TIC-TAC-TOE CAN, WITH HEXAGONS! HEXO IS PSPACE-HARD

NUMBERBASHER

Abstract. We prove, by reduction from Quantified 3SAT, that Connect$(n, 6, 2, 1)$ on a hexagonal grid is PSPACE-complete.

Contents

1. Introduction

To answer the question of *"can tic-tac-toe, with hexagons?"*, webgoatguy first created infinite hexagonal tic tac toe, or HeXO, in March of 2026. HeXO is essentially the game of Connect$(\infty, 6, 2, 1)$ on a hexagonal grid, where Connect(n, k, p, q) is a connect-k game played with two players, each of which makes p moves except for the first ply in which the player only makes q moves, on a board with size n. By convention, we will assume for the rest of this paper that Connect is played on a hexagonal grid.

By providing a polynomial-size, polynomial-size reduction from Quantified 3SAT, to Connect$(n, 6, 2, 1)$, we show that Connect$(n, 6, 2, 1)$ is PSPACE-complete, and hence, by reduction from Connect$(n, 6, 2, 1)$ to HeXO, that HeXO is PSPACE-hard.

2. Membership

Let the problem of Connect by defined as deciding whether player 1 (portrayed as yellow) is winning. It is easy to see that, by performing a brute force algorithm that searches through the entirety of the game tree, since the game tree has polynomial depth (specifically, $O(n^2)$ squares, so $O(n^2)$ plys), the call stack requires polynomial space, so we can solve the problem with space proportional to some polynomial in terms of the size of the problem.

Therefore, Connect is in PSPACE. Notably, HeXO is played on an infinite grid, so similar notions do not apply to HeXO.

3. Introduction to HeXO Strategies

Moves which threaten a win next ply are called *threats*.

Winning in HeXO largely revolves around creating *double threats*, sometimes shortened to *doubles*, which are threats which require two moves to respond to. The most common *double* is the *quad*:

(As a convention, we will have yellow be the attacking player, and hence most threats will be made by yellow.) Notice how blue now has to block on both of the highlighted squares; otherwise, yellow wins to the left or to the right:

Blue actually has the option of *gapping* at either side, which is to place the piece one further along the line:

For the most part, whether blue gaps is irrelevant. However, note that blue cannot gap on both sides, since yellow would then be able to win:

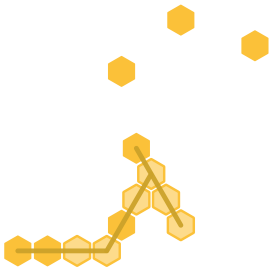
Another double is the *one-three-one*, specifically the dashed variant, which we will be using extensively:

Blue has to block on both sides of the three connecting pieces, so all of these are good blocks:

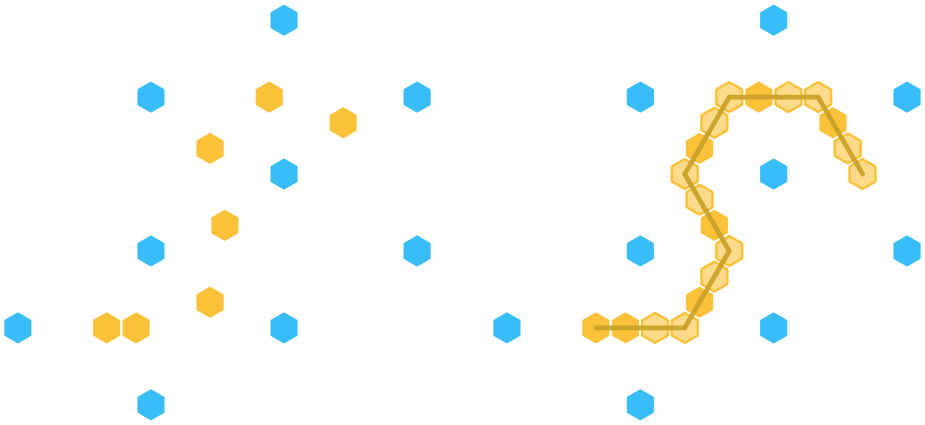
Again, which choice blue uses to block will largely be irrelevant.

These are the only doubles that we will see in our construction. Doubles are extremely useful, since they essentially act as wires or transmitters: yellow can spend two moves to construct a double, blue can then spend two moves to resolve a double, and then this process can repeat. For instance, in the following scenario, yellow can start from the bottom, repeatedly extending quads.

There are two things worthy of noting. First, yellow has multiple ways of attack. This allows for a lot of *cheese*, which is to say, unintended ways for yellow to progress:

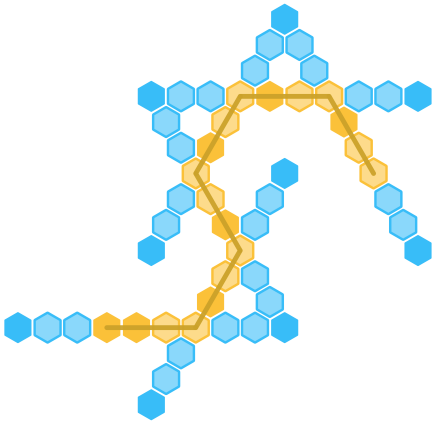


In order to stop this, we can wrap everything around a grid of blue cells:



If yellow does not extend exactly in the way as portrayed, then the grid made up by the blue pieces will block one side of the yellow threat, and hence yellow will not be making doubles.

Another thing worth noting is the possibility for blue to make *defensive threats*, or sometimes called *defths*, which is when, while blocking off doubles from yellow, blue creates a threat. Since blue usually has a lot of choices when blocking quads, the easiest way to scan for defths is to simply highlight all possible cells blue can occupy:



It is easy to see that no wire configuration can ever produce defths.

4. Introduction to Quantified 3SAT

SAT is the *boolean satisfiability problem*. The decision problem is whether there exists values for boolean variables $x_1, x_2, x_3, \cdots, x_n$, where n is the size of the problem, such that some clause, expressed in terms of x_i and the logic symbols $\neg$ (NOT), $\wedge$ (AND), and $\vee$ (OR). That is, for some proposition Prop, SAT evaluates $\exists x_1\ \exists x_2\ \exists x_3\ \cdots\ \exists x_n\ \text{Prop}(x_1, x_2, x_3, \cdots, x_n)$.

Quantified SAT introduces quantifiers to the SAT problem, namely $\exists$ (EXISTS) and $\forall$ (FORALL). Notably, the problem of quantified SAT is PSPACE-hard, which is to say, at least as hard as any problem solvable in space proportional to some polynomial in terms of the size of the problem. Hence, if we can find a polynomial-space, polynomial-size reduction from HeXO to quantified SAT, i.e. encode quantified SAT in HeXO, then we can show that HeXO is PSPACE-hard.

However, quantified SAT is too versatile to encode. Thankfully, we can use *quantified 3SAT* instead, where 3SAT is the satisfiability problem encoded as the conjunction (AND) of several *clauses*, each of which is the disjunction (OR) of exactly three *literals*, each of which is either x_i or $\neg x_i$. . Additionally, we will show a reduction from Connect$(n, 6, 2, 1)$ to HeXO. Therefore, what remains to be shown is a reduction from quantified 3SAT to Connect$(n, 6, 2, 1)$.

Recall that Connect$(n, 6, 2, 1)$ is the problem of deciding whether yellow can win. Hence, there is a very natural correspondence between a quantified SAT problem and a Connect problem: each boolean variable can be thought of as a choice between two possible alternatives, either x_i or $\neg x_i$. Each $\exists$ quantifier is a potential choice by yellow, the attacker, and each $\forall$ quantifier is a potential choice by blue, the defender. Then, the question amounts to if there exists a strategy for yellow to make the correct choices, for any set of choices by blue.

5. Proof Sketch

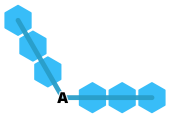
First, we will make sufficiently winning structures for blue such that if yellow at any point stops making doubles, blue can take the extra piece to win. Hence, we force yellow to constantly create doubles.

Yellow will have a starting point at which to start a sequence of doubles, and at each point there will be limited options for the next doubles to allow the wire to transmit. We will then create two gadgets, one for variable selection by yellow, i.e. where yellow can select where the wire goes out of two options, and one for variable selection by blue, i.e. where blue can select where the wires goes out of two options. We can make these selections in order by duplicating a copy of each option (duplicating comes automatically since our wires can just split off), joining the two options with OR, and have that be the wire going into the next selection.

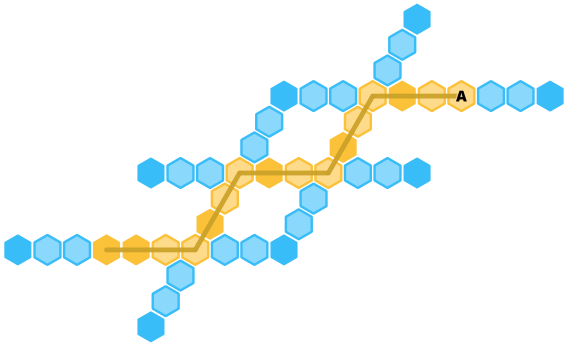
We also create the logical OR and AND gadgets, allowing us to construct the proposition in the form of a wire from the literals just selected. Finally, we lead this wire into a winning configuration for yellow if the wire has been completed.

6. Gadgets

6.1. **Ultimatum.** The ultimatum by blue forces yellow to always create doubles, since if blue can ever place a hex freely, they can create two quads, or a winning quadruple threat, by placing a hex at A.

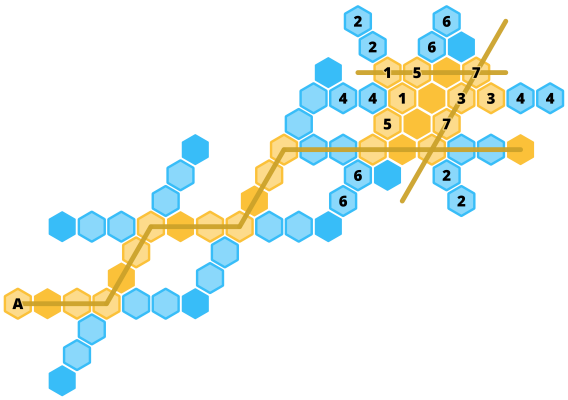


6.2. **Start.** The start gadget simply allows yellow to start making doubles.



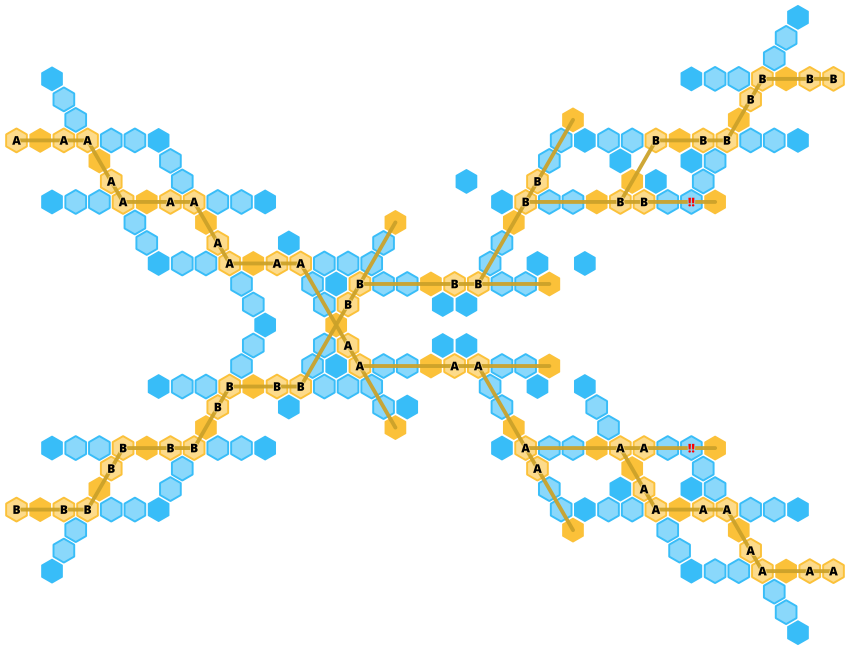
The wire extends outwards from A.

6.3. **End.** The end gadget allows yellow to win if the wire is activated.



A winning sequence for yellow, when A is activated, is shown.

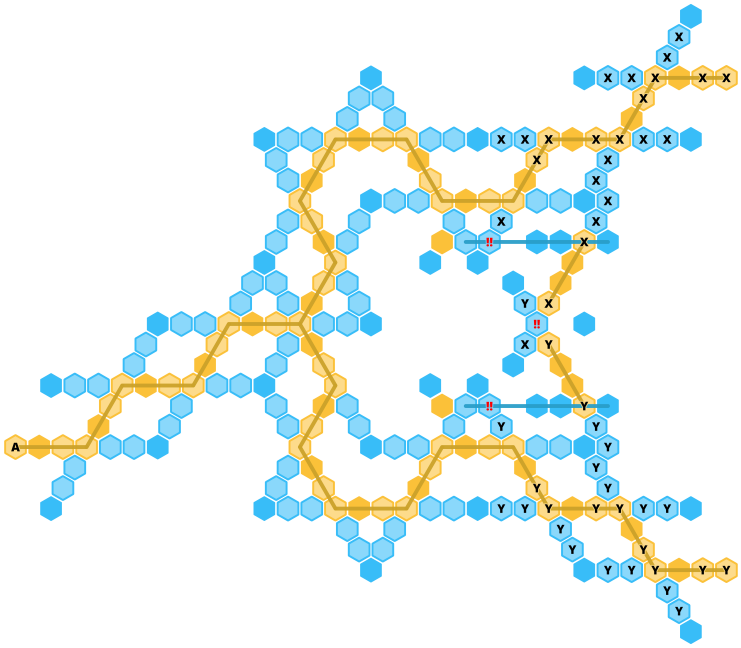
6.4. **Crossing.** It is critical to have a crossing gadget:



We simply have the two wires labelled with A and B travel to the right; the important portion is the crossing, and then we re-align the signals to the hexagonal grid to address any potential parity issues otherwise.

The two specific blocks by blue marked are required; blue has no reason to deviate. If another block is taken, then yellow can cheese the wires.

6.5. **Exists.** Now, let us make the exists gadget. That is, when we attempt to transmit from A, then we cannot reach both X and Y, but yellow decides which we can reach.



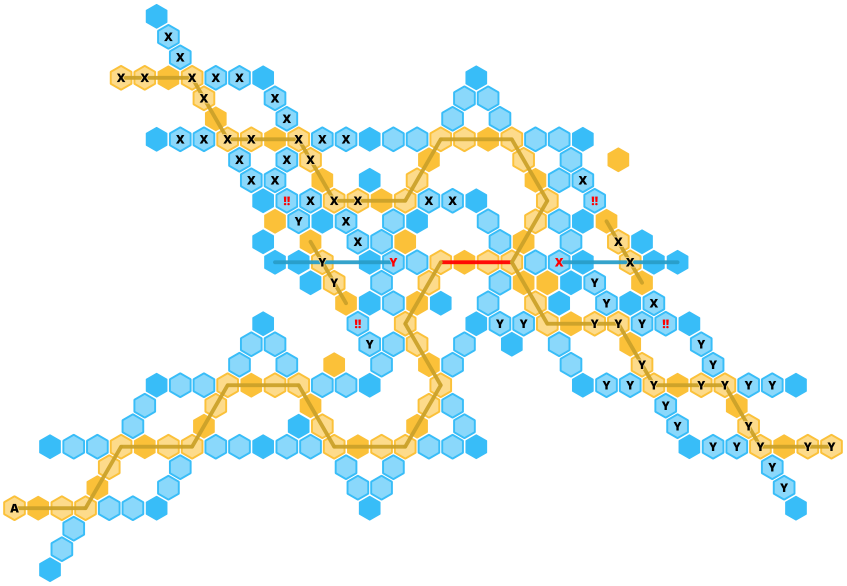
It is clear that the wire can transmit from A to either X or Y. What remains to be shown is that it cannot be transmitted to both. When A is transmitted to X, blue has the capacity to make a defth. Since yellow has to continue making doubles, yellow has to address the defth with a double. The situation is symmetric in the case of Y instead of X. However, the lines at which the doubles addressing the defths are made intersect at a common point, and hence once one of the paths are chosen, the other can no longer be chosen as the double is reduced to a single threat since one side is already blocked.

Some specific blocks from blue are required, but blue has no reason to deviate.

6.6. **OR.** Logical OR is exactly the above construction, with X and Y as inputs and A as the output! It is clear that either X or Y can transmit to A, so what remains to be shown is that X cannot transmit to Y (and vice versa), or otherwise we will have mingling between inputs in our circuitry.

But X cannot transmit to Y for the same reason that A cannot transmit to both X and Y, and hence we are done.

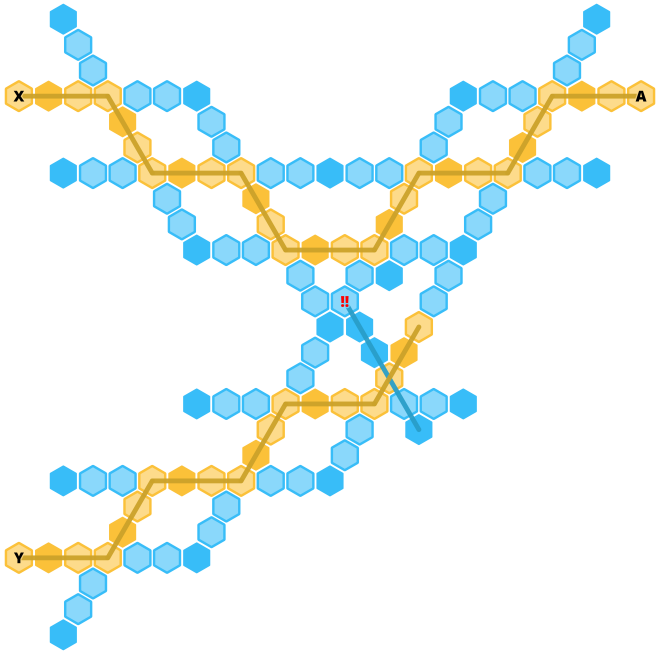
6.7. **Forall.** Now, let us make the forall gadget. That is, when we attempt to transmit from A, then we cannot reach both X and Y, but blue decides which we can reach.



During transmission, blue can decide to either block the highlighted yellow double at the highlighted X or at the highlighted Y, but critically not at both. Blue has no reason to block at neither X nor Y.

If blue blocks at the highlighted X, it creates a defth, forcing yellow to address the defth with a double. Then, after blue blocks the double, the transmission path to Y will be blocked, and hence yellow is forced to not select Y, i.e. select X. Similarly, if blue blocks at the highlighted Y, the same sequence occurs, and yellow is forced to select Y.

6.8. **AND.** Finally, we create the logical AND, which is simple.



Here, yellow must cut off the potential defth by blue with the Y wire, and then the X wire is allowed to pass through to transmit to A. If the Y wire is not activated, then blue will have a defth which cannot be addressed when X is transmitted to A.

6.9. **Parity.** As necessary, create islands of yellow or blue pieces in order to ensure that the current game state is valid.

Hence, this concludes our gadgets, and our proof.

7. Reduction to HeXO

We have proven that Connect$(n, 6, 2, 1)$ is PSPACE-complete. Define HeXO as the game of Connect$(\infty, 6, 2, 1)$, where the size of a problem is the number of existing pieces.

It is clear that the trivial reduction from Connect$(n, 6, 2, 1)$ to HeXO is of polynomial size, and hence HeXO is PSPACE-hard.

8. Generalizations

Connect$(n, k, 2, 1)$, for $k > 6$, is left as an exercise to the reader.

It is also easy to generalize this to 8-connected square grids.

9. Future Work

Lingering questions still remain: what is the hardness of Connect$(n, k, 2, 1)$ for $k = 5$? What about the game of Connect, played on a 4-connected square grid?

Perhaps the biggest open problem is, is HeXO PSPACE-complete, that is, is HeXO solvable in polynomial space? The author, personally, is uncertain. It yet seems possible to create a structure capable of replicating itself, and then have the game end in a controlled threat war, hence proving the game undecidable. Alternatively, it may be possible to rigorously assert the notion that pieces which are far enough away may never contribute the game, and hence the game is PSPACE-complete.

10. Acknowledgements

I would like to thank, first and foremost, my parents, fellow classmates, and teachers, for making this possible.

Without YouTube content creator `webgoatguy`, there would neither be this game, nor be a community even comparable to the one right now. We all owe great debt to webgoatguy for the development of this game. To the community I also owe great debt to those such as `Nikki`, `cynifer`, `Chiser`, `alfaozz`, and `ThatsCrispy`, etc., as well as the moderator team (except me) who kept the community together.

On Discord, `Val0`'s presentation on NP-hardness of HeXO inspired this result, and `Zye` and `2swap` offered valuable pointers as to where to start. Without the valuable site built and maintained by `WolverinDEV`, this project would be much harder.

Credit is due to `MineKing` for developing HeXO-Renderer, for making it open-source and contributable, and for offering me constant help and adding features for me throughout. All visuals were generated with an external tool developed for personal use which links to HeXO-Renderer.

I would also like to thank Professor Erik Demaine and his public resources as well as research work regarding hardness reductions. Perhaps there exists a simpler, or otherwise more generalizable, proof using Quantified Sided Var-Linked Rectilinear Planar 3SAT.