Modelling of Complex Systems IDP Project – Part 1

senne.berden@kuleuven.be dorde.markovic@kuleuven.be

March 2023

Due before **Monday April 1th 23:59** on Toledo.

1 Introduction

In this part of the project, you are tasked to write a formal specification of a maze, either alone or in a team of two. A small starter vocabulary with several predicates will be given to you. Your job is to extend this vocabulary, and use it to write a theory such that only valid mazes are models of the theory. What constitutes a valid maze will be specified later on in this document. First, we will clarify some terminology, and introduce the starter vocabulary for you to work with.

We will work in an $n \times n$ grid made up of cells. There are two top-level type of cells: empty cells and walls. Each empty cell is either a regular empty cell, the entrance, or the exit. Each wall is either an outer wall or an inner wall (although this distinction is merely conceptual, meaning that there is no separate predicate for inner and outer walls in the starter vocabulary).

This leads to the following starter vocabulary:

```
Vocabulary V_maze_fixed {
    type X isa nat
    type Y isa nat
    type Pos constructed from { P(X, Y) }

    Wall(Pos)
    Entrance : Pos
    Exit : Pos
}
```

with the following intended meaning:

- **X** is the set of X coordinates.
- **Y** is the set of Y coordinates.
- **Pos** is a type constructed from X and Y coordinates. The origin is located in the bottom-left corner of the grid. To refer to a cell with coordinates X_i and Y_j in your theory, use $P(X_i, Y_j)$. For example, to refer to the bottom-left square, use P(0, 0).
- **Wall($P(X_i, Y_j)$)** denotes that a static wall is located at position $P(X_i, Y_j)$.
- **Entrance = $P(X_i, Y_j)$** denotes that the entrance to the maze is located at $P(X_i, Y_j)$.
- **Exit = $P(X_i, Y_j)$** denotes that the exit of the maze is located at $P(X_i, Y_j)$.

A valid maze must abide by the following rules.

- The position of the start and end cell are fixed (relative to n, which specifies the size of the maze): the entrance is always positioned in the bottommost row (blue block in figures), second column; the exit (green block in figures) is always positioned in the topmost row, second-to-last column. The Y coordinates corresponds to rows and X to columns.

- All cells in the leftmost and rightmost columns, as well as all cells in the bottommost and topmost rows, are walls, except for the entrance and exit cells which never contain a wall. We will refer to these walls and only these walls as the outer walls. All other walls are inner walls.

- From each inner wall, an outer wall can be reached through a connection of neighbouring inner walls (where the neighbouring cells of a cells are the adjacent cells to the left, right, top or bottom of that cell; diagonally adjacent cells are *not* considered neighbours).

- Each empty cell can be reached from the entrance through a connection of neighbouring empty cells (where the neighbouring cells are defined as in the previous bullet point).

- The grid does not contain a 2×2 block of walls.

- The grid does not contain a 2×2 block of empty cells.

Next, we give some examples of valid and invalid mazes.

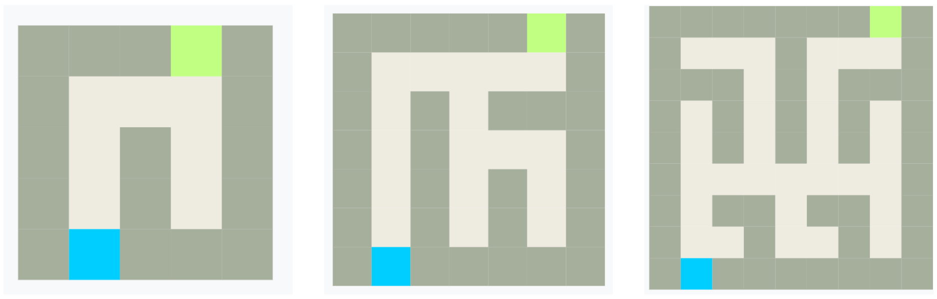

Figure 1: Some examples of valid mazes, for different values of n.

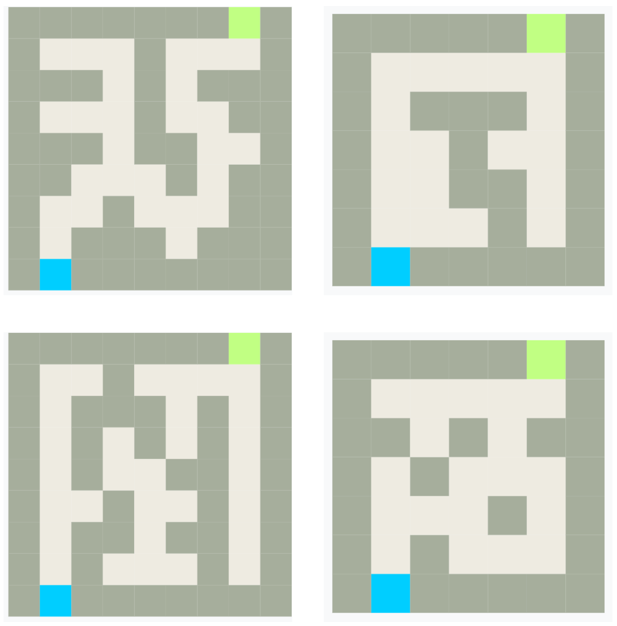

Figure 2: Some examples of invalid mazes. The maze in the top-left panel is invalid because there is at least one 2×2 block of walls. The maze in the top-right panel is invalid because there is at least one 2×2 block of empty cells. The maze in the bottom-left panel is invalid because there is at least one empty cell that is not reachable from the entrance through a connection of empty cells. The maze in the bottom-right panel is invalid because there is at least one inner wall from which an outer wall cannot be reached through a connection of inner walls.

2 Tips and tricks

- The IDP IDE supports autocompletion (CTRL+SPACE); use this to save yourself some time.
- It is often helpful to introduce new types, predicates and/or functions in the student vocabulary, and to then define and use them in the student theory. Introducing new symbols can make your theory more clear, more modular and easier to write and extend.
- The generation of a maze can take a long time for large grids. For this reason, you are encouraged to experiment with your specification on small structures initially.

3 Practical

3.1 Provided files

On Toledo you can find a `skeleton.zip` file containing:

1. An IDP skeleton file `maze.idp` consisting of:
   - A fixed vocabulary *V_maze_fixed* (**You cannot add anything to this**)
   - A student vocabulary *V_maze_student* extending the fixed vocabulary (This is the place to add extra types, predicate and/or function symbols)
   - An empty student theory *T_maze_student* over the student vocabulary (You need to add sentences to this)

- The `main` procedure that searches for one model and visualizes it. In order to try different scenarios you may alter the structure provided to method *onemodel*. Changing other parts of the main procedure is allowed for debugging purposes, but make sure to correctly restore it once you are done.

2. The directory `tests` containing two files `correct-mazes.idp` and `incorrect-mazes.idp`, which contains various structures over *V_maze_fixed* vocabulary. Each structure specifies valid or invalid maze. You should use these to test your maze formalization.

3. The `visualize.idp` file and the `idpd3` directory, which are needed for the visualization and which you may not edit or move.

3.2 Project output

The output of this project must consist of a `.zip` file. If you are working alone, the zip file should have a name of form:
`Lastname_Firstname_StudentNumber`
(where Lastname and Firstname are replaced by your last and first names, and StudentNumber is replaced by your student number, starting with *r* or *s*). If you are working in a team of two, the zip file should have a name of form:
`Lastname1_Firstname1_StudentNumber1_Lastname2_Firstname2_StudentNumber2`
The zip file should only contain the *maze.idp* file. In case you work in a team of two students, make sure that **only one team member submits the zip file**.

3.3 About the specification

- You have to start from the provided `.idp` files.
- You are **not allowed** to change the fixed vocabulary at all (not even renaming).
- You are allowed (and advised) to add new symbols to the student vocabulary. You are also allowed to add extra types.
- **Write comments**: clearly specify the meaning of every symbol you have added in your vocabulary and every line you have written in your theory.
- Use well-chosen names for introduced symbols and variables.
- Make your specification clear and readable.
- Use a consistent form/layout in your specification.
- Use consistent indentation.
- Avoid overly long logical sentences.
- When quantifying a variable, always specify the variable's type.
- Make sure there are no warnings and errors, except for the *Verifying and/or autocompleting structure* warnings and the errors on including files.

3.4 Grading

This part of the project will be evaluated through both manual inspection and automatic tests. Manual inspection will consider choices made in vocabulary and whether standard patterns for definitions taught in the exercise sessions were followed (e.g., transitive closure definitions). Automated tests will check whether model expansion leads to valid mazes. Other tests will assess whether valid mazes are indeed models of your theory. In total this part of the project corresponds to 1 point.

3.5 Teamwork

You can make this project either alone, or in a team of two students. You do not need to explicitly register your team. In case you work in a team, *exactly one team member* should submit a file for your team, following the naming convention specified in section 3.2. If you want to work in a team and are looking for a teammate, you are free to use the discussion forum named "Group formation IDP project part 1" on Toledo.

3.6 Restrictions

You are allowed to discuss this project with others, but are not allowed to copy any code from other students/teams. This is not an open source project, i.e., you are not allowed to put your code openly available on the web. If you want to use Git, do not use public GitHub repositories.

3.7 Discussion boards

You can use the discussion forum "Group formation IDP project part 1" to help in finding teammate. You can use the discussion forum "Questions about IDP project" (Toledo → MCS course → discussion board) to post your questions and to discuss issues with your colleagues and teaching assistants. In this way, you can find answers to your questions faster and save some time. You are allowed to include IDP code in your questions if desired (e.g., to illustrate the problem you are facing), on the condition that this code is **not related to this project**. In other words, if you post IDP code on the discussion forum, this code **may not be related to the maze project**. If you are not sure whether your code is allowed to be added to a question on the discussion board, first send your code and question to the teaching assistants via e-mail. In case you have purely technical questions regarding the IDP3 system, consider posting it on StackOverflow using the "idp3" tag.

Good luck!
In case you have any questions, do not hesitate to contact the teaching assistants.