Model Exploration

Initial Model

The original idea is just modelling the game of Tetris, however with feedback from Prof. Muise we were informed that we need to include the element of time but remove “sliding” and “T-spin” like moves. After the initial modeling we should attempt to explore different ways the SAT solver can work. (random moves, minimize height, etc..)

Propositions of pieces were originally numbers 0-19r,c with 0-3 as Z piece and so on, now we have decided to separate Tetris pieces into different propositions with the orientation, position and time.

Originally location had row and column, however to include time we will replace row location with time and have the block constantly move down with time.

Initial Steps:

In the beginning we had a very rough idea of how to model tetris, majority of the ideas in this model were scrapped after feedback from our peers and very little or none was brought into the final implementation.

* Tetris Blocks not seperated
* 0-19 with subsets denoting tetris piece types
* Many different propositions
* Rowclearing
* Blocked
* Fail
* Constraints
* Rotation Locking
* Faling piece
* Pieces cant be placed in same place
* No universal gravity, placed blocks float

Overall our initial model was from perspective that attempted to model tetris in a way that a programmer trying to program the game tetris itself. This can be seen in the way we were attempted to thinking about rotations in the present rather than viewing the rotations as a set of possible placements.

Intermediate Models

After attempting to create a SAT solver we realize that we were generating propositions and constraints in a way more appropiate for coding a game. We then adjusted propositions and constraints to suit the bauhaus library model better. To do this we knew we needed to reduce the propositions. To do this we only focused on the core of tetris which related to the tetris pieces themself. Everything else was dropped from the propositions. During the intermediate stages of our model exploration narrowed down proposition(s) changed a few times.

2. TetrisPiece(roundNumber, type, orientation, location, time)

This idea we only had one massive proposition that was meant to consider everything in one fell swoop. This may have been possible to properly implement however this model was a first attempt at implementation and was made with familiarizing ourselves with the encoding tools we were provided.

2. PieceConfig(type, orientation)
3. PiecePlacement(piececonfig, location, time)

With more familiarity with the encoding process, we decided to reference Prof. Muise’s Kaoodle project as the two games are similar. We wanted to see what parts of kanoodle could be shared with tetris, through this we were able to develop a working model of tetris.

2. TetrisPiece(type,rotation,x,y,time)
3. Cell(x,y,time)

After learning a bit about how Prof. Muise modeled kanoodle we figured out some elements that we could share. We then decided on the two propositions above, adding the element of time through the existence of rounds we decided to have the propositions representing the tetris piece in play on a given round and the state of each cell on a round. With the cell propositions we figured out that we could represent all previous rounds, in a sense the initial model’s blocked proposition evolved into our cell proposition.

Final Model

Propositions:

* TetrisPiece(type,rotation,x,y,time)
* Cell(x,y,time)

Constraints:

* There must be only one tetris piece placed in each round
* For all pieces, when placed they will then occupy those cells
* The piece must “fall” to the bottom of the grid
* No piece can be placed out of bounds
* For a cell to be occupied, a piece must have been placed in that location

After learning how to use the bauhaus library and experimenting with different propositions, we eventually settled on a model we though could work. We believe these are the two basic propositions needed to model tetris. With this we could now create constraints that would form the rules of tetris, before this we simply had a model which could place tetris pieces on a grid without rules, meaning that they could overlap, go out of bounds, phase through other pieces and so on. The constraints are meant to ground this model to the rules of tetris.

Model Counting(?)

There were different constants that we tried to use:

* 3x2, 3x3, 4x3, 4x4, 5x5
* 1 round & 2rounds

We found that with bauhaus and the docker desktop that we were exteremly limited in the solutions we could count. In a linux enviornment the maximum board size we could have without the program crashing was 4x3 with 1 round or 3x3 with 2 rounds.

Debugging

At some point we noticed that there we an absurd number of solutions despite having an extremely simple model. For example an O piece in a 3x2 grid would result in 52 solutions. The issue was a bug in the code which ended up being:

Text

Description automatically generated

The bug was that a generator was used on line 126 instead of a list, which when the programe attempted to iterate over the generator it picked up where it left off every time instead of starting from the beginning as intended. To fix this we changed the generator to a list so we could iterate over all the values more than once. This resolved the bug and the same test example resulted in the correct 2 solutions.