a bridge over one of the gaps.

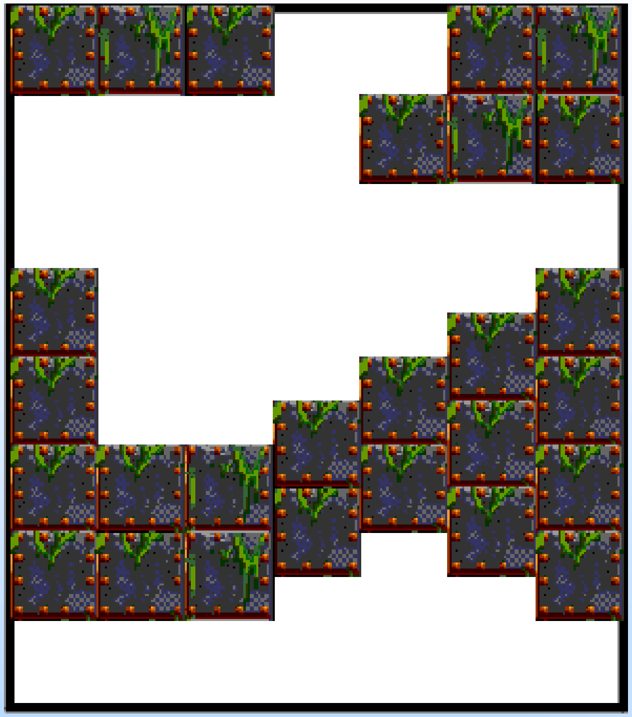
Variable Gadget

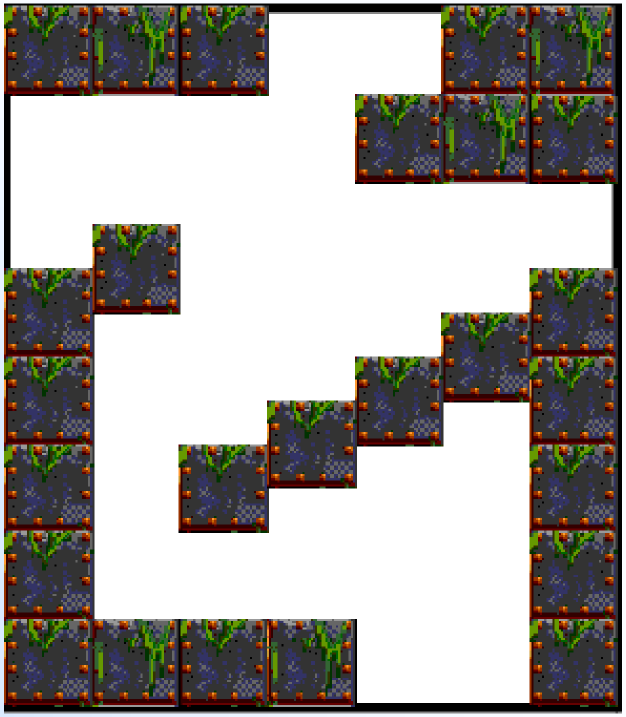


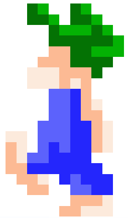


- Only way out is for Lemming to bash one door, build a bridge over one of the gaps.

Wiring





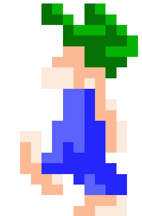


Junction forces lemming out to the right

Wire lets paths cross

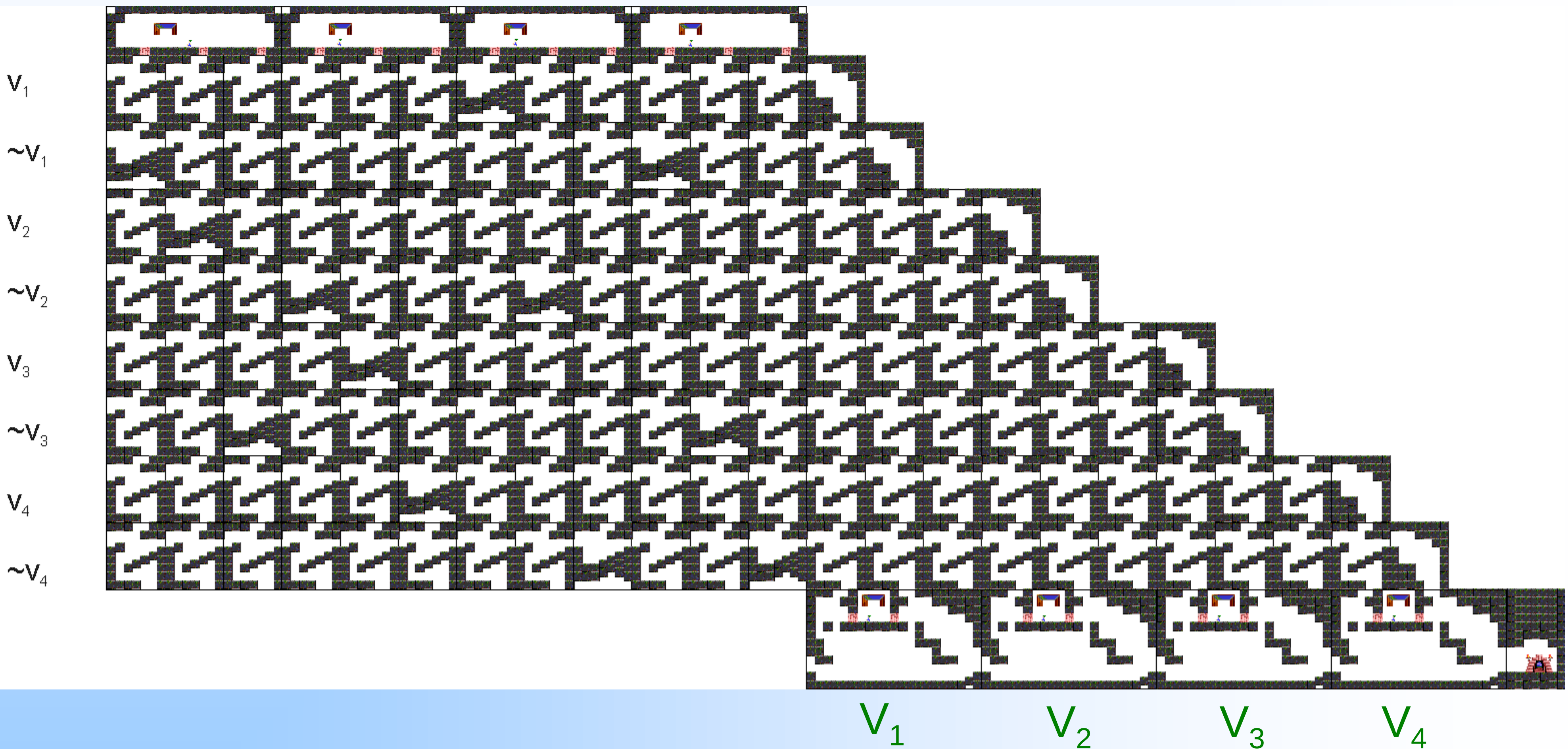
We will restrict number of skills availble so there are none spare to change paths

Putting it together



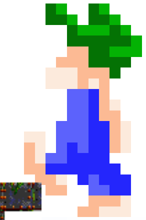
- Build a "routing grid": put a bunch of clause gadgets at the top, and a bunch of variable gadgets at the bottom right leading to the exit.
- Inside the grid, have one column for each clause literal ($3m$ columns in total), and one row for each variable and its negation ($2n$ rows).
- Put a junction in position $[3i + j, 2k]$ if j'th literal in i'th clause is x_k
- Put a junction in position $[3i + j, 2k+1]$ if j'th literal in i'th clause is $\sim x_k$

Example



$(\sim V_1 \vee V_2 \vee \sim V_3)$? $(\sim V_2 \vee V_3 \vee V_4)$? $(V_1 \vee \sim V_2 \vee \sim V_4)$? $(\sim V_1 \vee \sim V_3 \vee \sim V_4)$

Detail of Example

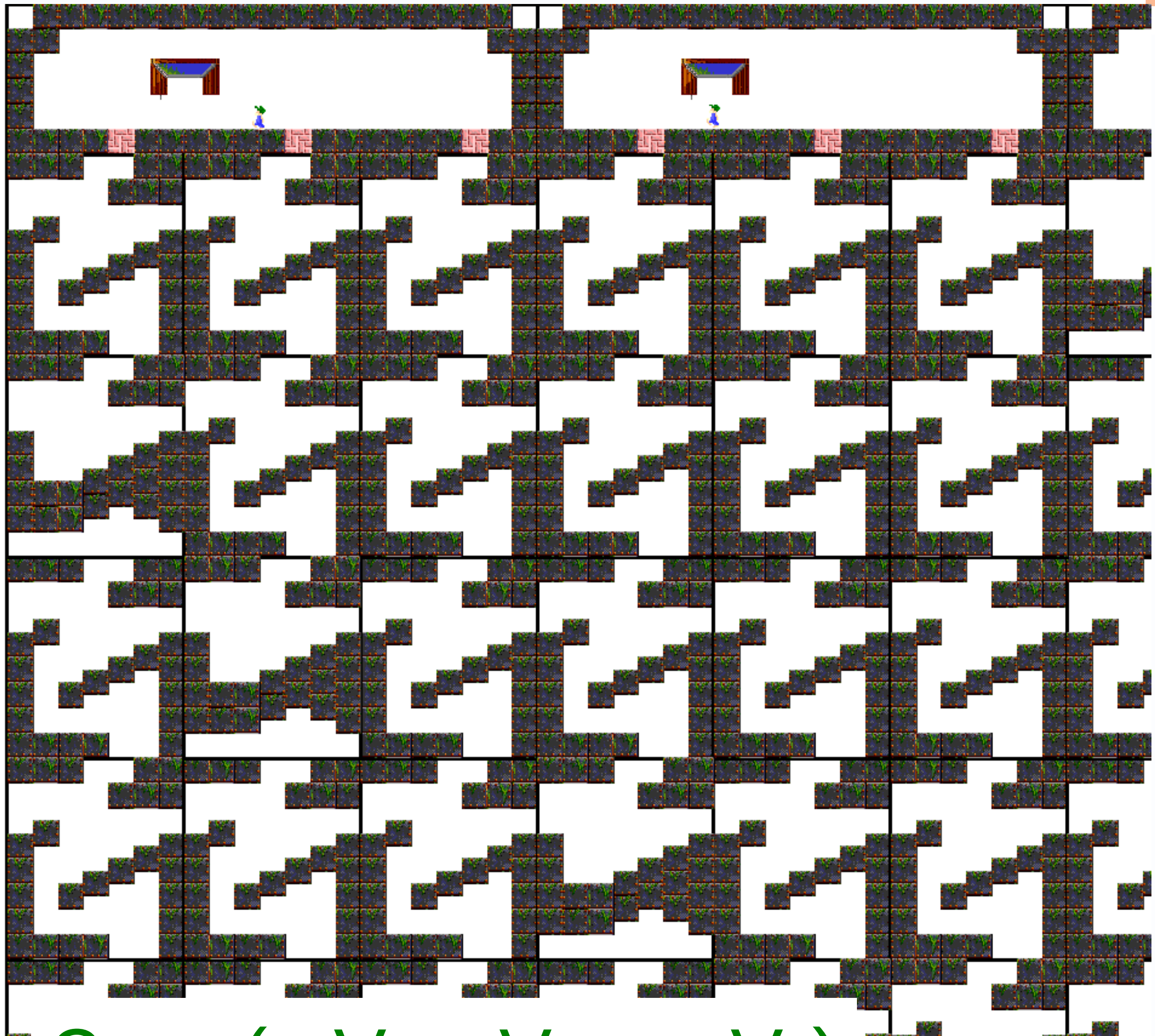


V_1

$\sim V_1$

V_2

$\sim V_2$



$$C_1 = (\sim V_1 \lor V_2 \lor \sim V_3)$$

More Detail

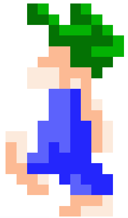
V_3

$\sim V_3$

V_4

$\sim V_4$

Proving the theorem



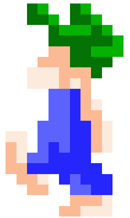
- To prove the theorem requires some case analysis and arguments.
- Need to argue that every solution to the lemmings level is a satisfying assignment to the 3SAT instance and vice-versa
- No details here, it's mostly straightforward... So

Lemmings is NP-Hard

Since the certificate can be checked in poly-time:

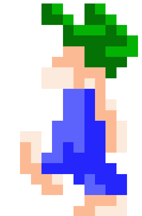
Lemmings is NP-Complete

Other variations



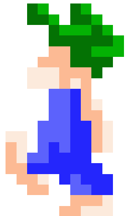
- OK, so we know Lemmings is NP-Complete, is that it?
- In the transformation, we used only temporary skills (bashers, diggers and builders) -- what about Lemmings with other skills?
- In fact, if we only have permanent skills, then the problem is decidable in polynomial time.

Decidable Lemmings



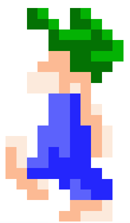
- Model the game board as a graph, G.
- Each location is represented by 4x2 nodes:
  4 corresponding to a lemming with no skills, with climbing, with floating, and with both.
  2 corresponding to facing left or facing right
- For each node, we know what node the lemming will go to. (Special node for "dead").
- We can also put edges corresponding to giving a lemming a certain skill.

A Simple Graph Problem



- Since the board does not change during play (no temporary skills to change the board), G is static, the problem reduces to reachability problems on G
- Still a little fiddly, since we have to choose how best to allocate the skills we do have.
  - Remove all lemmings that reach the exit unaided.
  - Then see how many exit with only climbing or only floating.
  - Then (after some calculations) see how many of the remainder make it out if given both skills.

Back To Hardness

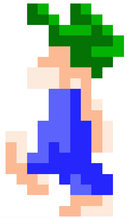


- OK, now we know that Lemmings is NP-Hard, and under restrictions, it is decidable. Are we done now?

- Not quite: the hardness proof is a little unsatisfying since it needs lot of entrances, and lots of Lemmings.

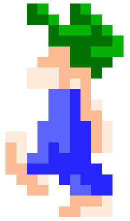
- What if there is only one Lemming? Is the game still NP-Hard?

Hardness of 1-Lemmings

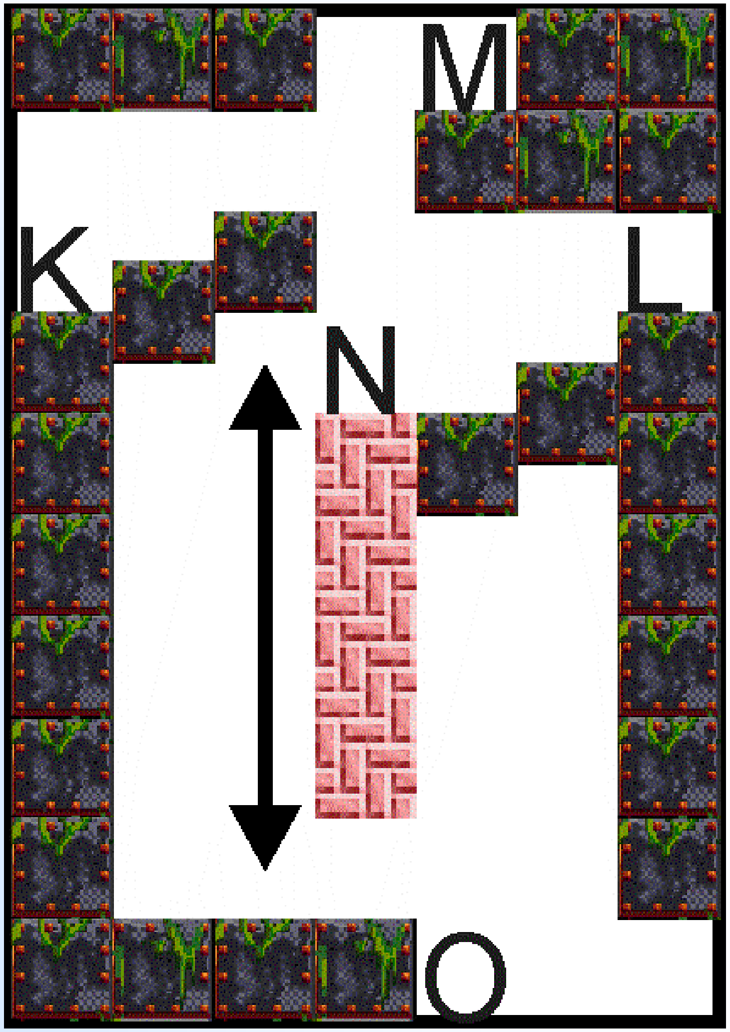


- Recent result (with Mike Paterson): yes, 1-Lemmings is also NP-Hard.
- (This supersedes the previous hardness result, but it's more detailed and requires more gadgets).
- This shows it's hard to approximate the number of Lemmings that can be saved, up to any factor.

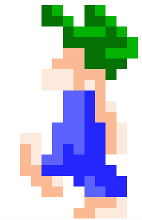
Main Gadget



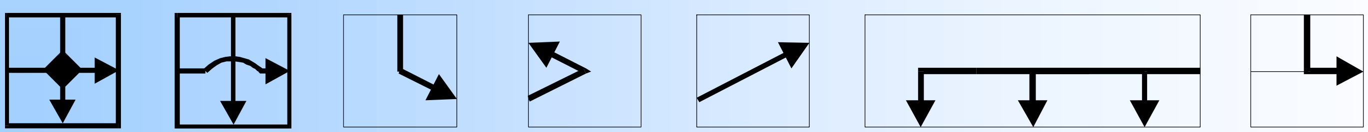
- The main new gadget is the one-way: if the Lemming has come down this gadget, then it cannot later go across it.
- First, the Lemming is sent down vertically through all one-ways corresponding to a variable.
- Then it must go left to right through clauses.



Wiring

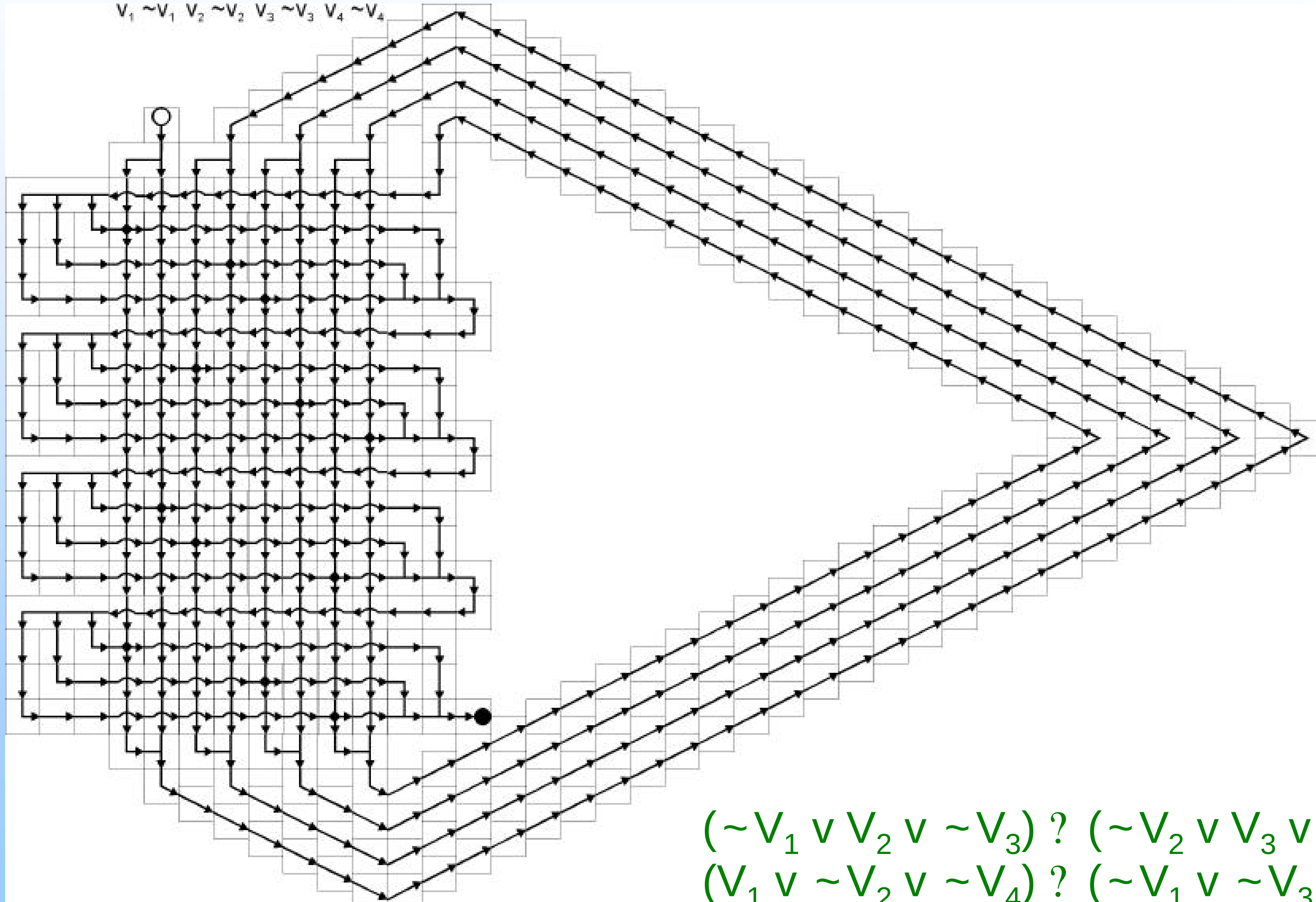


- We also need to loop the lemming back to the top after each vertical descent (setting a variable), and from left to right after each horizontal crossing (satisfying a clause)
- This requires some extra pieces of wiring, but nothing too difficult
- To ease presentation, represent with icons:



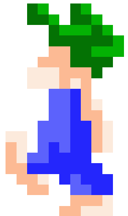
One way wire Clause Junction

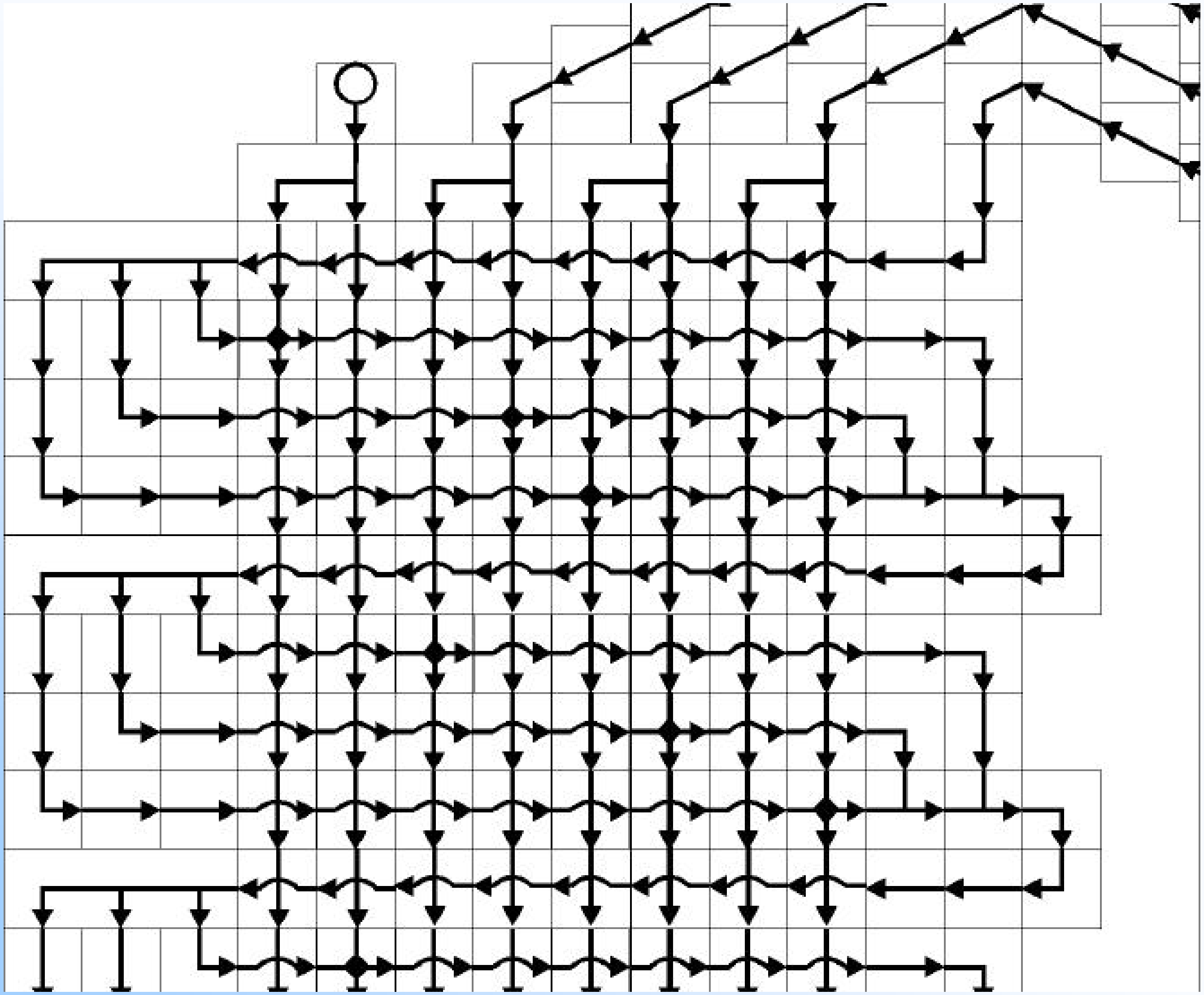
Example 2



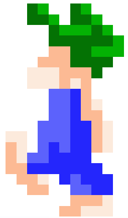
$(\sim V_1 \text{ v } V_2 \text{ v } \sim V_3)$? $(\sim V_2 \text{ v } V_3 \text{ v } V_4)$?
$(V_1 \text{ v } \sim V_2 \text{ v } \sim V_4)$? $(\sim V_1 \text{ v } \sim V_3 \text{ v } \sim V_4)$

Close up of Example



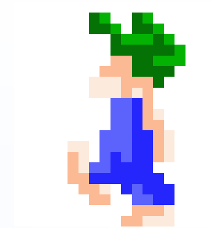


1-Lemmings is NP-Hard



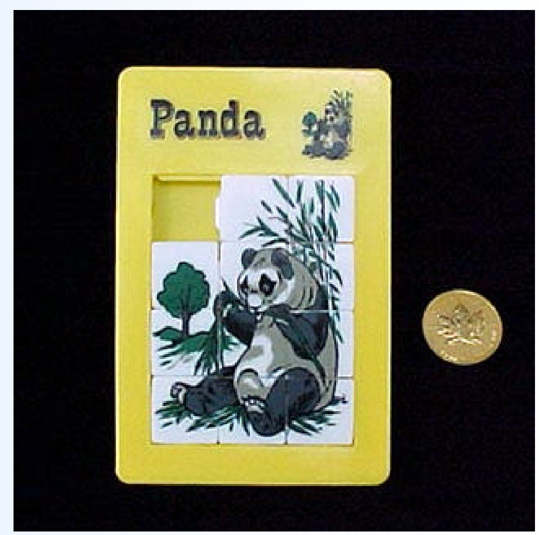
- As before, must argue that every path to the exit corresponds to a satisfying assignment to 3SAT and vice-versa
- Some details left to fill in, but the main idea is there
- So, Lemmings is NP-Hard with 1 Lemming
- Hence, hard to approximate how many lemmings can be saved on any level…

Conclusions

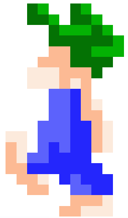


- Now we know why Lemmings was so tough...
- In the real game, most levels are not so repetitive
- But, levels with only floaters and climbers are usually easier.

Is there something deeper here: are most 'good' puzzle games NP-hard? Are there many good puzzles that are known to be in P? (eg sliding block puzzles, 14-15)



Open Problems



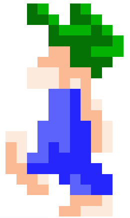
- Is Lemmings P-Space Complete? Can it encode Turing machines? (details: need to encode bits - might be able to do this using bridges, which can be destroyed).

- Are there weaker restrictions under which it is in P?

- What about other games? Solitaire, FreeCell, Doom, etc.? Is your favorite game NP-Hard? Or weaker: does it encode CFG languages?

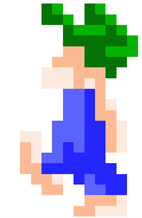
- Does it make sense to talk about the complexity of certain games?

Answer to puzzle 1



- In "toe-tac-tic" the first player can always force a draw.
- She takes the center square, and then her tactic is to "mirror" every move the second player makes.
- Then the only way she would make a line is if the second player already has.

Answer to puzzle 2



Write S as a magic square.

Every subset of S that sums to 15 is a row in the magic square.

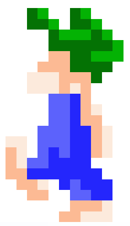
Each player taking an item from S = putting a marker on that number

6	1	8
7	5	3
2	9	4

Game is won when they have three in a row.

So game is equivalent (isomorphic) to Tic-Tac-Toe

Super Mario?



```
#########################################################################
# RSG-SMB-TAB-1.1                                                        #
##########################################################################
How to Win "Super Mario Bros"                Nintendo Entertainment System
WORLD 1 - LEVEL 1
                               +----------------------------------+
  Key:     < = Left            |                                  |
           > = Right           |    ^                             |
           ^ = Up              |  <   >                   O   O   |
           v = Down            |    v    select  start     B   A  |
           B = B button        |                                  |
           A = A button        +----------------------------------+

< ------------------------------------------------------------------------
> 000000000000000000000000000000000000000000000000000000000000000000000000
^ ------------------------------------------------------------------------
v ------------------------------------------------------------------------
B --------------------------0000000000000000000000000000000000000000000000
A ------------------------------------------------------------------------


< ------------------------------------------------------------------------
> 000000000000000000000000000000000000000000000000000000000000000000000000
^ ------------------------------------------------------------------------
v ------------------------------------------------------------------------
B 000000000000000000000000000000000000000000000000000000000000000000000000
A ------------------------------------------------------------------------
```