# Mazezam is NP-complete

Ben North
ben@redfrontdoor.org

1 September, 2008

## Problem statement

We claim that the game Mazezam is NP-complete, in the sense that any NP problem can be reduced to asking the question 'is there a solution to this mazezam level?' for a particular level constructed (in polynomial time) from the given NP problem.

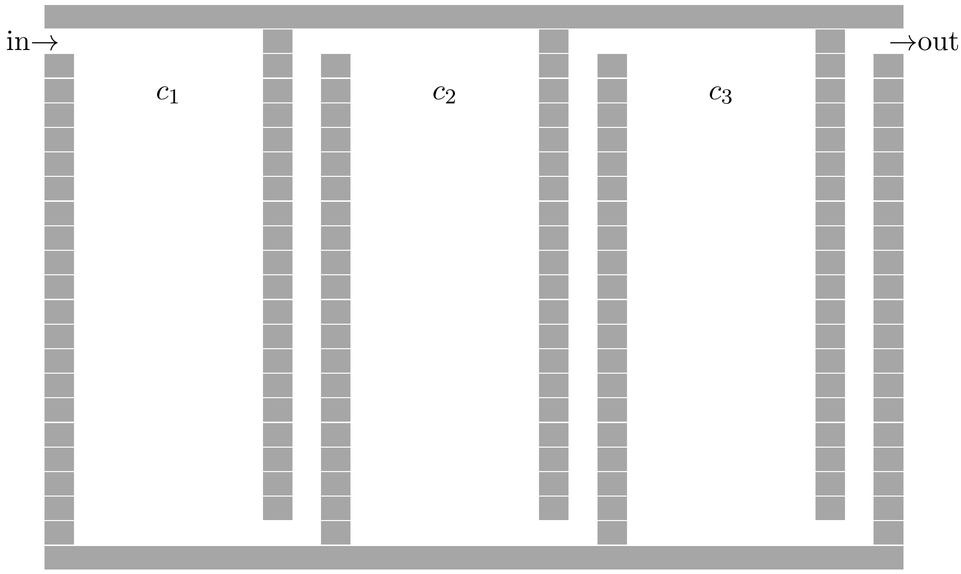
First reduce the NP problem to an equivalent Boolean satisfiability problem, where the Boolean formula is in conjunctive normal form, and each clause has at most three variables. This reduction can be done because the latter problem, '3SAT', is NP-complete. To show NP-completeness of Mazezam, we must be able to reduce a question of the form

> 'Is there an assignment of *True*/*False* to the variables $p_1, p_2, \ldots, p_n$ under which the formula
>
> $$\phi = c_1 \wedge c_2 \wedge \cdots \wedge c_K$$
>
> evaluates to *True*, where each clause $c_k$ is of the form
>
> $$c_k = \ell_{k1} \vee \ell_{k2} \vee \ell_{k3}$$
>
> and each literal $\ell_{km}$ is of one of the forms
>
> $$\ell_{km} = p_i \quad \text{or} \quad \ell_{km} = \neg p_i$$
>
> for some $i$ depending on $k$ and $m$?'

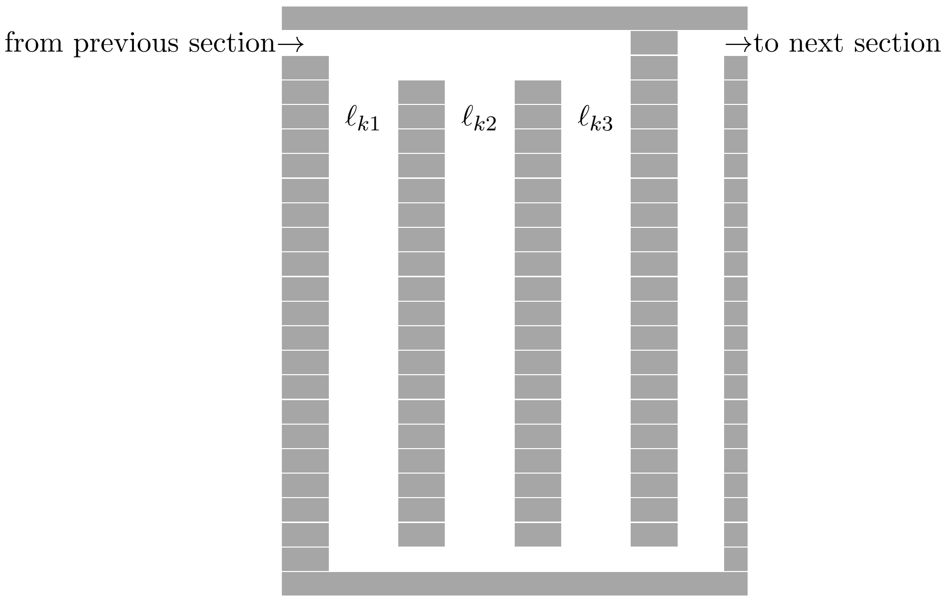
to the question of solubility of a Mazezam level constructed from the Boolean formula $\phi$ in polynomial time.

## Summary of approach

Each clause in $\phi$ will be represented in the Mazezam level by a section which the player must navigate from top to bottom, with a corridor to the right of it which brings the player back up to the top. Being able to navigate a section will correspond to the clause represented by that section evaluating to *True*. The player must navigate all of these sections in sequence, which corresponds to the conjunction of all clauses evaluating to *True*. In this example, there are three clause-sections; the details of what goes in the sections will be filled in shortly:

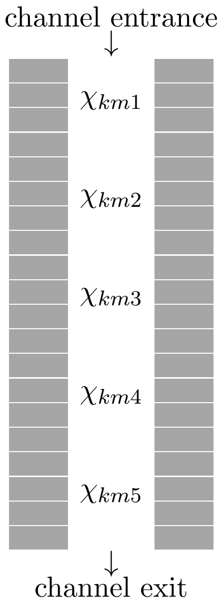



Each clause $c_k$, being a disjunction of three literals, will be represented by three parallel vertical channels, the $m$th of which will be passable iff $\ell_{km}$ evaluates to *True*. The player will be able to pass downwards through the clause's section iff she is able to pass through at least one of the literals' channels. This effects the required disjunction:




Finally, the channel representing $\ell_{km}$ will be made up of one cell $\chi_{kmi}$ per variable in $\phi$. If $\ell_{km} = p_i$, then the cell $\chi_{kmi}$ corresponding to $p_i$ will be passable iff $p_i$ is assigned the value

*True*; if $\ell_{km} = \neg p_i$ , then the cell $\chi_{kmi}$ will be passable iff $p_i$ is assigned the value *False*; if $\ell_{km}$ is of neither of these forms, the cell $\chi_{kmi}$ will be passable whatever the value assigned to $p_i$. Thus the player will be able to pass downwards through the channel corresponding to $\ell_{km}$ iff $\ell_{km}$ evaluates to *True*. This example shows the $m$th channel for clause $c_k$, in the case where there are five variables involved in $\phi$:

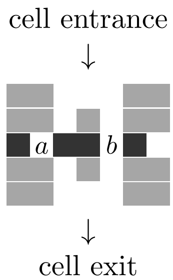



## Details of construction

Having described the scheme from the top down, we construct it from the bottom up.

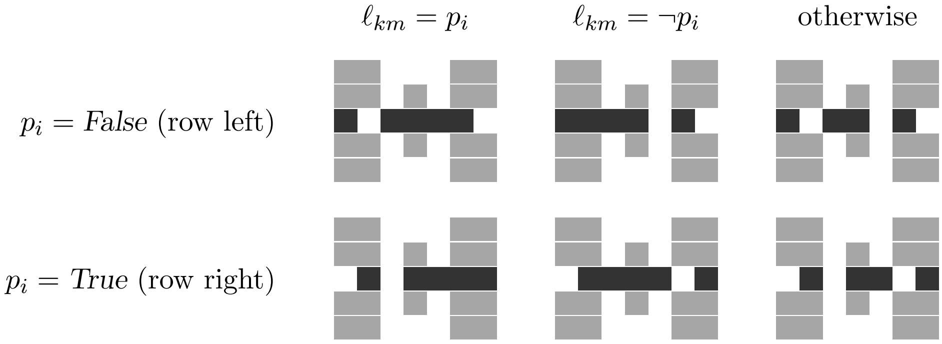
The only (relevant) movable rows in the level will be those representing the variables $p_i$. Each such row will have exactly two possible positions, 'left' and 'right'. If the row representing $p_i$ is in the left position, it will represent the assignment of *False* to $p_i$; if in the right position, *True*.

We first look at the construction of the cells $\chi_{kmi}$. It will have the following general shape, shown between sections of the two walls of its channel. The darker row of squares indicates the movable row corresponding to the variable $p_i$; it is shown in its 'left' position, and can also be moved one square to the right.



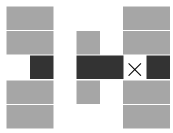

The square $a$ is occupied by a block iff $\ell_{km} = \neg p_i$; the square $b$ is occupied iff $\ell_{km} = p_i$. Therefore, if $\ell_{km}$ is neither of these (i.e., it involves $p_{i'}$ for some $i' \neq i$), squares $a$ and $b$ are both left unoccupied.

The effect of this on the player's ability to pass downwards through the cell is illustrated for the six possible cases:




Here it can be seen that the player can always pass through all cells $\chi_{kmi}$, except possibly the one cell for which $\ell_{km}$ involves variable $p_i$. For that cell, the player can pass through precisely when $\ell_{km}$ evaluates to *True*.

Note also that the player cannot pass through unless the dark row is left in the same position (left or right) as it was when the player entered the cell. Although the player can move the row while in the cell, for example by pushing the row rightwards when passing through the '$\ell_{km} = \neg p_i$, $p_i =$ *False*' case, this would result in a situation like the following, where the player's position is marked with '$\times$':

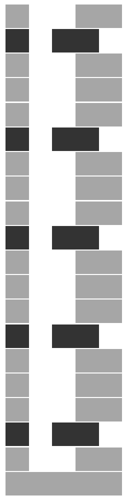


To proceed, the player has no choice but to push the row leftwards again, to where it was before she entered the cell. This argument applies to all cases.

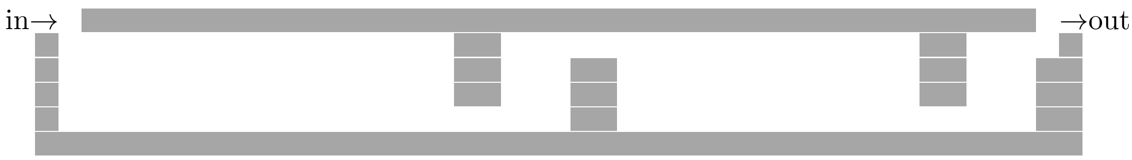
The upward 'corridors' are constructed along the same lines as the cells where the player can pass for either truth-value assignment. This is necessary so that we can apply the above argument to show that the movable rows are in the same position after the player has passed through the corridor as before she has passed through.

## Remaining details

The player must have some way of setting the positions of the movable rows corresponding to the variables. This is done by providing a section down the left-hand edge of the level where the player can move freely, setting each 'variable' row to its left or right position as desired:



Also, the player must be provided with means of access and egress from the level, in a way which does not compromise the construction. This is done by providing a strip across the top of the level:



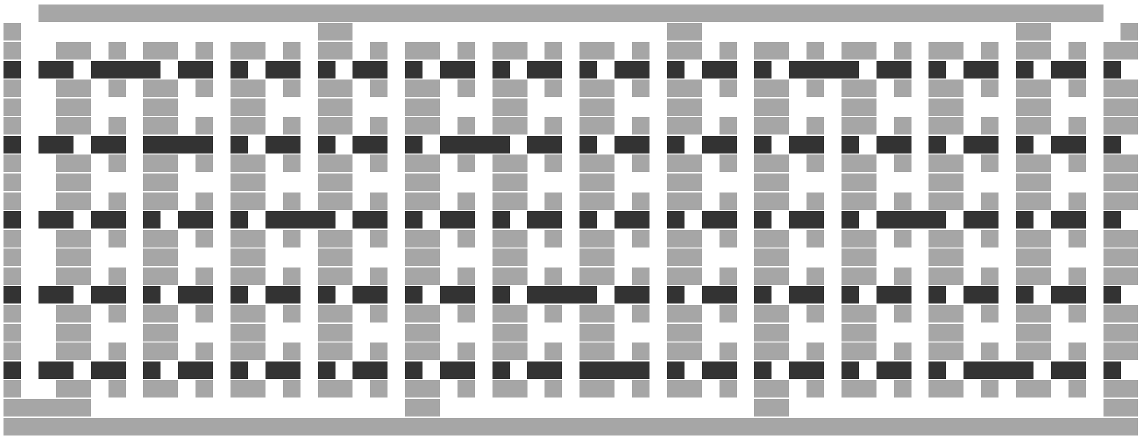

Note that although the very top row can be moved rightwards by the player, if she does so, it blocks the exit. We can therefore assume that in a successful solution to the level, the top row does not move.

## Final level example

Putting this all together, the following Mazezam level represents the question of whether there is an assignment of truth values to the variables $p_1, p_2, \ldots, p_5$ which causes the formula

$$\phi = (p_1 \vee \neg p_2 \vee p_3) \ \wedge \ (p_2 \vee p_4 \vee \neg p_5) \ \wedge \ (p_1 \vee p_3 \vee p_5)$$

to evaluate to *True*:



In this small example, we can see by inspection that setting

$$p_1 = \textit{True}; \quad p_2 = \textit{True}; \quad p_3 = \textit{True}; \quad p_4 = \textit{False}; \quad p_5 = \textit{False}$$

(among other assignments) will cause $\phi$ to evaluate to *True*. Indeed, moving the rows, by means of the left-hand section to represent this assignment, gives this configuration of the level, which can then be traversed by the player: