# **2048**

# 20 (Slightly) Different 2048 VersionsProject Summary

The idea of the game of 2048 is that you move and merge tiles with equivalent values to generate tiles with larger values. The grid on which you can move the tile is a 4 \* 4 grid. Every time you can move in one of the four directions: up, down, left, right. Once a tile with value 2048 is being created, you win; if there is a deadlock, which means there are no possible ways to further make a move or merge, you lose the game. We would like to model the logic behind this game. Particularly we would like to model the mechanics of movements. We will be diving into the status of each tile before and after movements, whether they can move, and toward which direction they are moving.

# Propositions

TODO

* Bring in the figure for the edge labels
* Make sure they are numbers and not letters
* Finish rewriting to mirror the names used in the code

Draft:

# location (tile, loc): tile t is at location loc, the location is where on the board that things exist

# adjacent(tile, orientation): check if there are adjacent tiles near t\*

# empty (tile, orientation): check if adjacent places are empty

# is\_empty (loc): check if a location loc is empty

# can\_merge (tile, orientation): check if the tile can merge

# row\_can\_move (row): check if at least 1 move can be made in that row

# column\_can\_move (column): check if at least 1 move can be made in that column

# able\_2\_move (tile, orientation): check if at least 1 move can be made on the given orientation

# random (loc): after a move, a random new tile is generated at an location

# get (nextTile, orientation): check the next tile in the given orientation, if the next tile is of the same value, return true

# row\_or\_column\_can\_move(orientation): check if a row or column can move

# Constraints

TODO:

* Go over the proposal constraints and refine for what’s currently done
* Add the new constraints that came into the project since then
  + Just summarize the super simple ones (e.g., no self loops)

Draft:

* No two tiles can be put to the same location. For tiles t1 and t2 with location loc:
* location(t1, loc) \/ location(t2, loc)
* If a move can be made and there the next tile in the given orientation is as of the same value. then they can merge.
* get (tile, orientation) /\ able\_2\_move (orientation) can\_merge (tile, orientation)
* no new tile can be generated on a not empty location
* ¬ is\_empty (loc)  ¬ random (loc)
* If the next tile in the given orientation is as of the same value, if the next tile in the direction is empty, then we can move.
* get (nextTile, orientation) \/ empty (tile, orientation)  able\_to\_move (tile, orientation)
* If at least 1 out of 4 tiles on the same row or column (depending on the orientation of movement, i.e. we use row if move left/right, use column if move up/down) can move, we are able to make at least 1 move in that orientation
* able\_to\_move (t1, orientation) \/ able\_to\_move (t2, orientation) \/ able\_to\_move (t3, orientation) \/ able\_to\_move (t4, orientation)  row\_or\_column\_can\_move (orientation)
* Only 1 tile is generated randomly amoung the all the empty locations.
* IDK how to write it

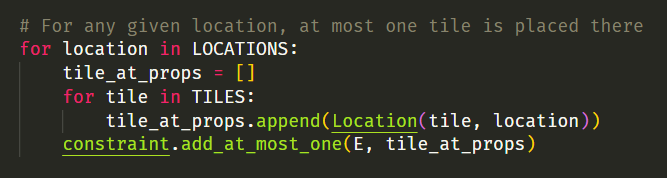
# Model Exploration

*List all the ways that you have explored your model – not only the final version, but intermediate versions as well. See (C3) in the project description for ideas.*

## Fixing the Bug With More Tiles Than Locations

When I started adding more tile possibilities, I found that 5 tiles happened to be placed on only four locations (2x2 grid). This *shouldn’t* be possible. This came about with 5 tiles, and just 4 locations, and looking at the final solution, I found that two separate tiles were assigned to the same location.

This immediately raised the fact that I was missing a constraint saying that at most one tile appears at a location. I fixed this by adding a constraint that looks like:



So for every location, we have this constraint added that says “at most one tile can be there”. Originally, I thought this might have to be “exactly one”, but then decided that I want to allow for some locations to have no tile placed.

After adding this constraint, I confirmed that having 5 tiles and 4 locations was unsolvable.

## Ensuring no Location Connections When No Tile

We wanted to make sure that locations don’t require a tile to be there. But leaving the connections on a location be free to be true/false, means that we might have phantom connections created. To test this theory out, I created an example that I would hope leads to no solution:

## A screen shot of a computer code Description automatically generated

Note that the 3 tiles in the setup are placed at 11, 12, and 21, and that we’re trying to have a location connection on the 22 location. This happened to be solvable, because there was nothing forcing these propositions to be false when no tile was put down. The way I resolved this was to use the following constraints:

A screen shot of a computer program

Description automatically generated

This was also the first time that I had to start using the And and Or functionality. After this constraint, the test case led to an unsolvable theory, just as I had hoped.

# Jape Proof Ideas

*List the ideas you have to build sequents & proofs that relate to your project.*

Haven’t started on my Jape proofs yet, but here are some ideas I might try in the remaining few weeks of the course:

1. For a single tile, if there are links 1-2, 3-4, and not 5-6, then 5 must be connected to 7 or 8.
2. If I can get to a location in 1 step, I can’t get there in 2.
   1. Perhaps, if location distances are unique, then if you get there in 3, you can’t get there in 4
3. Stretch goal: there is no way to get to a location in 2 different distances

# Requested Feedback

1. *How are the propositions and constraints of our model? We have been told that we need to focus more on the mechanics of tile movements on the feedback we received earlier, and we have tried hard to modify our propositions and constraints to make them more valid.*

# First-Order Extension

*Describe how you might extend your model to a predicate logic setting, including how both the propositions and constraints would be updated.* ***There is no need to implement this extension!***

Nope, just haven’t started this yet.

# Useful Notation

*Feel free to copy/paste the symbols here and remove this section before submitting.*