Extra Notes: On the Computational Hardness of Go

We've got several examples now of $\mathcal{NP}$-complete problems, but not too many examples of problems that are $\mathcal{PS}$-hard. The increased difficulty of these problems makes them somewhat less accessible, but it's worth seeing a few of them anyway. With that in mind, let's hve a look at the complexity of the game of Go.

Contents

1 A Very Brief Introduction to Go **1**
1.1 The Game of Go . . . 2

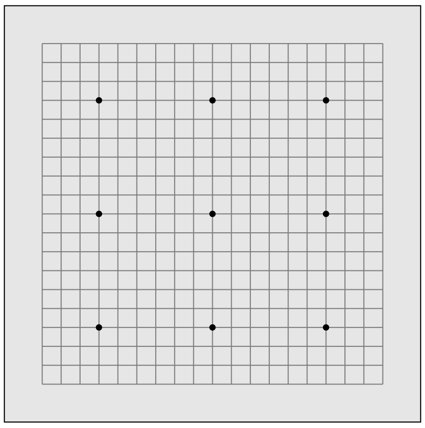
2 Go and Computational Complexity **6**
2.1 The Reduction . . . 9

1 A Very Brief Introduction to Go

Many generalized versions of games – especially games involving two or more players – are known to be either $\mathcal{PS}$-complete or $\mathcal{EXP}$-complete. We'll see a little about why this is the case in this section: we'll investigate the hardness of the game of Go. Now, there's a little difficulty to be found in determining the hardness of a board game of this sort, since it is generally played on a fixed-size board (e.g., Chess is played on an 8×8 sized board, and Go is played on a 19×19 size board). So we have to find a good way of generalizing the game to boards of arbitrary size. Moreover, we can make use of different rule sets when defining a game, and the different rule sets we use may result in different complexities. We'll see here that a small difference in the allowed rules for Go – which effectively determine whether we allow repeated board configurations – will determine the overall complexity of the generalized game. Go is an interesting enough game that we'll need to start by describing its rules. We'll then describe a generalization of these rules along with a reduction from Generalized Geography into this generalized form of Go. The main proofs in this section are from (Lichtenstein and Sipser, 1980).

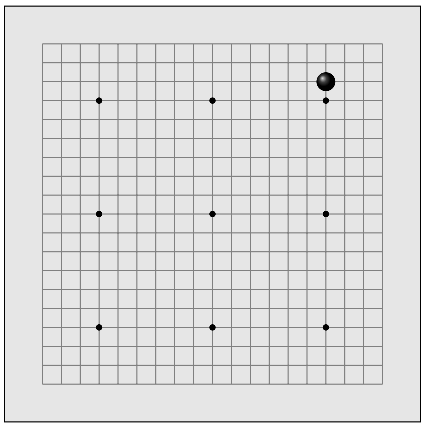
1.1 The Game of Go

The game of Go is generally played on a board that forms a 19×19 grid:

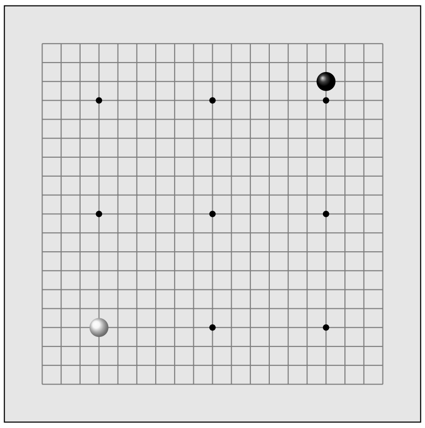


There are two players – black and white. Each player takes turns placing stones onto the board, like so:

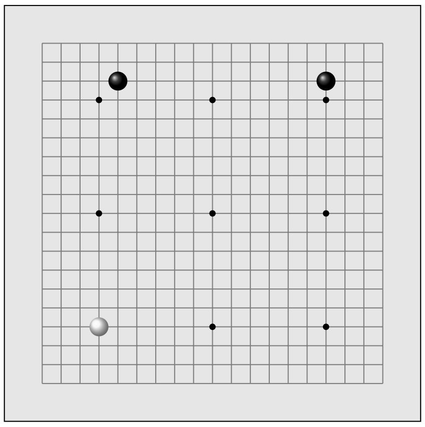
Turn 1:



Turn 2:



Turn 3:



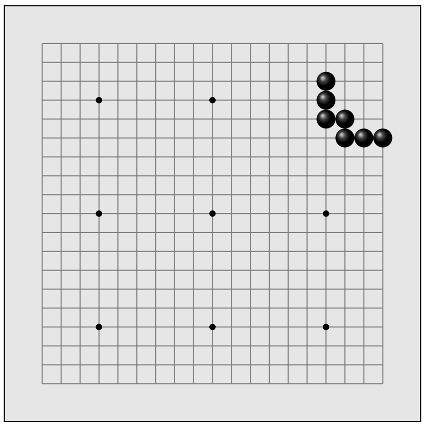
. . .

Note that black plays first, and that the stones are not moved once placed. They will be removed if captured, though, and we'll get to that soon.

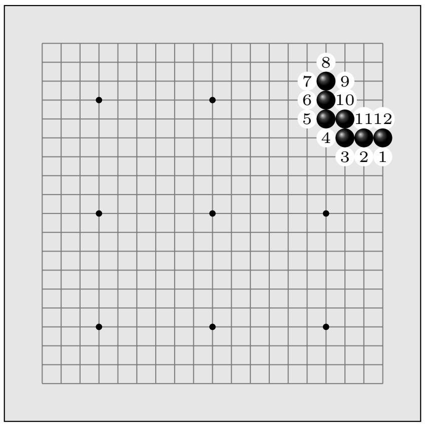
An Important Note:

In the usual rules of Go (without handicaps), it is cusomary for black to play first. But in (Lichtenstein and Sipser, 1980) the players compete on a partially filled board, and in the configurations used white plays first. We'll describe the rules here as usual for correctness sake, but also keep the proof as is. So the proof will switch the order of players.

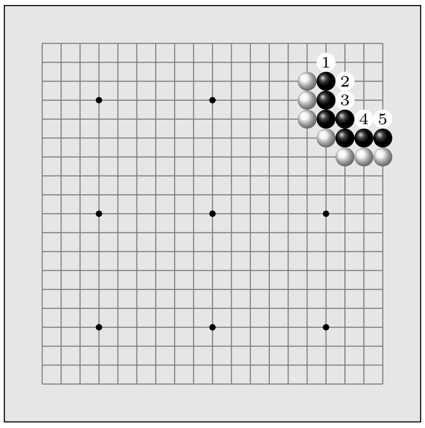
Now, if two stones of the same colour are horizontally or vertically adjacent (but not diagonally adjacent), we consider them to be part of the same block of stones. E.g., in the following board there is one block of seven black stones:



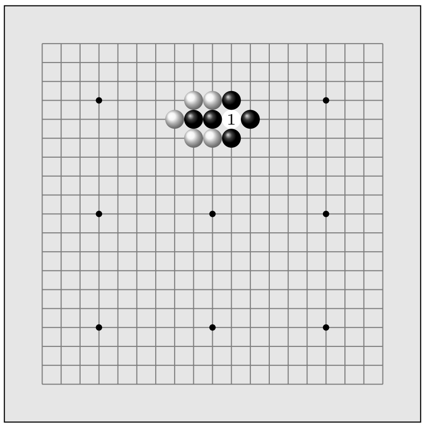
When considering a block of stones, we care about how many free spaces there are available surrounding it – the free spaces being the adjacent board spots that are not taken by any stone of either colour. We refer to these spacves as *liberties*. In our above example the black group has twelve liberties:



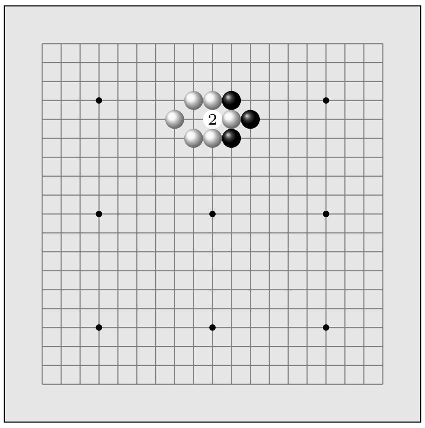
Note that we do not count liberties at the edge of the board – they have to be spaces avaiable for playing. Furthermore, of the opposing player (in this case white) places a stone into one of these liberties, it is no longer a liberty. I.e., in the following board the group of black stones has only five liberties:



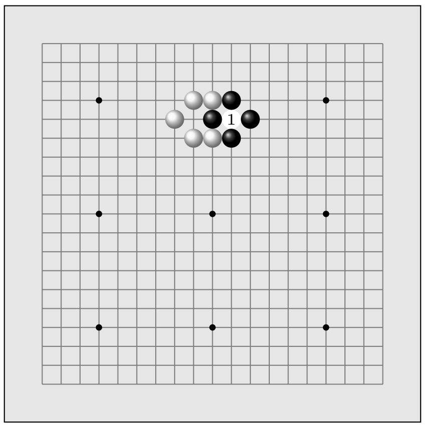
A block is captured if the opponent places a stone in such a way that the block has no remaining liberties: in the above example, if white were to play in positions 1-5 (and black made no effort to intervene), then the block of black stones would be captured by the white player. If a capturing move itself forms a block that has no liberties, the attacker takes precedence: i.e., suppose we had the following board:



If white were to play at position 1, the block of two black stones would be captured, even though position 1 initially has no liberties. Since the immediate capture of the black stones produces a liberty, the white stone survives:

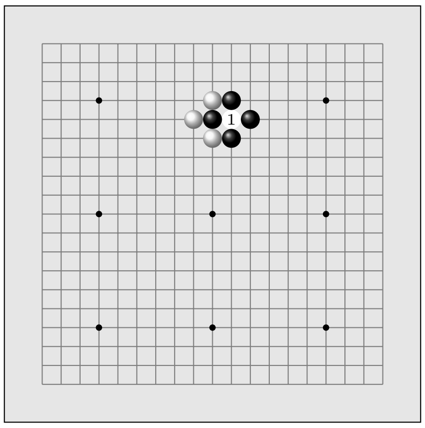


Of course, this leaves black free to play at position 2, which immediately results in the capture of the white stone:



If white were to try to play now at position 1, there would be no liberties, and neither would there be a capture. So white would lose this stone (usually this is simply not considered to an allowable move).

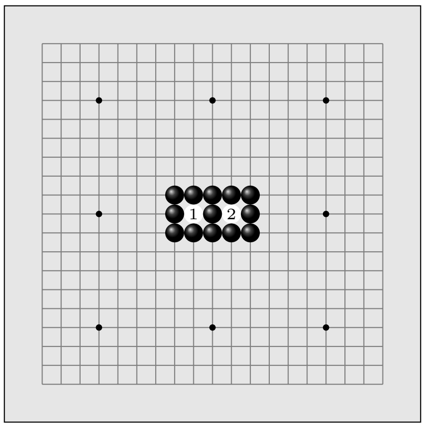
On the other hand, a very similar configuration can be used to form a loop:



This starts off very similar to the previous example, in that white can play at 1 to take the black stone, and in that black can then retake the white stone. But then we would have returned to the initial configuration. When this situation occurs – if a capture and a recapture causes an immediate repeat in a board configuration, then the recapture is forbidden for a single turn. This is referred to as the *rule of ko*. So in this example, if white were to play at position 1, black would have to wait for one turn before retaking the position.

The rule of ko prevents repeats in board position that are one turn apart, but it does not prevent longer loops: if we have three or more ko patterns on the board, a longer cycle can arise. Different rule sets for Go treat this situation in various ways, and the specific rule used will be found to determine the complexity of the overall game.

Of particular interest, we can see that there are configurations that cannot be captured, e.g.,



In order to capture this group of stones white would need to play at both 1 and 2. But this can't be done in one turn, and white can only play in these positions if it immediately captures the surrounding stones. So a pattern like this is secure for black. We will make heavy use of these patterns in the following proofs.

A group of stones is considered *alive* if it cannot be captured, and *dead* if it cannot be saved. A player's score in the game is determined by counting:

- The number of the opponent's stones that have been captured.
- The number of dead stones that the opponent has on the board.
- The number of free grid cells on the board that the opponent cannot take – where, if the opposing player tries to play a stone, the stone will be dead. These cells is referred to as a player's territory.

The game concludes when niether player can gain territory or capture stones through subsequent moves.

2 Go and Computational Complexity

Now, the game of Go as is is played on a fixed-size board, and so the number of possible games that can be played is bounded (although the actual number of possible games is, of course, too large to reasonably search). As such, its asymptotic complexity is always bounded by a constant. However, we can easily remedy this difficulty by playing the game on an $n \times n$ board, where $w \in \mathbb{N}$. We will also allow arbitrary starting configurations on the board and arbitrary starting players – this is to allow us to treat every turn in the game as a different instace of the overall decision problem. We did the same thing when we introduced the Generalized Geography game.

The issue we have to face is how we deal with the ko rules and repeated board configurations. With this in mind we'll define the following languages:

BOUNDED-TIME-GO

Definition of BOUNDED-TIME-GO:

Instance: A number $n \in \mathbb{N}$, a partially filled $n \times n$ Go board B, a starting player S in $\{$White, Black$\}$, and a time bound $K \in \mathbb{N}$, where K is written in unary.

Question: Does the starting player S have a strategy on the board B that forces a win in at most K moves?

GO-SUPERKO-RULES

Definition of GO-SUPERKO-RULES:

Instance: A number $n \in \mathbb{N}$, a partially filled $n \times n$ Go board B, and a starting player S in $\{$White, Black$\}$.

Question: Does the starting player S have a winning strategy on the board B if we do not allow the board configurations to repeat?

The superko rules of Go state that no board configuration is ever allowed to repeat in a given game.

GO-JAPANESE-RULES

Definition of GO-JAPANESE-RULES:

Instance: A number $n \in \mathbb{N}$, a partially filled $n \times n$ Go board B, and a starting player S in $\{$White, Black$\}$.

Question: Does the starting player S have a winning strategy on the board B if we require each player to take at least one turn between repeated board configurations?

The Japanese rules of Go state that in any ko battle the players must take at least one turn between repeating configurations.

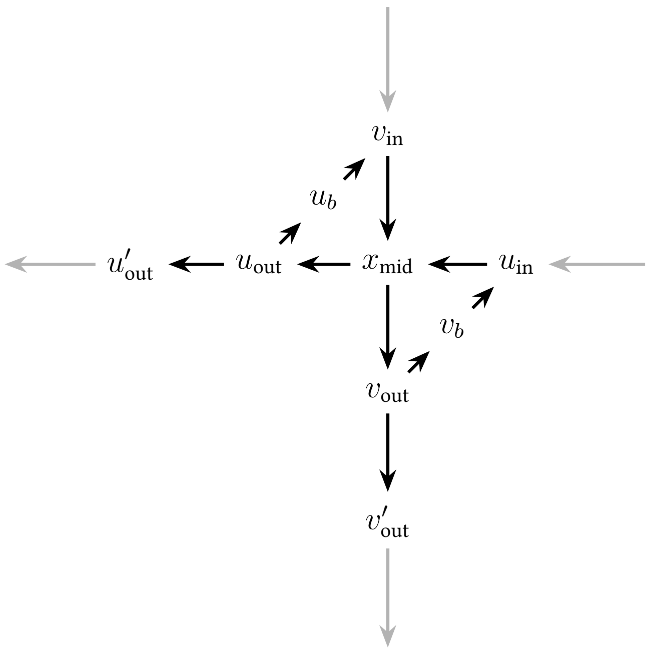
The first of these languages – the BOUNDED-TIME-GO language – can be shown to be in $\mathcal{PS}$ using a proof that is similar to the proof for GG. The GO-JAPANESE-RULES language is known to be $\mathcal{EXP}$-complete (Robson, 1983). The complexity of the GO-SUPERKO-RULES language is not known. However, all three languages are known to be $\mathcal{PS}$-hard (and so BOUNDED-TIME-GO is $\mathcal{PS}$-complete), and we'll show the reduction here. There are no repeated configurations in this reduction, and the number of moves is limited, so the same reduction will apply for each language.

We'll reduce from the version of GG in which the graph G is planar and in which no vertex has a total degree (in-degree plus out-degree) of more than three.

With this in mind, suppose that we have an instance $\langle G \rangle$ of GG in which G is planar and in which each node has a total degree of at most three.

> *In fact, we'll assume that G is the output of a reduction from a* TQBF *reduction – every node will be a player 1 choice, a player 2 choice, a junction that has two inputs and a single output, or will be a check: a node v with two entries a and b, set up so that if player 1 tries to enter v through edge b, then:*
>
> - *If player 1 has already moved through v using the edge a, then player 1 immediately loses.*
> - *But if player 1 enters through edge b and edge a has not previously been used, then player 1 immediately wins.*

Now, we have not made these ssumptions in the GG reduction we gave in class. We'll leave most of the required modifications as an exercie for the reader, but we'll note that the following widget can be used to replace edge crossings when drawing a graph:



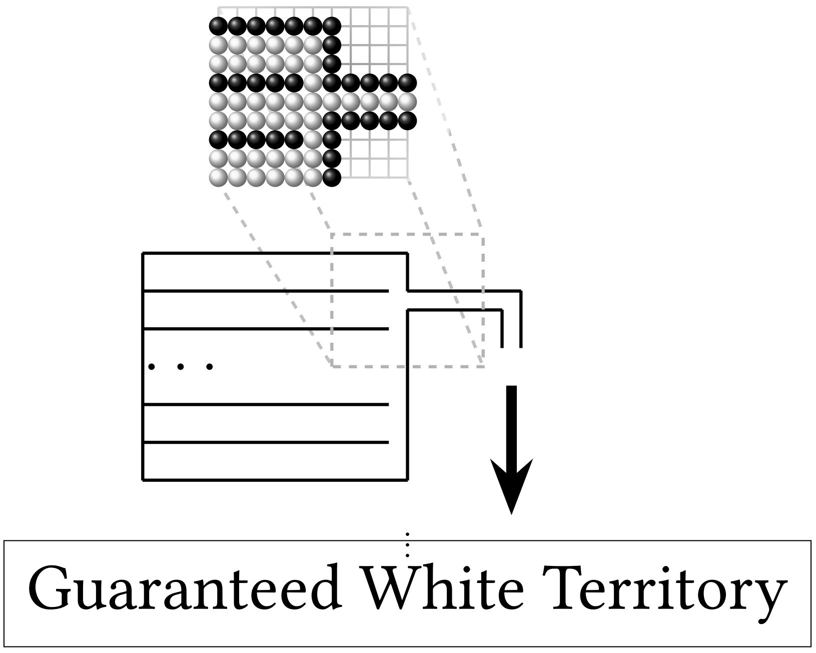
$$\begin{tikzcd}
 & & & & {} \arrow[d, gray] & & & \\
 & & & & v_{\text{in}} \arrow[dd] & & & \\
 & & & u_b \arrow[ur] & & & & \\
 {} & u'_{\text{out}} \arrow[l, gray] & u_{\text{out}} \arrow[l] \arrow[ur] & & x_{\text{mid}} \arrow[ll] \arrow[dd] & & u_{\text{in}} \arrow[ll] & {} \arrow[l, gray] \\
 & & & & & v_b \arrow[ur] & & \\
 & & & & v_{\text{out}} \arrow[d] \arrow[ur] & & & \\
 & & & & v'_{\text{out}} \arrow[d, gray] & & & \\
 & & & & {} & & &
\end{tikzcd}$$

If you investigate this widget you'll see that if the play enters one side of this widget and both players play optimally, then the play must leave the opposite side. You'll also see that each of these widgets can only be used once per game – we can't cross this junction from both entries. So finding the best place to put the edge crossings is not trivial. We note here that there is a way of doing it, though.

2.1 The Reduction

The basic idea of the reduction is to build a board in which both the black and the white players have a large guaranteed territory, but in which black's territory is larger than that of white. So unless white can obtain territory, black will win.

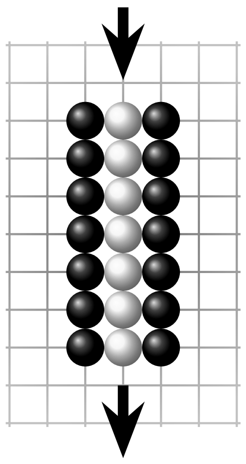
We'll set up the board so that there is very little territory to be gained, except for in one place – we'll build a large area that is contained in a black border, but is filled with white stones. The white stones will not be alive, though, unless they attach to white stones outside of the border:

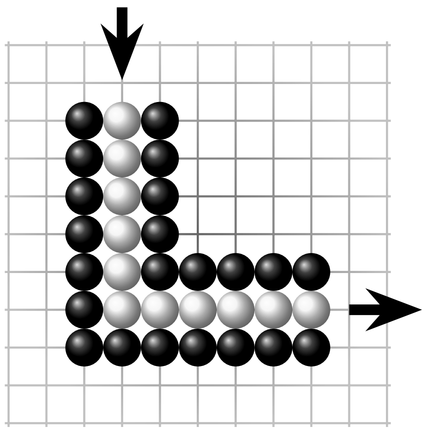


We can see that if black manages to kill these white stones, then there will be no way for white to retake any of the area in the border, and so it will all be black territory. But if white can save these stones, then black will not be able to take the territory. By making this area large enough, we can set up the board so that the outcome of the game depends on whether black can kill the white stones.

This means that the remaining game will essentially be a race between black and white: white will be trying to extend the white group to connect it to a living group, while black will be trying to black and kill the white group.

We can constrain the possible extensions for the white group so that any movement must be along "wires" on the board:

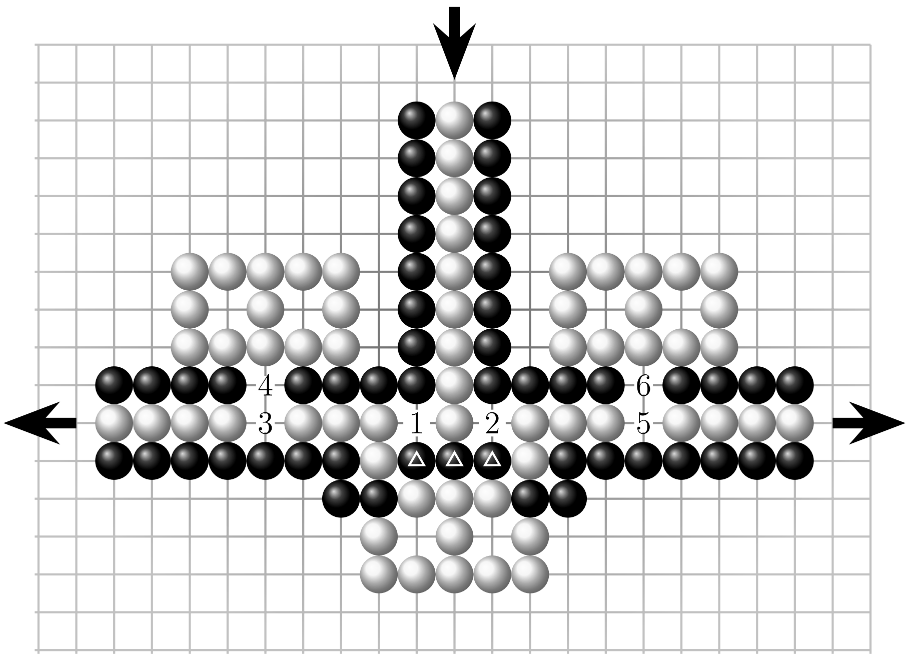




These wires will act like the edges of the graph G.

We'll build different widgets for the various types of vertices we can see on the GG board. To keep the analysis simple we'll set up our board so that if both players play optimally, then player 1 – who will be the white player in the Go game – will always play first upon entering any of the widgets.

To see what this looks like, here's a widget that encodes a player 1 choice: again, white will take the place of player 1, and so will move first upon entering the widget. This means that the in-arrow of the widget is connected to the large group white stones, and the aim is to grow the group through the widget without getting them killed.



Upon entering this widget the optimal moves for each player are as follows:

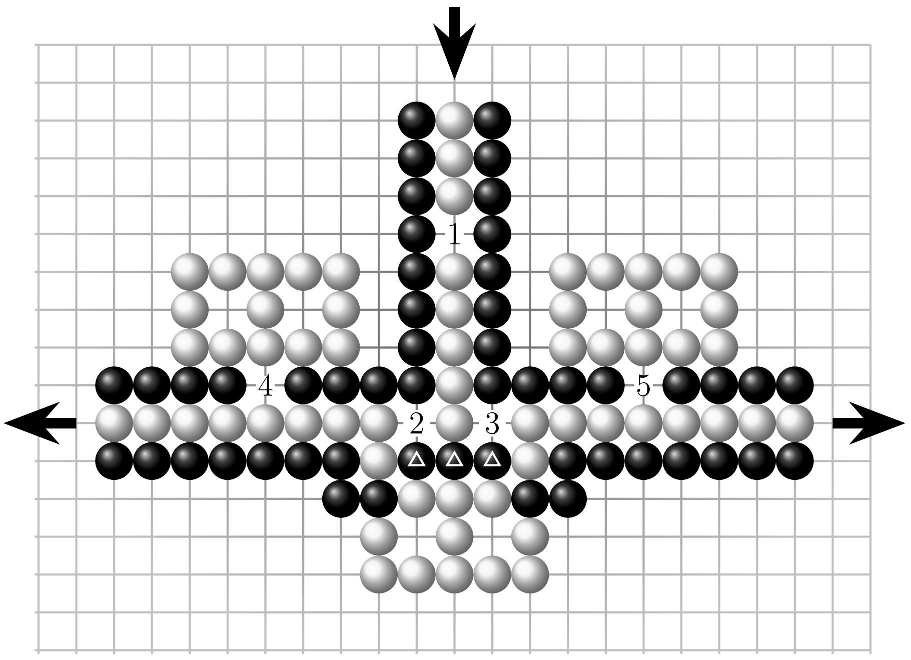
- We assert that white must immediately play in either positions 1 or 2. If white does not play in either of these spots, then the move is either in 3 or it is not.

    - If the move is in position 3, then black responds by playing in position 1. If white's next move is in position 2 then black responds in position 5 and white loses. If white does not play in 2 then black plays in 2 in the next move and white again loses.
    - If the move is not in position 3, then the situation is the mirror image: black responds by playing in 2. Again, if white's response is in 1 then black wins by playing in 3, while if white does not play in 1 then black's next move is a win by playing in 1.

  So white must play in one of 1 or 2 in order to avoid an immediate loss. We'll assume that the choice is 1, with the understanding that the rest of the analysis would simply be a mirror image if the play were at 2.

- Given that white's first move is at 1, black's response must be at 2.
    - White has two liberties at 2 and 3, and so black cannot immediately win by killing the white stones. If black were to play anywhere other than 2, then white's next move would be 2. This would kill the three marked black stones, and would guarantee a white win: if white were to play in any of the three freed spaces then the white group would connect to the living white group beneath the marked stones, rendering the whole white group alive. And black could not block all three spaces in one move.
- Once black has moved at 2, white will have only one liberty left at 3. White must play here in order to remain alive.
- Once white has played at 3, black must respond at 4 in order to prevent white from connecting to the living group above 4. At this point it is white's move, and the play moves on to the next widget.

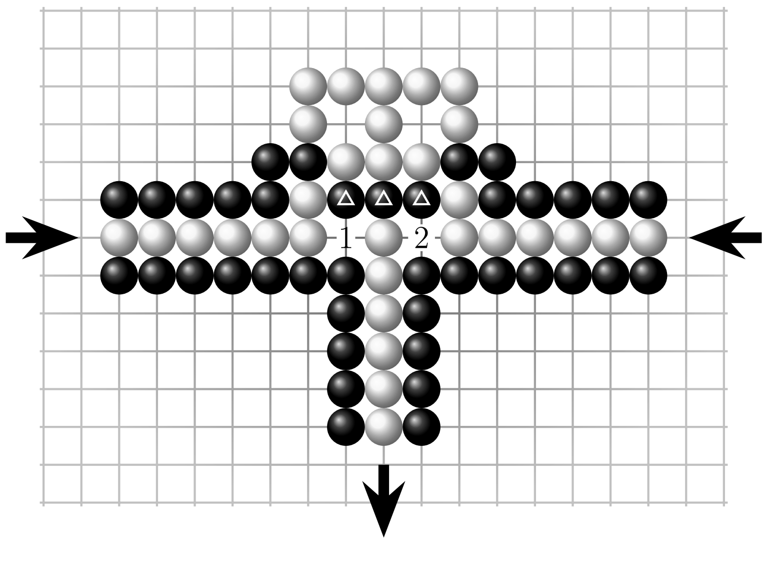
So all told, this widget allows the white player to choose the direction the path of white stones will take. The black choice widget is as follows:



If both players play optimally:

- As always, white plays first. But if the first move is not at 1, then black will respond by playing there, killing the white group. So white is forced to start playing at 1.
- Once white has played at 1, we assert that black must move at either position 2 or 3. If not, then black must either move in position 5 or must move elsewhere.
    - If black moves in position 5, then white can respond by playing at 2.
        - If black responds by playing at 3, then white wins by playing at 4.
        - If black does not play at 3, then white plays at 3 to kill the marked black stones. As in the white player widget, this will guarantee a white win, since white can then connect to the living stones beneath the marked black group.
    - If black does not move in position 5, then white can respond by playing at 3. The next steps are a mirror image of the steps taken if black does play at 5.
- So black moves in 2 or 3. Let's assume the play is at 2, with the understanding that the remaining moves are a mirror image if the move is at 3 instead. In this case white has one liberty left at 3, and so must play at 3 to live.
- At this point black must play at 5 in order to prevent white from connecting to the living group. So the play leaves the widget, and once again it is white's turn to move.

In this case we see that we can encode a choice for player 2 into the Go game. And every node with a single input ant two outputs must be one of these two possibilities – the remaining nodes must have two inputs and one output. Let's look at the simpler possibility: we have two inputs, and we want to guarantee that the single possible output is chosen. We do this as follows:

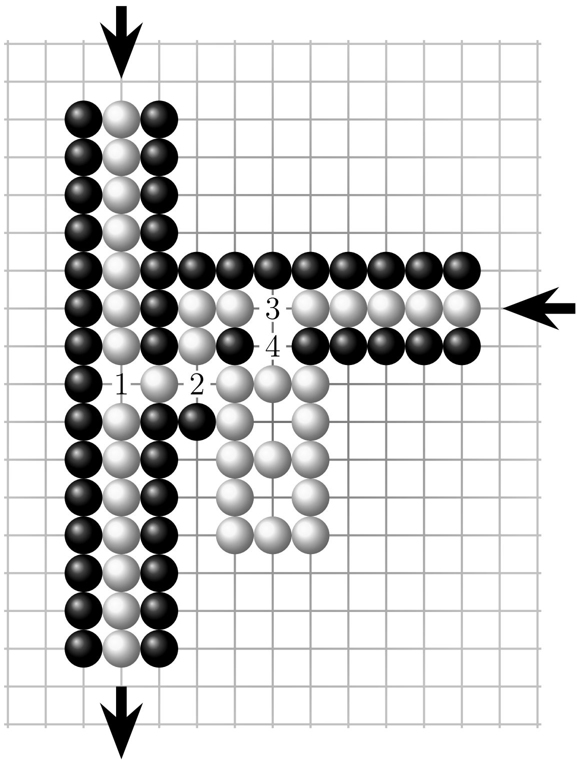


As always, white moves first upon entering the widget. This first move os forced, since there is only one liberty left for white. If the left entrance is used, white must play at 1 or lose, and if the right entrance is used the first move must be at 2.

We argue that once white has made the first move, black must play in the opposite position – if white plays in 1 black must play in 2, and vice-versa. If black tries any other move, white can take both positions, thereby killing the three marked stones. As ion the previous widgets, this means that white will be able to build a connection to the living group above these stones, forcing a win.

To finish off, we recall that the last part of the graph we built for the GG was the check: we set up a final edge $e = (u, v)$ to be followed such that, if player 1 has chosen the previous moves correctly (or if player 2 has made mistakes), v will not have been visited, but all of its neighbours will have been: player 2 will lose. But if v has been visited player 1 will lose and player 2 will win.

We'll encode this type of node into the following widget:



If there is a pass through the vertex v before the final check, we know which edge it will pass in through (assuming we're using the graph from the GG reduction). This edge will correspond to the top entrance of the widget. If both players play optimally upon moving through this widget:

- White must play at 1, since this is the only liberty left for the white group.
- Black must then play at 2 in order to prevent white from connecting to the living white group in the widget.

At this point play wil leave through the bottom of the widget with white making the next choice.

Now, suppose that the play enters this widget from the choice side (the side entrance):

- White must play at 3, since this is the only liberty left for the white group.
- If play has already passed through this widget:

    - Position 2 is already taken by black, and so white has only the liberty at 4 remaining. Since, after white moves at 3, it is black's turn, black can force a win by playing at 4 and killing the white group.

- If play has not already passed through this widget:
    - Position 2 is not taken by either player, and so it is a liberty for the white group. That is, white has liberties at both 2 and 4, and black has only one move to choose either of these liberties. Once black has made this choice, white fills the other liberty to connect to the living white group in the widget. This will render the whole white group alive, giving white the win.

We note that any plananr graph whose vertices has a total degree of three can be embedded in a grid, and so can be encoded, with a polynomial blowup in size, into a suitably large Go board that uses these widgets. So we can reduce GG to any of our three Go-based langauges, and so Go is $\mathcal{PS}$-hard.

> *Of course, the particular type of board setup that we use in this reduction is not likely to occur in any real game of Go, but we have at least demonstrated that the difficulty is there.*

References

David Lichtenstein and Michael Sipser. Go is polynomial-space hard. *Journal of the ACM (JACM)*, 27(2):393–401, 1980.

John Michael Robson. The complexity of go. *Proc. IFIP, 1983*, 1983.