PSPACE-Completeness of Bloxorz and of Games with 2-Buttons

Tom C. van der Zanden[1(✉)] and Hans L. Bodlaender[1,2]

[1] Department of Computer Science, Utrecht University, Utrecht, The Netherlands
T.C.vanderZanden@students.uu.nl
[2] Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands
H.L.Bodlaender@uu.nl

Abstract. Bloxorz is an online puzzle game where players move a $1 \times 1 \times 2$ block by tilting it on a subset of the two dimensional grid, that also features switches that open and close trapdoors. The puzzle is to move the block from its initial position to an upright position on the goal square. We show that the problem of deciding whether a given Bloxorz level is solvable is PSPACE-complete and that this remains so even when all trapdoors are initially closed or all trapdoors are initially open. We also answer an open question of Viglietta [6], showing that 2-buttons are sufficient for PSPACE-hardness of general puzzle games. We also examine the hardness of some variants of Bloxorz, including variants where the block is a $1 \times 1 \times 1$ cube, and variants with single-use tiles.

1 Introduction

We study the computational complexity of the online puzzle game Bloxorz. We show that deciding whether it is possible to reach the goal square in a level of Bloxorz featuring switches and trapdoors is PSPACE-complete, which we will prove by reduction from Nondeterministic Constraint Logic. We first give a proof that requires us to be able to choose the initial state (open or closed) of each trapdoor. We will then also show that the problem remains PSPACE-complete even when the trapdoors are required to be initially all closed or all open.

Viglietta [6] studied the complexity of general computer games featuring certain common elements such as "destroyable paths, collectible items, doors opened by keys or activated by buttons or pressure plates, etc.". Viglietta established that a game featuring 2-buttons and doors is NP-hard and that a game featuring 3-buttons is PSPACE-hard. Viglietta asked whether this result could be improved to show that a game with 2-buttons is PSPACE-hard. We settle this question, showing that 2-buttons are indeed sufficient for PSPACE-hardness.

We also examine several variants of Bloxorz, and compare the hardness of decision and optimization versions of these. We show that Bloxorz with single-use tiles but without switches or trapdoors is NP-complete. Bloxorz with a $1 \times 1 \times 1$ cube instead of the $1 \times 1 \times 2$ block is NP-complete when both single-use tiles

V.Th. Paschos and P. Widmayer (Eds.): CIAC 2015, LNCS 9079, pp. 403–415, 2015.
DOI: 10.1007/978-3-319-18173-8_30

and switches and trapdoors are included, but becomes polynomially solvable if either is not present in the level.

1.1 Bloxorz

Bloxorz [1] is a "brain twisting puzzle game of rolling blocks and switching bridges" by Damien Clarke. A $1 \times 1 \times 2$ block is tilted around a subset of the two dimensional grid. Games using similar mechanics to Bloxorz are available on multiple platforms, including Android [2] and Apple iOS [3].

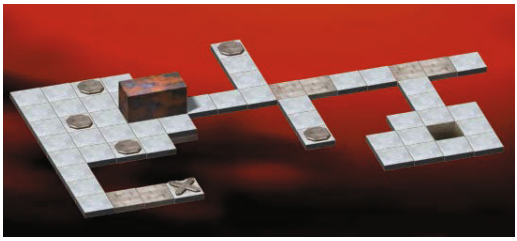
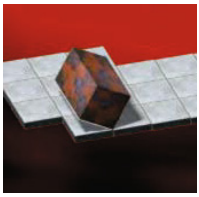


Fig. 1. An example Bloxorz level

The block can be in two states: lying down on a rectangular side, or standing up on a square face. From a standing position, the player may make a tilting move (Figure 2a) to place the block on its side. From a lying position, the player can either make a tilting move (Figure 2a) to stand the block up, or a rolling move (Figure 2b) after which the block will still be in a lying state.

(a) Tilting move



(b) Rolling move

Fig. 2. The two types of move available: (a) tilting and (b) rolling

Not all moves are possible: the player may only make a move if after that move, the block is still fully supported by the game level (the online game visualizes this by the block toppling off the stage if an illegal move is attempted).

The goal of the puzzle is to reach some specified goal square (Figure 3b) and get the block to fall through it. This implies the block must be in a standing orientation upon reaching the goal square but this constraint does not contribute to the hardness.

The online version of Bloxorz features various gadgets that make the game more interesting: switches that open and close trapdoors, switches that can only

(a) Trapdoor and switch

(b) Goal square

(c) Reaching the goal square

Fig. 3. More game elements: (a) switches and bridges, (b),(c) goal square

be triggered with the block in a specific orientation, weak tiles that will not support the block standing up, block-splitting portals and more. We will consider the version of Bloxorz with only switches, trapdoors and of course, regular squares. An example of a trapdoor and switch is shown in Figure 3a. Note that the full version of Bloxorz features pressure plates and thus is easily PSPACE-complete (Metatheorem 4c, [6]), we consider a more limited version with switches rather than pressure plates.

A *trapdoor* is a special game square that can be either open or closed. If the trapdoor is open, it is not allowed for the block to be on top of it. Each trapdoor is controlled by exactly one switch and each switch only controls one trapdoor.

A *switch* is a special game square that is associated with exactly one trapdoor. If a move places the block over a switch, the state of the associated trapdoor is toggled (from open to closed and vice-versa). A switch may be triggered more than once. If a move places a block over two switches then they will both be triggered.

If a move places a block over a switch and its associated trapdoor simultaneously, the effect of pressing the switch will be considered before checking the legality of the move. Our results remain valid if we check the legality of the move before activating the switch.

Note that in the original version of the game trapdoors are instead called "bridges" and are of size 1×2 but our constructions will use 1×1 trapdoors. However, our proofs can easily be adapted to work with 1×2 trapdoors instead.

We now formally define Bloxorz as a decision problem:

Bloxorz

Instance: A level of Bloxorz, given as a list of game tiles (normal tiles, trapdoors and switches) with their positions, a one-to-one mapping between trapdoors and switches, for each trapdoor an initial state (open/closed) and a start/goal square.

Question: Is there a sequence of legal moves from the start to the goal square?

The precise encoding used for problem instances is not important. We consider the size of a level to be the number of tiles in it.

1.2 Previous Work

Buchin [9] showed that solving rolling block mazes is PSPACE-complete. The block in Bloxorz moves in exactly the same way as a block in a rolling block maze, but a rolling block maze may feature multiple blocks (which may interfere with each other's movement) while Bloxorz only features a single block. Bloxorz also differs from rolling block mazes in that it features switches and trapdoors. These are required for the hardness since a rolling block maze with only a single block is solvable in logarithmic space [12], since it is a connectivity problem.

Viglietta [6] established several metatheorems regarding the complexity of games with buttons[1], doors and pressure plates.

A *door* is a special game square that may be either in a closed or in an open state, and the avatar may pass through the door if and only if it is open. Bloxorz features trapdoors which are similar to doors except that the notions of open and closed are reversed (i.e., the avatar may pass over the trapdoor if and only if it is closed).

The state of a door may be altered by pressure plates and buttons. Viglietta defines a *pressure plate* as a special game square "that is operated whenever the avatar steps on it, and its effect may be either the opening or the closure of a specific door".

In addition, Viglietta defines *buttons* which function similar to pressure plates except that the player has a choice whether to trigger the button or not, while a pressure plate is activated immediately when the player steps on it.

In Bloxorz, a switch instead toggles the state of a trapdoor between open and closed, while a pressure plate or button may either open or close a door, but may not toggle between the two states. Like a pressure plate, a switch is toggled when the block moves over it but due to the $1 \times 1 \times 2$ shape of the block two switches may be triggered simultaneously.

A single door may be controlled by up to two pressure plates: one that opens it, and one that closes it. In Bloxorz, one trapdoor is controlled by exactly one switch.

A k-button may control up to k doors simultaneously, possibly opening some while closing others. Viglietta established that a game featuring doors and k-buttons with $k \geq 2$ is NP-hard, while a game with $k \geq 3$ is PSPACE-hard.

Viglietta poses the question whether his result for $k = 2$ can be improved upon, that is, if a game with 2-buttons is PSPACE-hard. We show that this is the case.

Our reduction is based on the work by Hearn and Demaine [8] on Nondeterministic Constraint Logic.

[1] In the conference version of his paper [5], Viglietta uses the term "switch" instead of "button". In the final version of the paper [6], he uses "button" to describe the same object. We use this newer terminology since it better describes the functionality of the object.

2 Nondeterministic Constraint Logic

First, we briefly introduce (restricted) Nondeterministic Constraint Logic (NCL).

A *constraint graph* is an undirected graph $G = (V, E)$ with for each vertex and edge a weight that is a positive integer. We only need to consider weights of 1 and 2. The weight of a vertex is called its *minimum inflow*.

A *configuration* is an assignment of an orientation to each edge, which is *legal* if and only if, for each vertex v, the sum of the weights of the edges pointing into v is at least the minimum inflow of v.

An *edge reversal* is the operation that reverses the orientation of an edge. Thus, its weight starts counting towards the other vertex. A reversal of (v, w) is *legal* if the minimum inflow constraints for v and w remain satisfied.

We consider only constraint graphs featuring two special types of vertices:

- An *AND vertex* is a vertex of degree 3, with minimum inflow 2 and with incident edge weights of 1, 1 and 2. It works like a logical AND gate because either the weight 2 edge must be directed inwards or the two weight 1 edges must be directed inwards.
- An *OR vertex* is a vertex of degree 3, with minimum inflow 2 and with incident edge weights all equal to 2. It works like a logical OR since to satisfy its minimum inflow requirement, at least one of its edges must be directed inwards.

Restricted NCL

Instance: A constraint graph G built exclusively from AND and OR vertices, a legal initial configuration for G and a given target edge e from G.

Question: Is there a sequence of legal edge reversals that, starting from the specified initial configuration, reverses e?

Theorem 1 (Hearn and Demaine [8]). Restricted NCL *is* PSPACE*-complete.*

Note that Restricted NCL remains PSPACE-complete even when restricted to planar graphs, but we do not make use of this property in our proof. This is because the structure of the NCL graph is encoded in the switch-trapdoor correspondence of the Bloxorz level, rather than in its physical structure.

3 PSPACE-Completeness of Standard Bloxorz

Theorem 2. Bloxorz *is* PSPACE*-complete.*

Proof. By reduction from Restricted NCL. We first establish PSPACE-hardness. We first describe the various constructions used to represent elements from a constraint graph, then show how to put them together into a Bloxorz level. Finally, we show that Bloxorz is in PSPACE.

3.1 Overview

Figure 4 shows a simplified overview of our construction. The white area represents a section of the game level where the block can move freely. The grey boxes represent copies of the vertex and edge gadgets, which will be described in detail later. The start and goal squares are marked with S and G respectively.

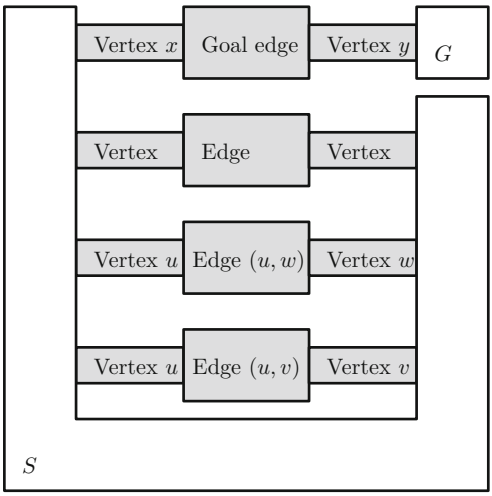


Fig. 4. Overview of the construction

A solution to the Bloxorz level will correspond to a solution to the NCL problem as follows: the player will cross the gadgets from left to right and from right to left a number of times to solve the level. Crossing the edge gadget (u, v) from left to right (resp. from right to left) corresponds to changing the orientation of the edge (u, v) to be towards u (v). To cross the edge gadget, the player will need to activate a number of switches in the edge gadget which also changes the state of the trapdoors in the vertex gadgets. Since reaching the goal square requires crossing the goal edge gadget, the level can thus be solved if and only if the goal edge can be reversed.

The vertex gadgets are constructed so that it is possible to leave the edge via them if and only if their inflow constraint is satisfied. So, if the player crosses the edge (u, v) from left to right, orienting the edge towards u, the vertex v loses inflow while u gains inflow. Since gaining inflow can not cause a constraint to be violated, we only need to check the constraint at vertex v.

Note that the input graph only has degree 3 vertices. For every vertex in the input graph, 3 copies of the vertex gadget are created - one for each of the incident edges. These are related: in our example, crossing the edge (u, v) would also affect the state of the trapdoors in the gadget for vertex u attached to edge (u, w). Also note that the structure of the NCL graph is encoded by the switch-trapdoor correspondence and not in the physical structure of the level.

3.2 Edge Gadget

Figure 5 shows the construction used to represent an edge from an NCL graph. The gray squares represent normal tiles, the squares with fat borders represent trapdoors. Circles represent switches, and the lettering shows the switch-trapdoor correspondence. Trapdoors and switches with white infill correspond to open trapdoors and conversely, grey switches and trapdoors correspond to closed trapdoors. The switches labelled with u, v correspond to trapdoors that are inside vertex gadgets, which will be introduced later.

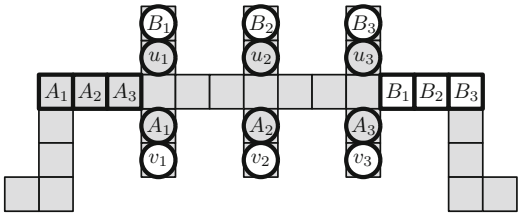


Fig. 5. The edge gadget. The vertex gadget corresponding to vertex v will be attached to the left side of this gadget, and a vertex gadget corresponding to vertex u to the right side.

On either end, the edge gadget will be attached to a corner gadget. The edge gadget is shown pointing to the right vertex, i.e., its weight is counting towards the right vertex.

To reorient the edge gadget, the player would enter it from the left and travel over the trapdoors labelled A. They would then trigger all the switches exactly once, which leaves the trapdoors labelled A open while closing the trapdoors labelled B. The block can then leave the edge via the right side, travelling over the trapdoors labelled B. Note that due to the $1 \times 1 \times 2$ shape of the block it is only possible for the switches to be triggered in pairs (e.g., B_1 is always triggered at the same time as u_1) so this leaves the trapdoors labelled v closed and the ones labelled u open. The edge gadget is now in the opposite orientation (left) and to reverse it again, one would make the same moves but starting from the right.

Note that when we reorient the edge from right to left, the right vertex loses weight while the left vertex gains weight. This means that the only place a constraint could possibly have been violated is the right vertex. Since the trapdoors labelled A are now open, we have to leave via the right side by travelling over the trapdoors labelled B. The vertex gadget will be constructed in such a way that it enforces the constraint: the vertex gadget will prevent us from leaving the edge gadget if its constraint is not satisfied.

Above, we described a "canonical" movement of the block over the edge. Clearly, the player can decide to move the block over the edge gadget in many different ways. For example, they could enter it from the left, activate all the

switches on the top side, closing the trapdoors labelled B and opening those labelled u. They could then leave the edge again from the left side, and the edge's weight is essentially removed from the graph: it no longer gives its weight to either vertex. And even more cause for concern: we did this without checking that the constraint of the right vertex was not violated!

However it is never beneficial to only complete a reorientation partially. If at any point we want the weight of the edge to start counting towards the left vertex we have to open (at least one of) the trapdoors labelled A which forces us to exit via the right side and check the constraint for the right vertex. Also, note that the edge will never give its weight to more than one vertex since that would mean both a trapdoor labelled A is open and one labelled B is open which makes it impossible to leave the edge.

Claim. Upon leaving the edge gadget, it is not possible for a trapdoor labelled u to be closed while a trapdoor labelled v is also closed.

Proof. Leaving the edge gadget requires all the trapdoors A to be closed or all the trapdoors labelled B to be closed, which in turn enforces the condition of the lemma. □

This enables us to define the orientation of the edge gadget: the edge gadget is oriented away from the vertex u if the trapdoors labelled u are open and oriented away from v if all the trapdoors labelled v are open. If both trapdoors u and v are open then we may consider the edge gadget oriented either way.

3.3 Vertices

A vertex is represented by three separate, identical gadgets, that attach to the edge gadgets corresponding to its incident edges. The edge has three switches corresponding to trapdoors in each of the three copies of the vertex gadget. Every vertex has three trapdoors, and every trapdoor is associated with a distinct edge connecting to the vertex.

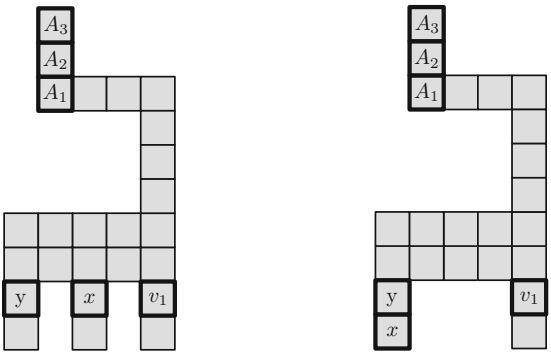
OR Vertex Gadget. Figure 6a shows one of the three identical components that together form one OR vertex gadget. It is drawn with part of an edge gadget attached. The trapdoors labelled x, y and v_1 each correspond to one of the three edges incident to the vertex. The following claim is self-evident. Note that it enforces the minimum inflow constraint: if one of the trapdoors is closed then the edge associated with that trapdoor must be pointing towards this vertex.

Claim. It is possible for the block to leave the edge via the OR vertex gadget if and only if at least one of the trapdoors is closed.

AND Vertex Gadget. Figure 6b shows one of the three identical components that together form one AND vertex. The following claim regarding the AND vertex gadget is self-evident:

Claim. It is possible for the block to leave the edge via the AND vertex gadget if and only if the trapdoor labelled v_1 is closed or both the trapdoors labelled x and y are closed.

Thus, the edge associated with trapdoor v_1 functions as the weight-2 edge and the edges associated with trapdoors x and y function as weight-1 edges. The vertex gadget enforces that the minimum inflow constraint is satisfied.



(a) OR vertex (b) AND vertex

Fig. 6. Vertex gadgets, showing the three identical components that make up: (a) an OR vertex, (b) an AND vertex. The vertex gadgets attach to edge gadgets at the top and to the rest of the game level at the bottom. Note that the trapdoors labelled with A are part of the edge gadget.

3.4 Final Details

We now show how to assemble the component gadgets, forming a Bloxorz level.

For every edge in the constraint graph, one copy of the edge gadget is created. For every vertex, three copies of the appropriate vertex gadget are created and attached to their respective edges. Then, we add some additional blocks so that every vertex can be reached from every other vertex, with the sole exception of the vertex to which the target edge is pointing. It is not attached to any other vertex, instead, it is attached to the goal square. We can place the start square anywhere on the level, except on those squares from which the goal square can be reached directly.

Claim. An instance of Bloxorz constructed in this way from a Restricted NCL instance has a solution if and only if there exists a sequence of edge reversals that reverses the target edge in the NCL instance.

Proof. If we have a solution to the NCL instance then the sequence of edge reversals translates directly to a solution for the Bloxorz level by traversing the edge gadgets in the same order.

A solution that solves the Bloxorz level can be translated in to a solution for the NCL instance as follows: every time the player traverses an edge gadget associated with edge (u, v) and leaves via the vertex gadget u, this corresponds to reorienting edge (u, v) away from u. Since such a reversal only increases the weight of v, its minimum inflow must remain satisfied. The inflow constraint for u must also remain satisfied since this is enforced by the vertex gadget. □

Claim. Bloxorz is in PSPACE.

Proof. Since each state can be represented with polynomial space and we can generate the successor states efficiently, a nondeterministic search of the search space shows that Bloxorz is in NPSPACE. By Savitch's theorem [10], NPSPACE = PSPACE, so Bloxorz is in PSPACE. □

Since we have shown that Bloxorz is both contained in PSPACE and PSPACE-hard, Bloxorz is PSPACE-complete, and we have thus shown Theorem 2. □

Our proof requires that we are able to choose the initial state of each trapdoor (open or closed) when specifying the problem instance. The following theorems show this is not essential to the hardness:

Theorem 3 ([4]). Bloxorz *remains* PSPACE*-complete even when limited to the instances where all trapdoors are initially open.*

Theorem 4 ([4]). Bloxorz *remains* PSPACE*-complete even when limited to the instances where all trapdoors are initially closed.*

4 Hardness of General Puzzles with Buttons

Viglietta [6] established that a game featuring 2-buttons and doors is NP-hard and asked whether this could be improved to show such games are PSPACE-hard. We show that such games are indeed PSPACE-hard.

Note that for a door, the notions of open and closed are inverted from that of a trapdoor.

Theorem 5. *A game where the avatar has to reach an exit location to win that features 2-buttons and doors is* PSPACE*-hard.*

Proof. Our construction to show PSPACE-hardness of Bloxorz can be adapted to show PSPACE-hardness of a game featuring 2-buttons and doors. Replace every trapdoor by a door. In the edge gadget of Figure 5 replace every pair of switches B_i and u_i by a pair of 2-buttons: one 2-button that opens B_i and closes u_i and another that closes B_i and opens u_i. Do the same for switches labelled A and v. This new construction functions in the exact same way as the edge gadget for Bloxorz. □

Figure 7 shows an example of this construction. The small white rectangles depict buttons, and are labelled to show which doors they open (marked with a plus sign) and which doors they close (marked with a minus sign).

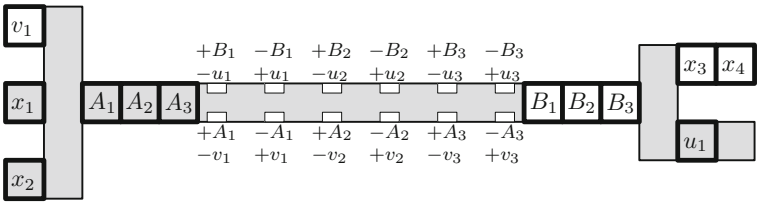


Fig. 7. Edge gadget for 2-button games, depicted with an OR vertex gadget on the left and an AND vertex gadget on the right.

Theorem 6 ([4]). *A game that matches the conditions of Theorem 5 is* PSPACE*-hard, even when all doors are initially closed. When all doors are initially open the problem can be decided in logarithmic space [12].*

5 Variants of Bloxorz

In order to more closely identify what features make Bloxorz hard, we now look at various simplified variants and determine their complexity. Table 1 shows the complexity for various variants: we list the complexity of the corresponding decision problem (determining whether a solution exists) and optimization problem (determining whether a solution with at most k moves exists) and also the number of moves required in the worst case.

Some of the results in this table are trivial so we will not discuss them.

Table 1. Complexity of various Bloxorz variants, listing the complexity of the decision variant, optimization variant (deciding whether a solution of certain maximum length exists) and the solution length in the worst case.

	$\mathbf{1 \times 1 \times 2}$			$\mathbf{1 \times 1 \times 1}$		
	Dec.	Opt.	Moves	Dec.	Opt.	Moves
None	P	P	$\Theta(n)$	P	P	$\Theta(n)$
Switches+Trapdoors	PSPACE-C	PSPACE-C	$2^{\Theta(n)}$[2]	P	NP-C[3]	$\Theta(n^2)$
Single-use tiles	NP-C	NP-C	$\Theta(n)$	P	P	$\Theta(n)$
S+T+Single-use	PSPACE-C	PSPACE-C	$2^{\Theta(n)}$[4]	NP-C[5]	NP-C	$\Theta(n^2)$

[2] This follows from our PSPACE-completeness proof. Solving a configuration created in this way from an NCL instance can take exponentially many moves.

[3] Metatheorem 2 of [7]. Collecting items is implemented by forcing a number of trapdoors to be closed before the exit square can be reached.

[4] See footnote 2.

[5] Metatheorem 1 of [6]. Location traversal can be implemented by forcing the player to close a number of trapdoors.

The heading of table 1 shows what size of block and which type of problem is considered, while the leftmost column shows which (if any) special features can be present in the game level.

We consider the regular $1 \times 1 \times 2$ variant of Bloxorz but also consider what happens when we use a $1 \times 1 \times 1$ cube instead. We consider various game elements: trapdoors and switches as previously examined but also single-use tiles. Single-use tiles are game squares that the block can only pass over once before they break and are removed from the game board. Single-use tiles are actually not featured in the original Bloxorz game. We discuss them here since they are an example of a game feature that, when combined with switches and trapdoors will make the decision for the $1 \times 1 \times 1$ case NP-complete.

An interesting observation is that $1 \times 1 \times 2$ Bloxorz with single-use tiles is NP-complete while this is not the case for $1 \times 1 \times 1$ Bloxorz:

Theorem 7 ([4]). *$1 \times 1 \times 2$ Bloxorz with single-use tiles is NP-complete, while $1 \times 1 \times 1$ Bloxorz with single-use tiles can be solved in polynomial time.*

Furthermore, we can decide Bloxorz with trapdoors and switches in polynomial time when we do not use a rectangular block but a $1 \times 1 \times 1$ cube:

Theorem 8 ([4]). *In the $1 \times 1 \times 1$ case, a Bloxorz instance with switches and trapdoors can be solved in polynomial time, taking $\Theta(n^2)$ moves in the worst case.*

However, when trapdoors, switches and single-use tiles are combined, $1 \times 1 \times 1$ Bloxorz becomes NP-complete again.

6 Conclusion

We have shown by a reduction from Restricted NCL that Bloxorz is PSPACE-complete, even when the trapdoors are initially fixed either open or closed. We also showed how our proof can be adapted to general games and showed that games with 2-buttons and doors are not only NP-hard, but also PSPACE-hard, answering an open question of Viglietta [6].

We then examined some other variants of Bloxorz, including variants where the $1 \times 1 \times 2$ block is replaced by a $1 \times 1 \times 1$ cube and variants where single-use tiles are included. We showed that Bloxorz with single-use tiles but without switches or trapdoors is NP-complete, and that most variants with a $1 \times 1 \times 1$ block can be polynomially solved. Only the $1 \times 1 \times 1$ variant featuring both single-use tiles and switches and trapdoors is NP-complete.

We think our result clearly illustrates the power of the NCL framework. It is not at all obvious that a universal quantifier may be constructed from the elements in Bloxorz; the work by Demaine and Hearn [8,11] is instrumental for establishing the complexity of this and other puzzle and reconfiguration problems.

Acknowledgement. We thank Damien Clarke for graciously allowing us to reproduce screenshots of Bloxorz and the referees for their useful comments.

References

1. Bloxorz, DX Interactive. http://dxinteractive.com/#!/bloxorz
2. Ice Cube Caveman, Google Play Store. https://play.google.com/store/apps/details?id=com.TwistedGames.Caveman
3. Ice Cube Caveman, iTunes App Store. https://itunes.apple.com/app/ice-cube-caveman/id456179222?mt=8
4. van der Zanden, T.C., Bodlaender, H.L.: PSPACE-Completeness of Bloxorz and of Games with 2-Buttons arXiv:1411.5951 (2014)
5. Viglietta, G.: Gaming is a hard job, but someone has to do it!. In: Kranakis, E., Krizanc, D., Luccio, F. (eds.) FUN 2012. LNCS, vol. 7288, pp. 357–367. Springer, Heidelberg (2012)
6. Viglietta, G.: Gaming Is a Hard Job, but Someone Has to Do It!. Theory Comput. Syst. **54**(4), 595–621 (2014)
7. Forišek, M.: Computational complexity of two-dimensional platform games. In: Boldi, P. (ed.) FUN 2010. LNCS, vol. 6099, pp. 214–227. Springer, Heidelberg (2010)
8. Hearn, R.A., Demaine, E.D.: PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. Theoretical Computer Science **343**, 72–96 (2005)
9. Buchin, K., Buchin, M.: Rolling block mazes are PSPACE-complete. Journal of Information Processing **20**(3), 719–722 (2012)
10. Savitch, W.J.: Relationships between nondeterministic and deterministic tape complexities. Journal of Computer and System Sciences **4**, 177–192 (1970)
11. Hearn, R.A., Demaine, E.D.: Games Puzzles, and Computation. A K Peters/CRC Press (2009)
12. Reingold, O.: Undirected connectivity in log-space. Journal of the ACM **55**(4), article 17 (2008)