Celeste is NP-Complete

Complexity Analysis

So what is Complexity Theory?

- Computational complexity theory focuses on classifying computational problems according to their resource usage, and relating these classes to each other.

- A problem is regarded as inherently difficult if its solution requires significant resources, whatever the algorithm used.

- The theory formalizes this intuition, by introducing mathematical models of computation to study these problems and quantifying their computational complexity, i.e., the amount of resources needed to solve them, such as time and storage

P and NP Classes

Unit of time: Number of steps taken by a Turing machine.

P: Problems solvable in polynomial(of size of input) time.

NP: Problems whose solution can be verified in polynomial time given a witness.. All the problems in P are by default in NP.

NP-Hard: Problems which are at least as hard as all the NP problems. Algorithm to solve this problem can be converted into algorithms to solve any other NP problem in polynomial time. The conversion is called a reduction.

NP-Hardness

Example of a Reduction (From Wikipedia):

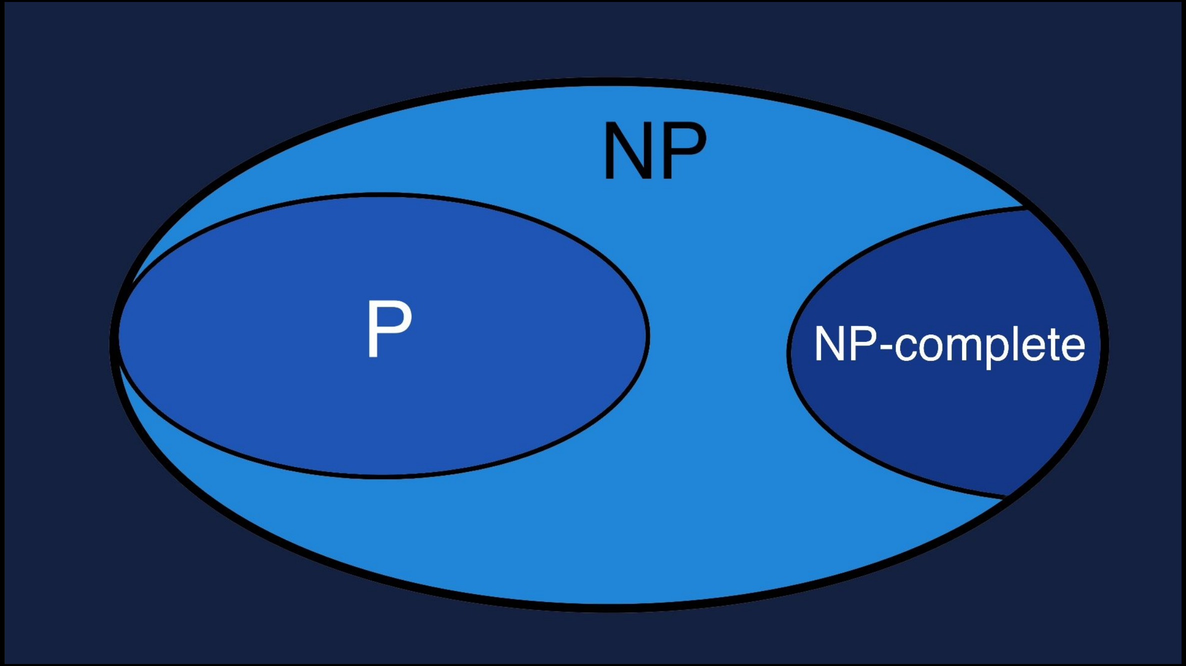
If you have a fast algorithm to square numbers, you can use it to multiply 2 numbers, the conversion will be:

$$a \times b = \frac{\left((a+b)^2 - a^2 - b^2\right)}{2}$$

NP-Complete: A problem which is NP and is NP-Hard. Subclass of NP which contains the hardest problems of NP.

As Grothendieck once taught us, objects aren't of great importance; It is the relationship between them that are.

Presumed Hierarchy



Relations between classes

Recognizing puzzles harder than solving them? P vs NP

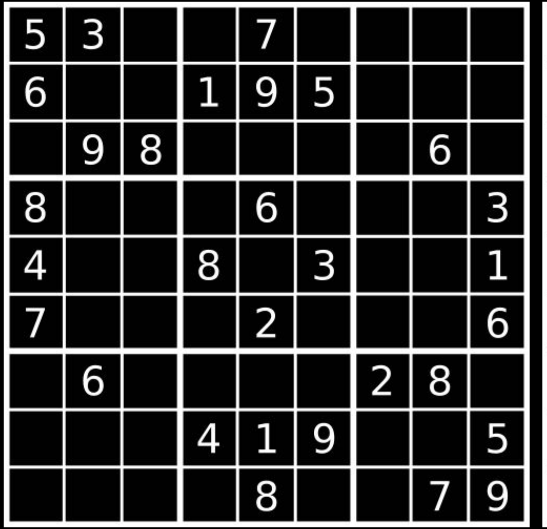
"If P=NP, then the world would be a profoundly different place than we usually assume it to be. There would be no special value in 'creative leaps,' no fundamental gap between solving a problem and recognizing the solution once it's found."

- Scott Aaronson

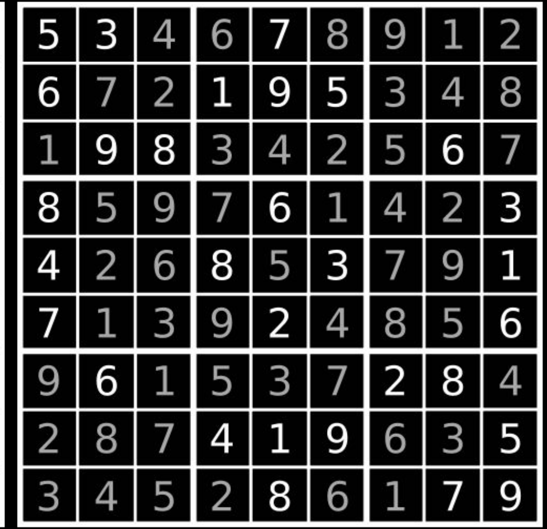
"One day I will find the right words, and they will be simple."

-Jack Kerouac

Sudoku



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9



5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

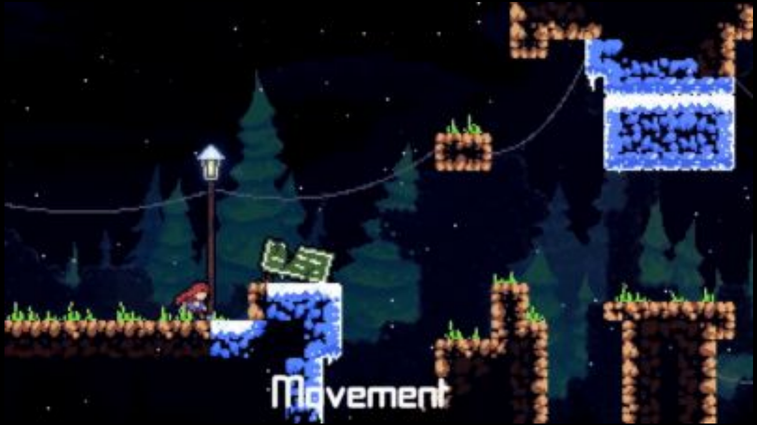
Solving Sudoku is much harder than verifying if the solution is correct.

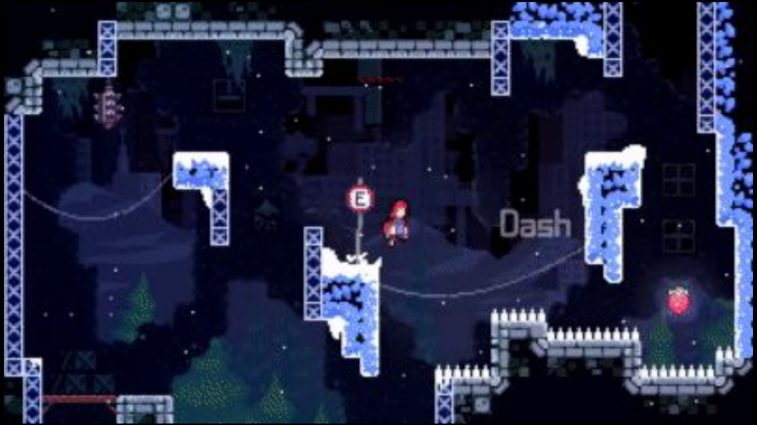
Proving Celeste to be NP-Complete

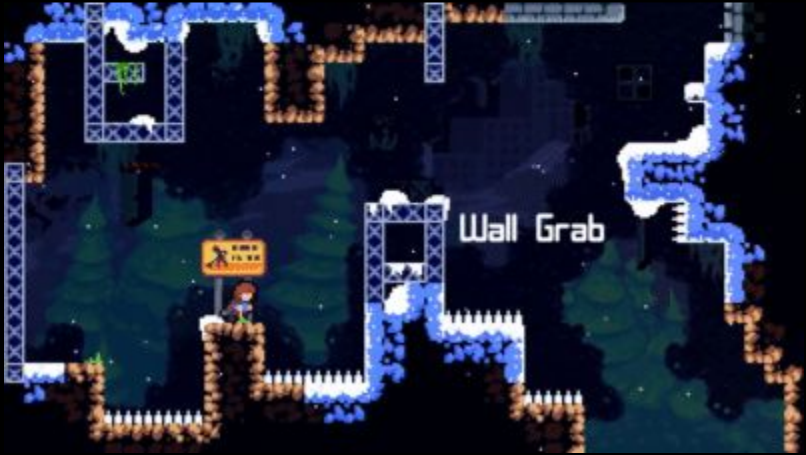
"Gaming is Hard, but someone must ~~do~~ prove it."

- A plagiarised quote by Kunwar

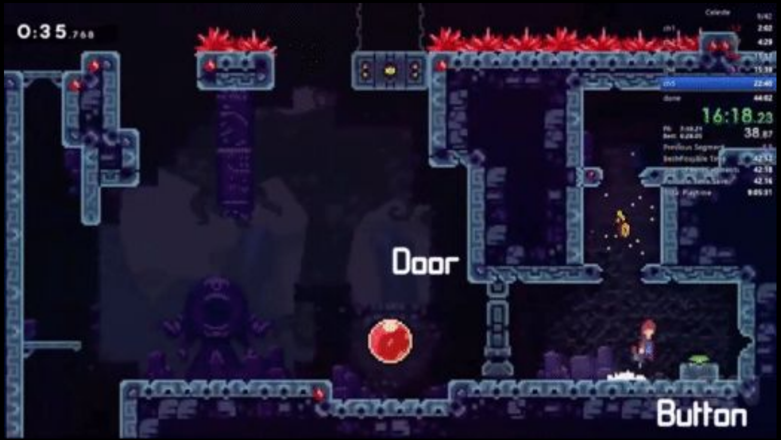
The Game's Mechanics

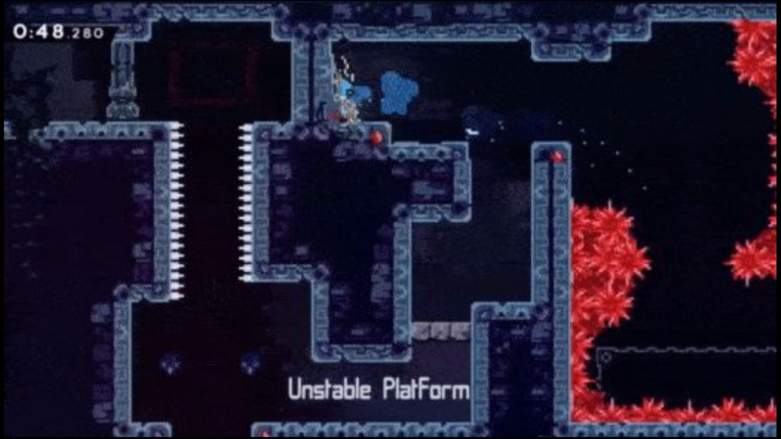


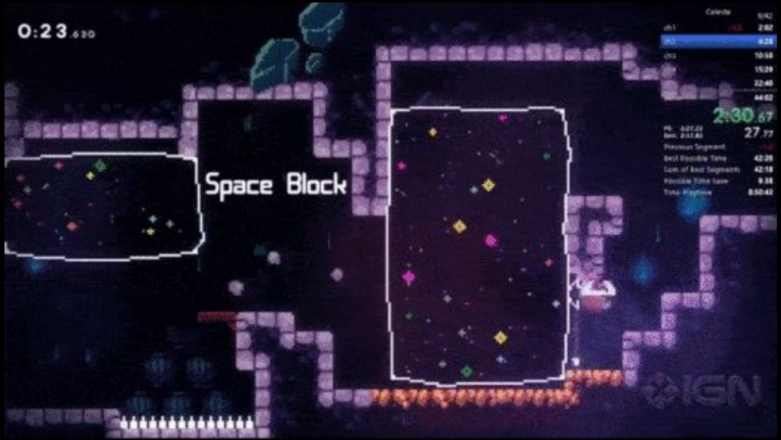




The Game's Mechanics



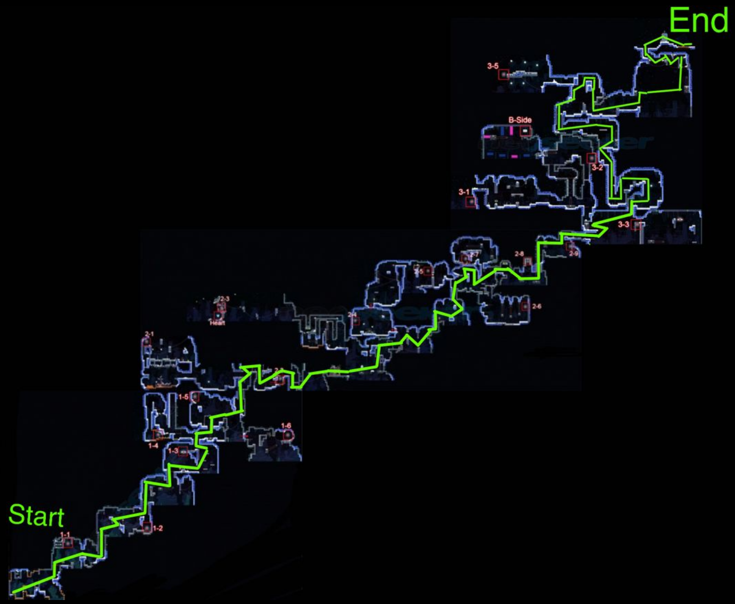




Celeste is NP

Given a level and a path, that is the moves required to reach the endpoint, You can verify if the path is correct just by applying those moves.

The moves are polynomial, why? We repeat the sections only after visiting other sections. Since the number of sections themselves are polynomial, we can only have polynomial moves before we complete the level. .



This clearly implies that the **game is NP.** For example, for the 1st level, we can map the path from start to end as seen below

Now since we have proven that Celeste is NP. We can try to prove that it is NP-Hard, essentially proving that it is NP-Complete.

Framework

3SAT

Boolean Satisfiability of a 3 Conjunctive Normal Form.

What is Boolean Satisfiability? Given a Boolean formula, Does a substitution for the variables exist such that the formula evaluates to True.

3 Conjunctive Normal form: AND of clauses which contain 3 literals.

Example:

$$(x1 \lor x2 \lor !x3) \land (x2 \lor x3 \lor x1)$$

Satisfiable substitution:

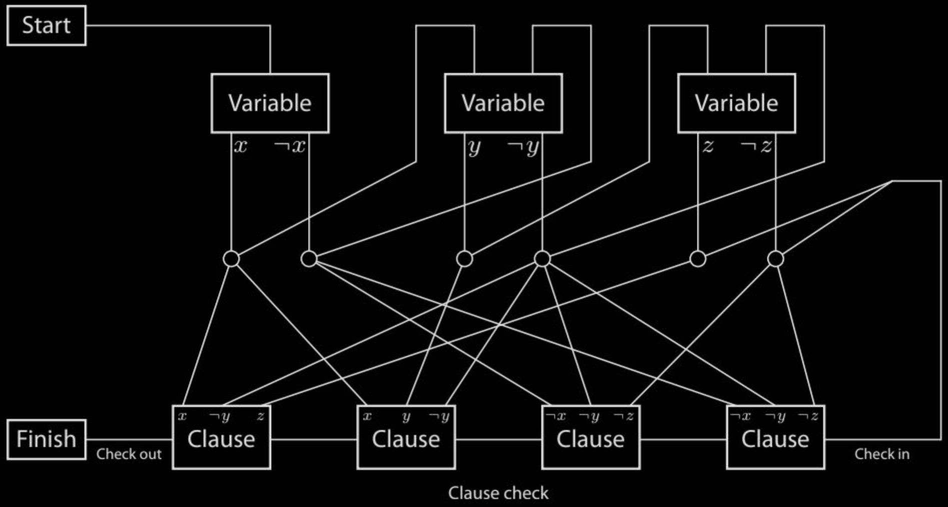
$x1 = 1$

$x2 = 1$

$x3 = 1$

3-SAT Reductions

To prove the NP-Hardness of Celeste, we here describe a framework for reducing 3SAT to a 2-D platform game. This framework is based on https://arxiv.org/pdf/1203.1895.pdf. Using this framework in hand, we can prove the hardness of games by just constructing the necessary gadgets.



3-SAT Reductions

Required Gadgets:

Start and Finish: The start and end gadgets contain the spawn point and the end goal respectively.

Variable: Each variable gadget must force the player to make a binary choice (select x or ~x). Once a choice is taken the other choice should not be accessible. Each variable gadget should be accessible from and only from the previous variable gadget is such a way that it is independent of the choice of the previous gadget and going back is not allowed.

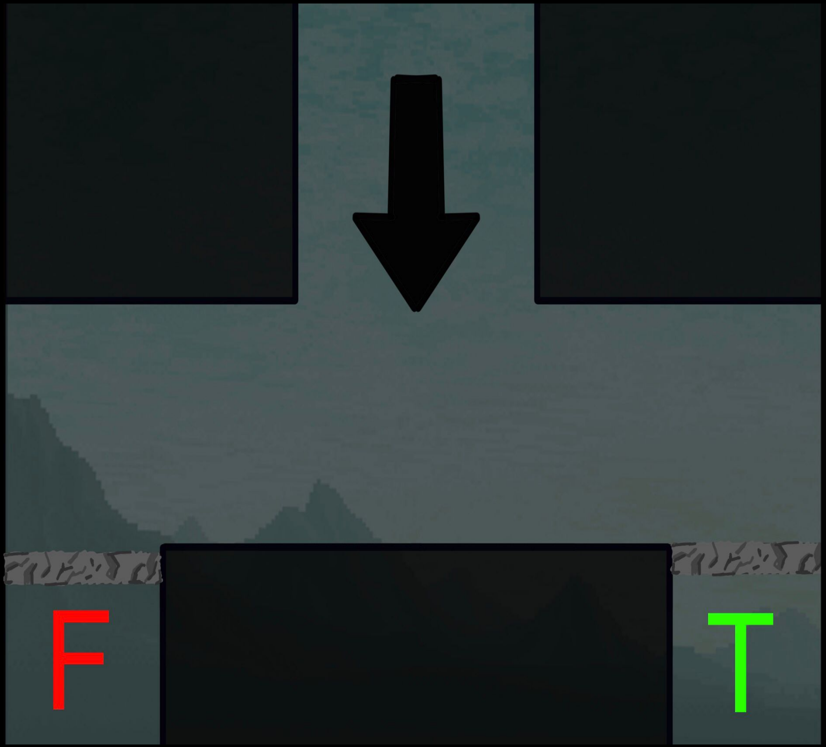
Clause: Each literal in the clause must be connected to the corresponding variable. Also, when the player visits the clause, there should be a way to unlock the corresponding clause.

Check: After all the variables are passed through, all the clauses are run through sequentially. If the clause is unlocked, then the player moves on to the next clause else loses.

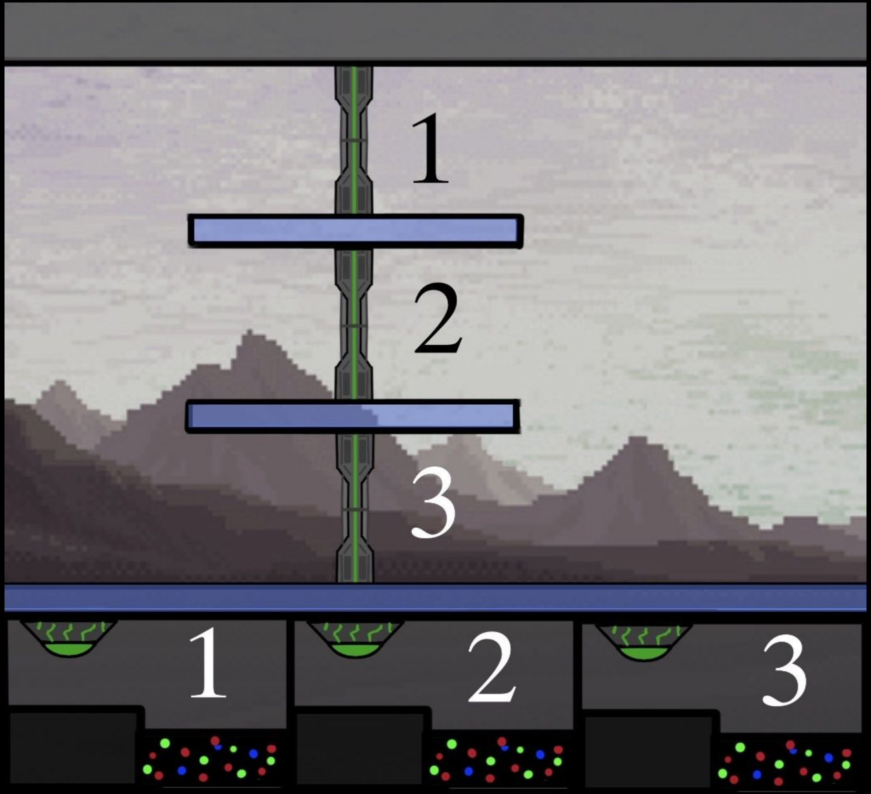
Crossover: The crossover gadget allows passage via two pathways that cross each other. The passage must be such that there is no leakage among them.

Gadgets

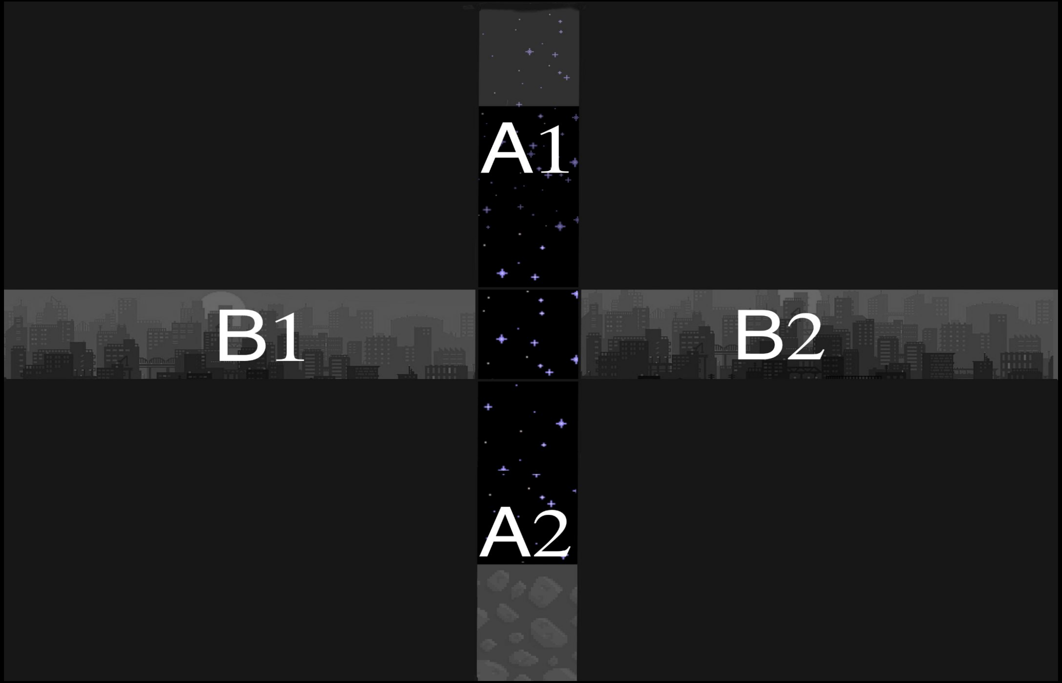
Variable Gadget



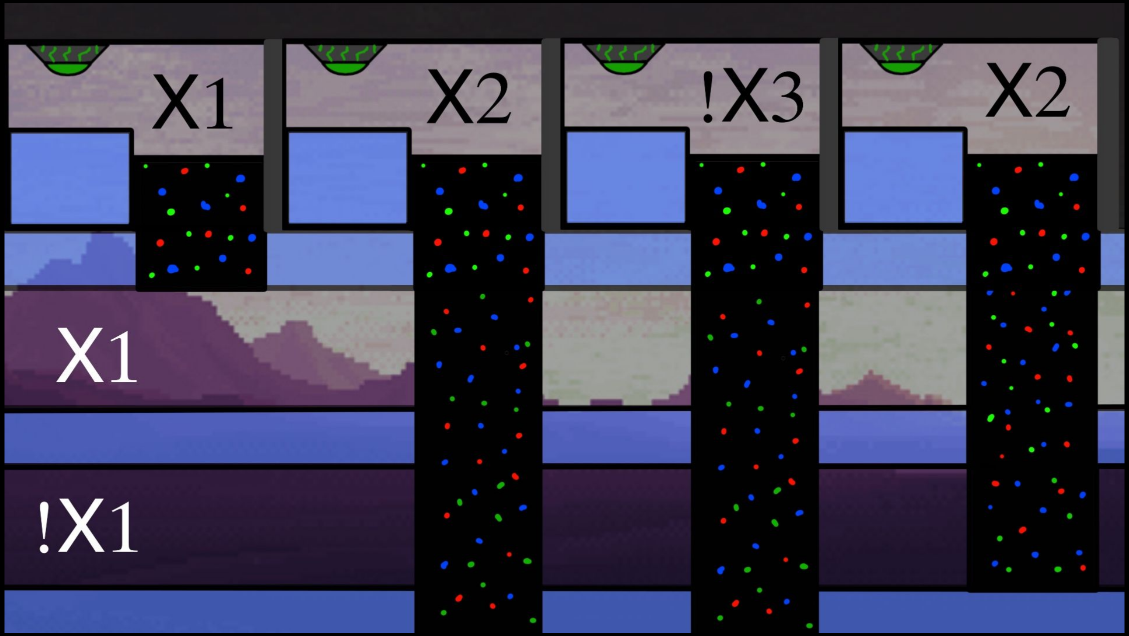
Clause Gadget



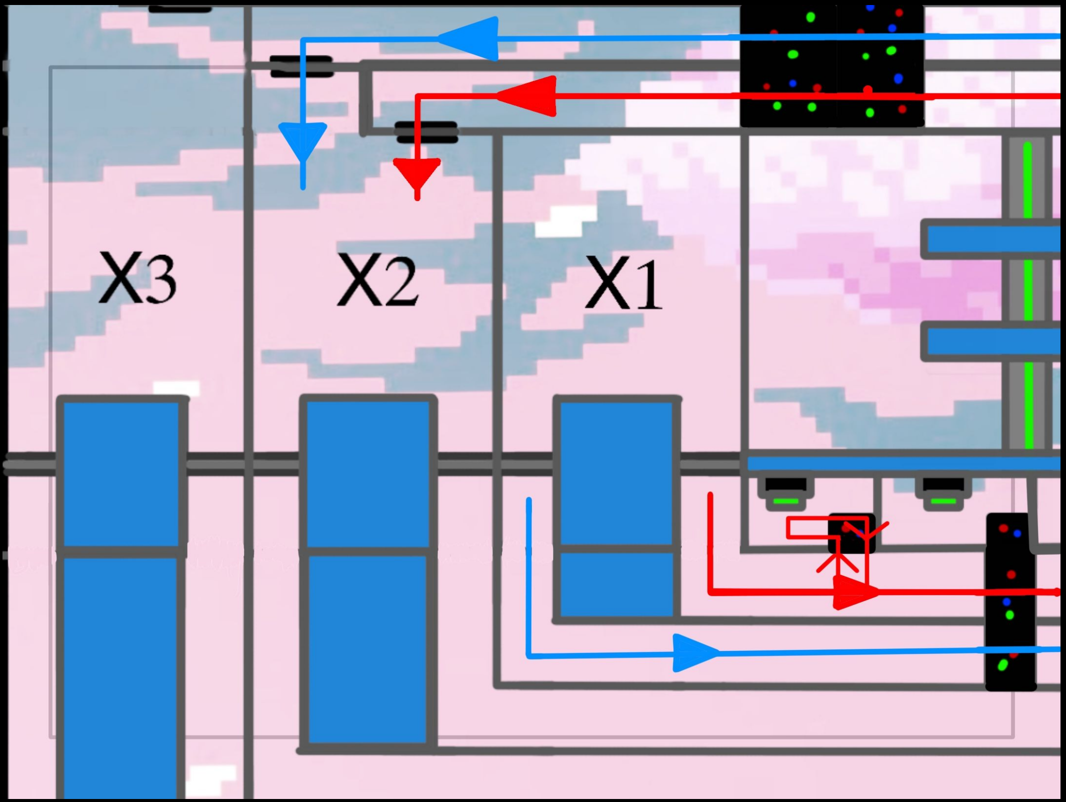
Crossover Gadget



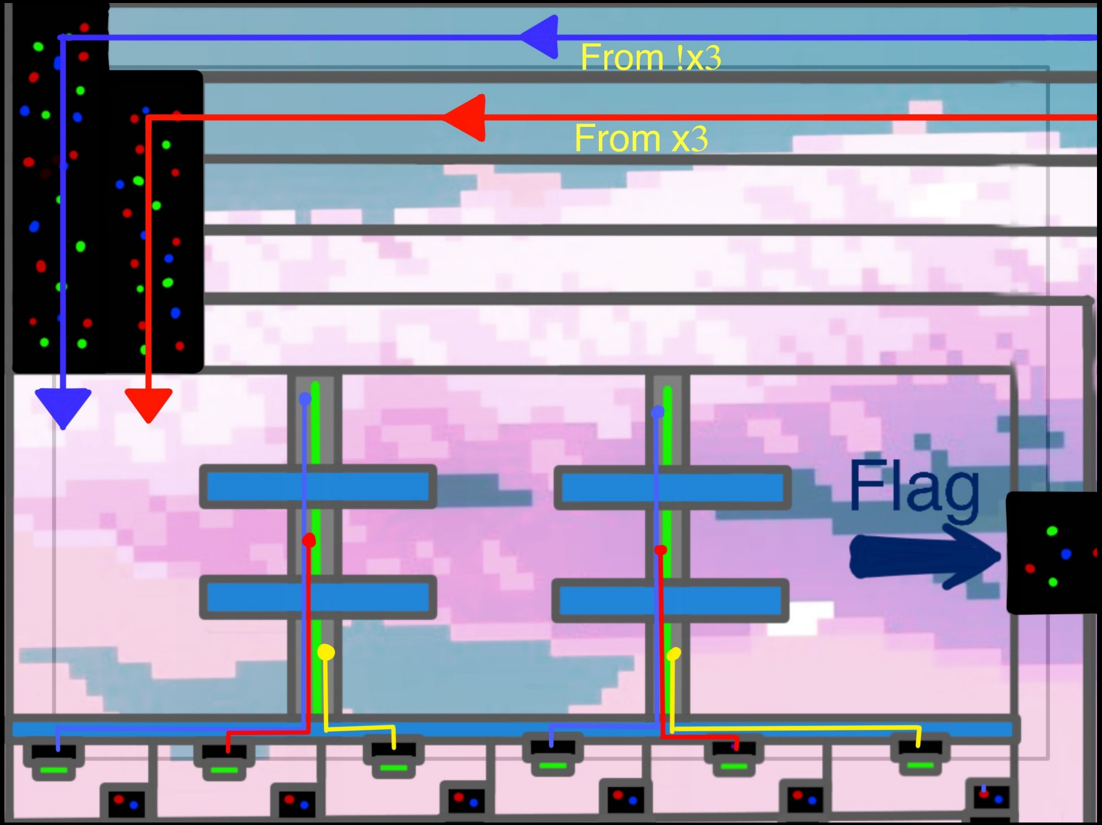
Tunnel Gadget



Transitioning



Final passage

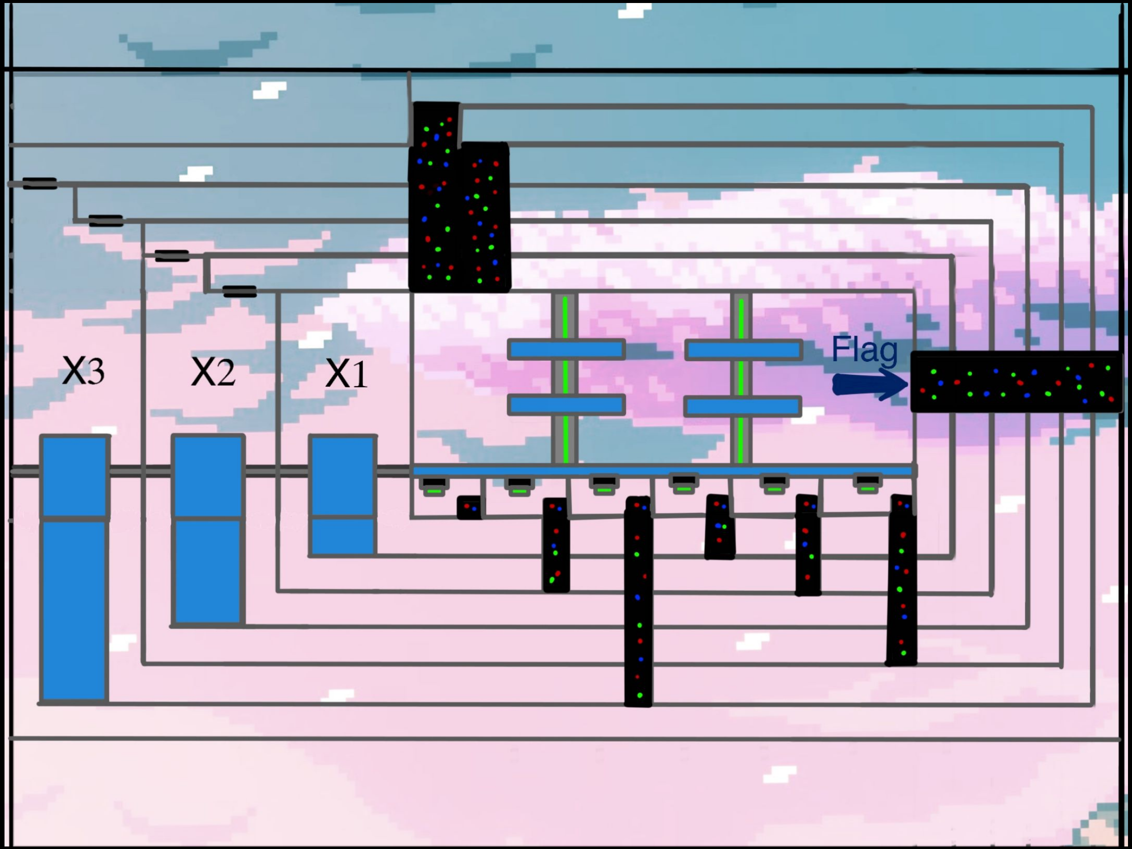


Our Level

We designed and reduced a level in Celeste to the 3CNF boolean expression:

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$$

Hence, we proved that Celeste is at least as hard as 3-SAT.



Substituting into the formula

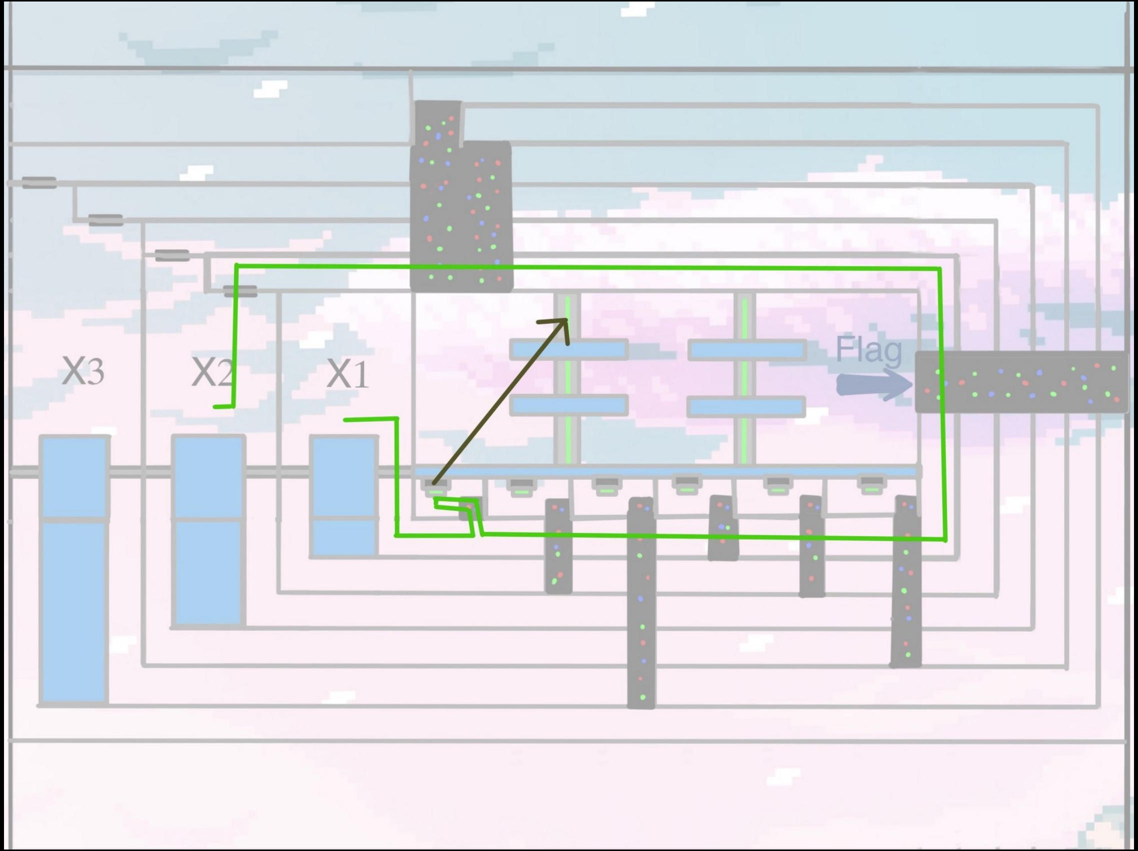
Values:

x_1 = 1

x_2 = 0

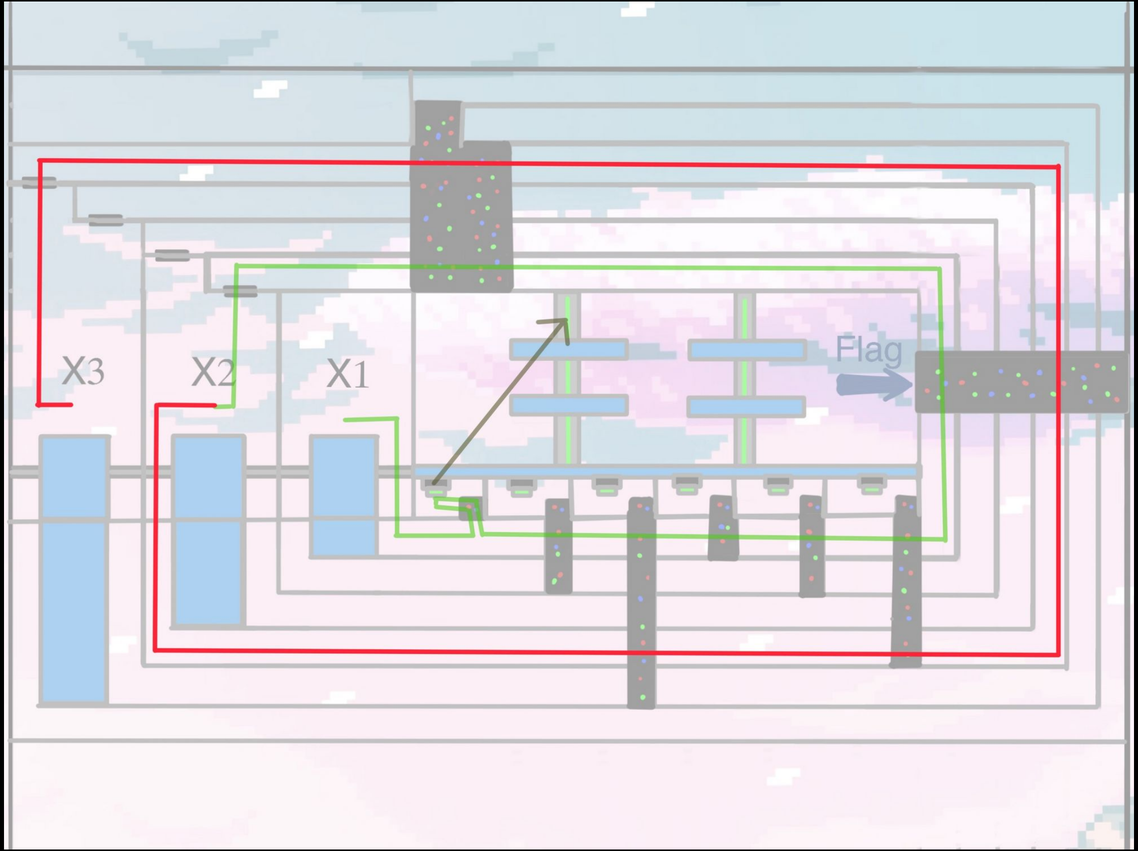
x_3 = 1

x_1 = 1



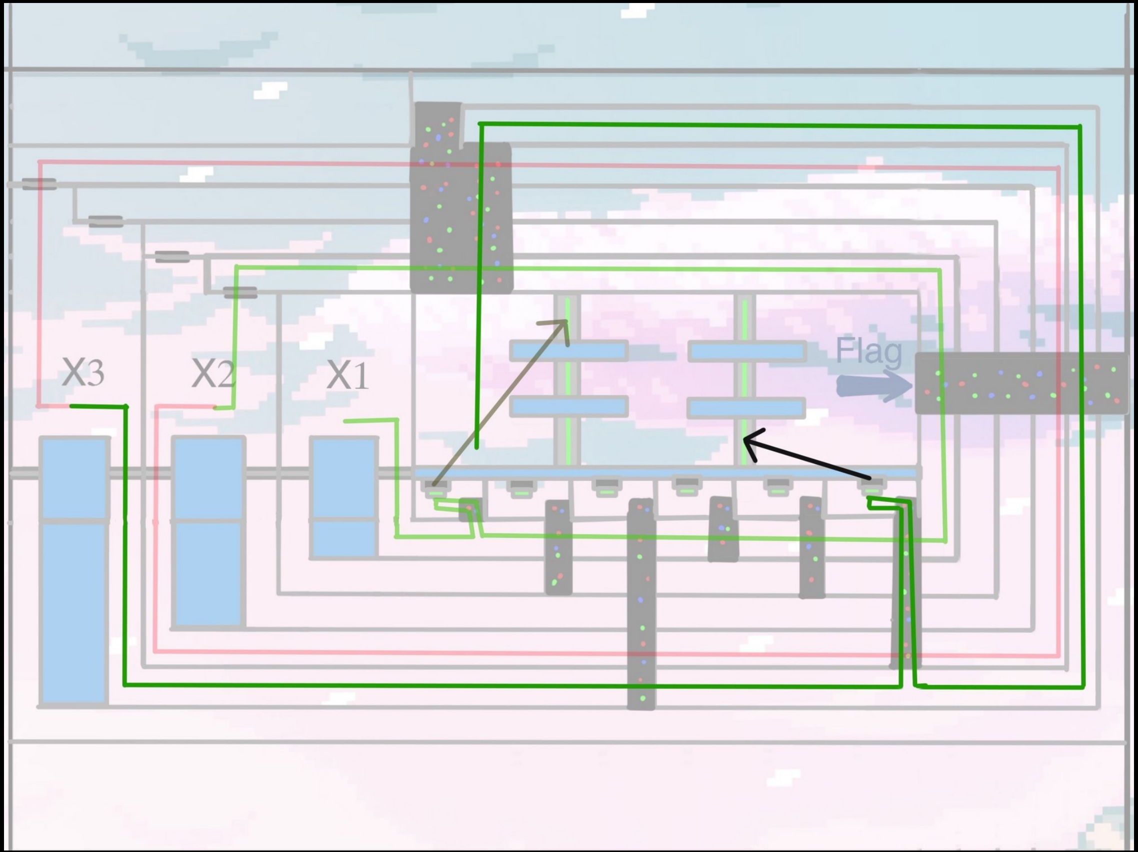
$$(x_1 \lor x_2 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3)$$

x_2 = 0



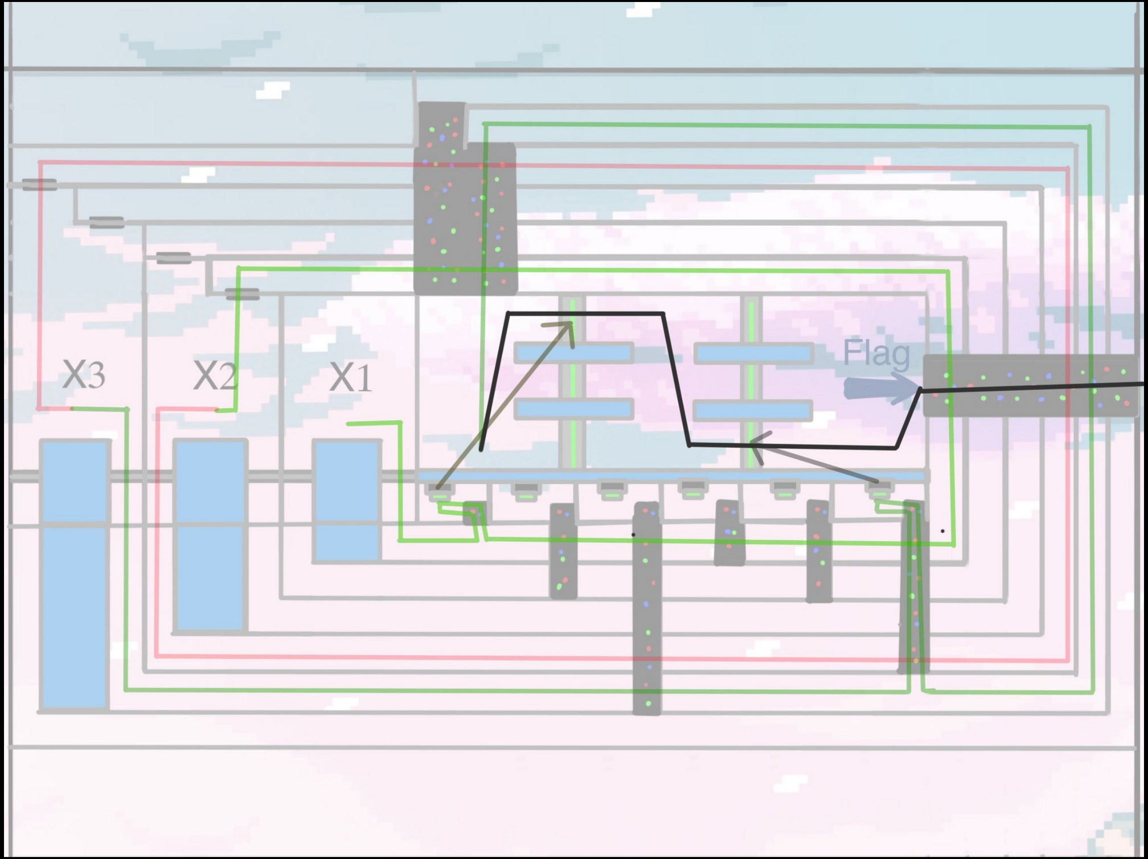
$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$$

x_3 = 1



$$(x_1 \lor x_2 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3)$$

▐ Final Passage ▌



$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$$

Conclusion

- We proved that Celeste is at least as hard as 3SAT, making it NP-Hard.

- In conclusion, the game is both NP and NP-Hard, making it an NP-Complete puzzle.

PSPACE-Complete?

What is PSPACE-Complete?

If a decision problem in PSPACE and every other problem that can be solved PSPACE can be reduced to it in polynomial time , the problem is said to be PSPACE-complete.

This means that the PSPACE-complete are the hardest problems in PSPACE, as solution for this problem can be used to solve any other problem in PSPACE.

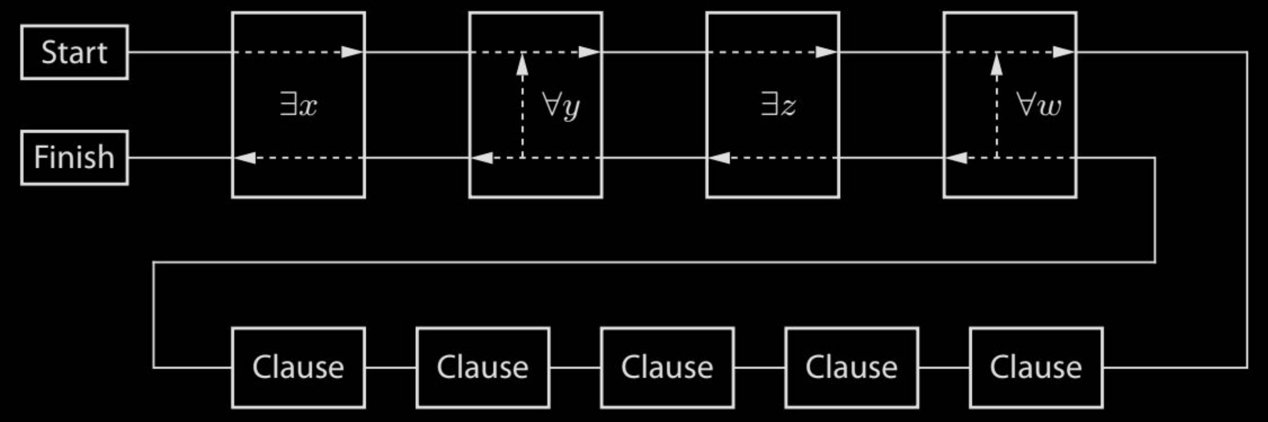
From Savitch's theorem, PSPACE is closed under nondeterminism, ie: PSPACE=NPSPACE. Hence PSPACE-complete are at least as hard NPSPACE problems too.

Additional features that Celeste needs

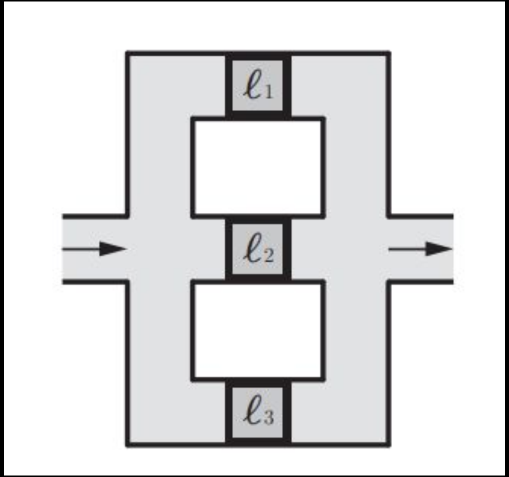
We now make an addition to the game. The door can now be closed using a red button. When the door is open, the red button is deflated and can be activated and when the door is closed, the green button can be activated.

Framework for PSPACE-Hardness

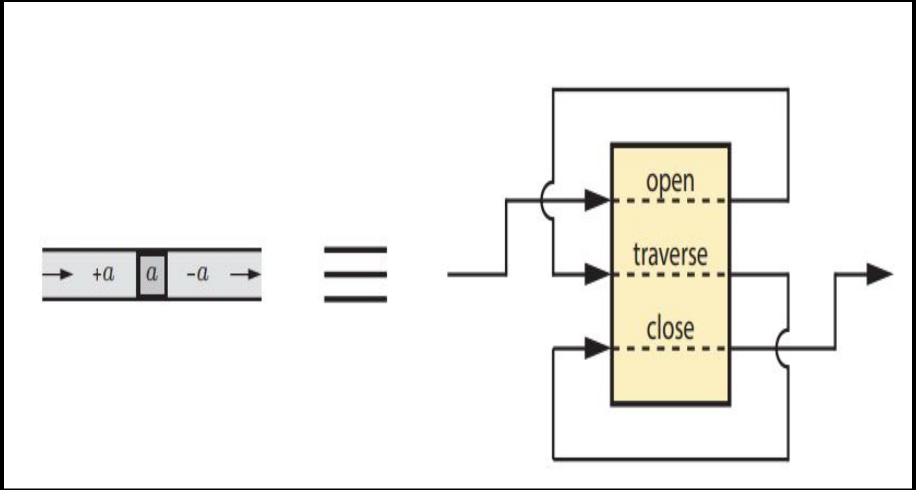
To prove the PSPACE-Hardness of Celeste, we here describe a framework for reducing TQBF to a 2-D platform game. This framework is based on the paper “Classic Nintendo Games are (Computationally) Hard”. Using this framework in hand, we can prove the hardness of games by just constructing the necessary gadgets.



Gadgets for TQBF Framework

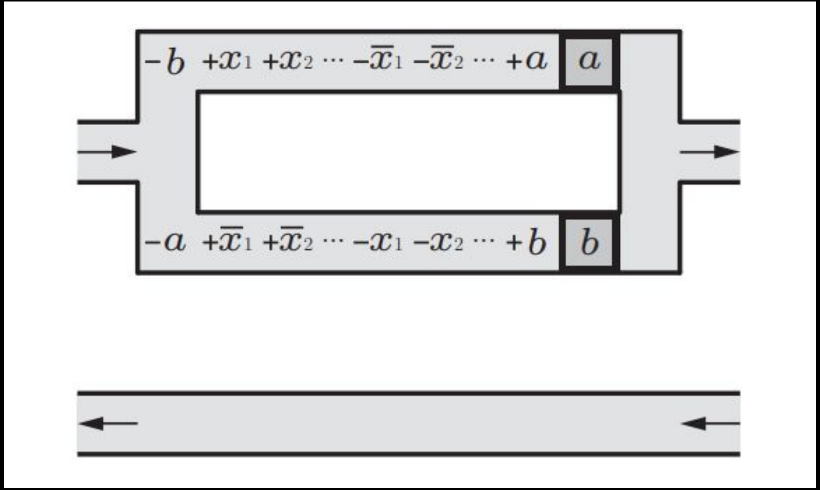


Clause Gadget

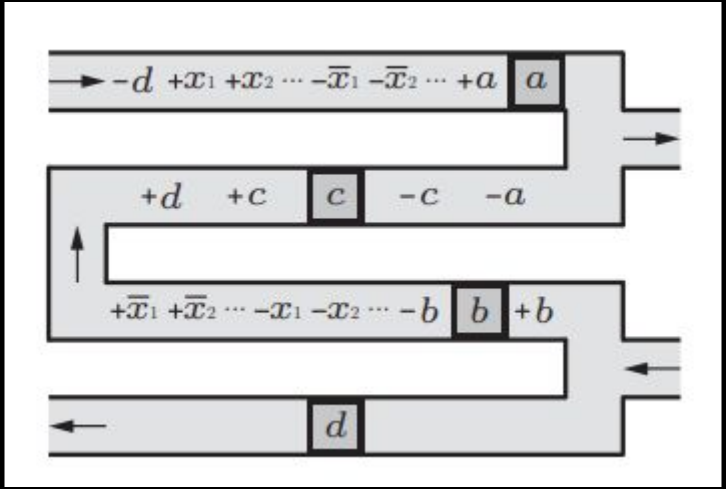


Shorthand notation for tunnel

Gadgets for TQBF Framework



$-b$ $+x_1$ $+x_2$ $\cdots$ $-\overline{x}_1$ $-\overline{x}_2$ $\cdots$ $+a$
a
$-a$ $+\overline{x}_1$ $+\overline{x}_2$ $\cdots$ $-x_1$ $-x_2$ $\cdots$ $+b$
b



$-d$ $+x_1$ $+x_2$ $\cdots$ $-\overline{x}_1$ $-\overline{x}_2$ $\cdots$ $+a$
a
$+d$ $+c$
c
$-c$ $-a$
$+\overline{x}_1$ $+\overline{x}_2$ $\cdots$ $-x_1$ $-x_2$ $\cdots$ $-b$
b
$+b$
d

Framework Traversal

Traversing a quantifier gadget sets the corresponding variable in the clauses. When passing through an existential quantifier gadget, the player can set it according to their choice. For the universal quantifier gadget, the variable is first set to true.

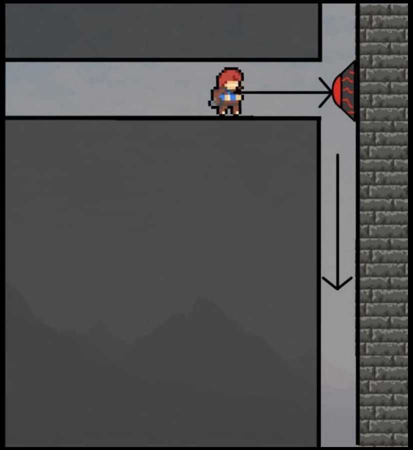
A clause can only be traversed if at least one of the variables is set in it. After traversing all the quantifier gadgets, the player does a clause check and is only able to pass if all the clauses are satisfied. If the player succeeds, they are routed to lower parts of the quantifier gadgets, where they are rerouted to the last universal quantifier in the sequence.

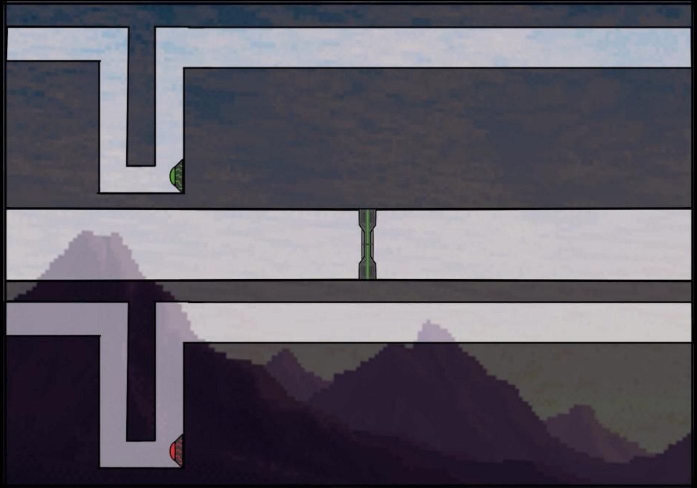
The corresponding variable is then set to False and the clauses are traversed again. This process continues and the player keeps backtracking and trying out all possibilities.

Door Implementation

Implementation of the Doors

For creating a door gadget we first need to create a way to force the player to dash into a button to activate it. We do this using a narrow tunnel and leaving only enough space to pass if the button is pressed.





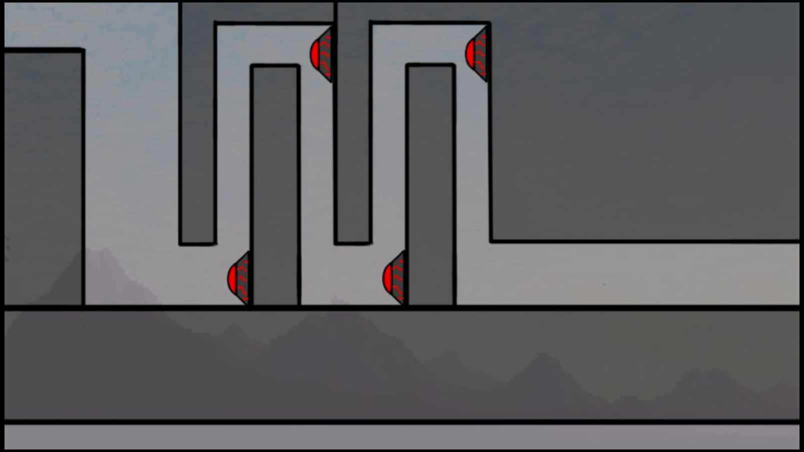
The button above will open the door, and the button below will close the door, and since the path is thin, Madeline will not be able to pass through until the button is pressed.

Tunnel Gadgets

To connect the Variable exit to the Buttons of the Clause, we use a Tunnel gadget.

Below the buttons, there are Tunnels leading from the exits of the variables. The variables have access to the buttons they can set true according to the boolean expression.

For example, if x_1 is chosen to be true, then Madeline gets access to the x_1 tunnel and $\neg x_1$ otherwise. x_1 tunnel has access to buttons that open a door to the clause having x_1.



But what exactly changed?

But what exactly changed?

On adding the close button, we added a requirement to keep track of all the doors that are open. Once a door is opened, it always remained open. Before we had to only keep track of the current state sequentially as all the doors will be opened in a sequence. Knowing that a door is open implied that all the previous doors were opened so as to reach the current door.

But after adding the close button, at any point of the game, all the doors are independent and knowledge of the open state of a door gives us no info about the other doors. So, we have to keep track of all the other doors. This makes the game harder and makes its PSPACE-Hard instead of NP-Hard.

Lack of knowledge about the status of all the doors makes the game PSPACE-Complete

Conclusion

In conclusion, when we add additional feature, the game becomes both PSPACE and PSPACE-Hard, making it an PSPACE-Complete puzzle.

Thank You

Kunwar Zeeshan