This is your TA: there are some interesting points made in this review. As there are multiple different ways that you can approach this model, it is up to you to decide if you wish to change the structure.

Hi Group 14, it looks like you have had a great start so far on the modelling of Mancala. I want to start off by addressing the model exploration part of the report, where you have multiple strategies to find the best move. If I were in your place, I would think of each as their own problem, and work towards solving each one separately. There are a few strategies that I would like to comment on:

1. Putting the final seed in the player's store

I see that you have made progress on this through your constraints $H_{0, A, 0} \rightarrow T_A$ and $H_{g, r, c} \rightarrow H_{g-1, r, c-1} \wedge P_{g+1, r, c-1}$. Something to think about though is if there are 13 or more gems in a pit, you will end up having two different P propositions true for the same pit. For example, if the player picks up gems from 14 gems from A1, we have $H_{13, A, 1}$, and since the pit is on player A's side, S_A. By implication elimination with your first constraint, $P_{13, A, 1}$. Since $H_{13, A, 1}$, by implication elimination with your fourth constraint, $H_{12, A, 0}$ and $P_{12, A, 0}$. We can continue using implication elimination with the fourth constraint until we have no more gems, but you will see at the end that since $H_{0, A, 1}$ and S_A then $P_{0, A, 1}$. While these constraints do make T_A when the final gem is dropped in player A's store, we run into a problem where a pit at r, c has two different amounts of gems in it which intuitively should not happen. This might cause some problems while you are programming your Python implementation, where you will not be able to say that there can be at most one value for H at a certain pit r, c. What happens then is that it might be hard to restrict $H_{0, A, 0}$ so that it is only true when you want it to be true. As of right now, if these constraints were the only ones in your implementation, nothing is stopping the SAT solver from stating that $H_{0, A, 0}$ and therefore T_A.

An alternative way to approaching this is to skip the middleman in the way that you don't need to simulate the "sowing" of seeds by implying $H_{g, r, c}$ in a circular motion, and instead skipping to the conclusion T_A. For example, you could have a constraint $H_{6, A, 6} \rightarrow T_A$ which skips having to be careful around the other pits. To put this more formally, $H_{a + 13b, A, a} \rightarrow T_A$ for any $0 \leq a \leq 6$ and any $b \geq 0$.

2. Block the opponent from putting a seed in their store

In order to implement this strategy, you would need to find a way to remove all instances of $H_{a + 13b, B, a}$. One way to do this is to choose a pit such that sowing seeds "passes by" the pit which leads to the store. This might be difficult to implement though as you will need to watch out in the case that you sow in a pit that would have led to pit B1. This would increase the number of seeds in the pit by one and give player B another option to choose from in order to put their final seed into their store. For example, consider a board with one seed in B1 ($H_{1, B, 1}$), one seed in B2 ($H_{1, B, 2}$), and two seeds in B3 ($H_{2, B, 3}$). If you tried to block player B from using B1 to put their final seed into their store, you would have to "pass by" B2 and B3, also incrementing their number of seeds by one ($H_{2, B, 2}$, $H_{3, B, 3}$). This inadvertently creates two options for player B to put their final seed into their store instead of one, making it a bad move. You might be able to solve this problem by counting the number of possible ways for player B to put their final seed into their store before and after the potential move and choosing that one only if the number of options for player B decreases.

For the propositions, S_r and O_c may be unnecessary. To check if a pit is on player A's side, look at the r subscript of P. If it is A, then it is on player A's side. Checking if two pits are opposite of each other does not require a separate proposition and can be done with some math. It looks like you have already been able to use pits that are opposite of each other judging by your last constraint $(H_{0, A, c} \wedge P_{1, A, c}) \wedge \neg P_{0, B, 7-c} \rightarrow C_{g, B, c}$ with the 7-c. Other than that, there is a small typo in the H proposition, where it should be "g gems" instead of "> 1 gems."

If you consider my advice for removing S_r, you will see in the constraints that $H_{g, a, c} \leftrightarrow P_{g, A, c}$ which is not true. For example, if player A has one seed in hand when sowing over their store, that does not imply that there is one seed in the store. Similarly, if player A has one seed in their store, that does not imply that there is one seed in their hand. It may be worth rethinking the relationship between $H_{g, a, c}$ and $P_{g, A, c}$ if one even exists.

The Jape proofs you have chosen do a great job of giving an example of various strategies. The one thing I find a bit odd about them is that there are premises in the proofs that do not appear in the constraints. For example, in the first proof we have the premise $S_{A, 4} \rightarrow H_{3, A, 3}$ which is not seen elsewhere in the report. I think that some explanation of the premises is necessary in order to understand what is being proven.

The first-order extension has been well explained, one thing you can add is the case that there are no seeds in player A's pits. Since the board is randomly initialized, there is a small probability that player A will not be able to make any moves. You can then put that if there does not exist any seeds on player A's side, then handle it accordingly.

For the requested feedback, I believe that it may be outdated as the logic model counts the exact number of seeds in hand using the $H_{g, r, c}$ proposition and that there are more than four constraints. I did touch on logic errors and constraint ideas which have been explained above. While I am unable to answer the third question about project complexity, I am interested in seeing how you will implement all this in Python!