MCS Project Part 1: Saving Arkham

Laurent.Janssens@cs.kuleuven.be
Ruben.Lapauw@cs.kuleuven.be

November 4, 2015

> Arkham is a small town plagued by other dimensional beings. These were stuck in their own dimension, but have figured out how to open gates to Arkham, allowing them to enter our world.
>
> You are an investigator, tasked with... investigating the events taking place in Arkham. Arriving in Arkham it quickly becomes clear the situation is dire and needs immediate attention. Your sources tell you that if too many gates are opened at the same time our dimension and theirs will collapse onto each other. Surely, this is a bad thing. Only you can prevent this, if you manage to close enough of these gates.

1 Introduction

The project consists of two parts:

- In this first part, you model the game in IDP: The investigator travels around town and can attack monsters and close gates. This should be done in the LTC formalism. We use this theory to verify some properties of the game.
- In the second part (in about a month), you model this same game in Event-B and use CTL and LTL to verify other properties of the game.

In Section 2 we describe the game in general. Section 3 explains what you have to do for the first part of the assignment. The project for this course must be completed in groups of two students. You are expected to work with the same person on both parts of the project!

2 The game

For this year's project you are tasked with modelling a game, where an investigator is trying to save a town by attacking monsters and closing gates. The game consists of a number of key concepts:

- The town

- The investigator
- Monsters
- Gates
- Clues
- Game turns

The will briefly explained in the following subsections.

2.1 The town

The town of Arkham is divided into several districts. Districts can be connected to one or more other districts. All buildings are divided across these districts, i.e. they are only ever connected to a single district. However, one district can be connected to multiple buildings. These connections are obviously symmetrical.

2.2 Game turns

The first turn of the game is a player turn, after that player turns and monster turns alternate unless the game is won or lost. Once the game is won or lost it stays that way. The game is won if there are more than the specified amount of gates closed at a certain time. The game is lost if there are more than the specified amount of gates open at a certain time or if the investigator can not move when he is supposed to.

Thus, if after a monster turn the investigator can not move, the game must already be lost at that time, not the next! The same applies for open and closed gates.

2.3 The investigator

The investigator can move through the town, by moving from one location to a connected location, during a player turn. While moving through the town he can perform one of two actions, he can attack monsters or close a gate. He can only attack when there are monsters in the same location as him and can only close a gate if he is in a building with an open gate. He can not attack and close a gate in the same turn. He is allowed to do neither.

The investigator is obligated to move every player turn. He can only move to locations which are connected to his current location and in which there are strictly less than three monsters. He can only attack and close gates during player turns.

2.4 Monsters

The monsters that exit the gates also roam around Arkham. They can only move during a monster turn, and all monsters must move during such a turn.

Just as the investigator monsters can only move to a location which is connected to theirs.

The monsters do not like the indoors and thus stay in the districts, i.e. they never enter buildings. The only way they end up inside of buildings is by coming through a gate. There can be up to three monsters in any district. When there are open gates, one monster comes through every gate, every monster turn.

Luckily the monsters are not very strong. Attacking them once is enough to send them back to their dimension. When attacking, an investigator kills all monsters in his location.

There are situations where it is impossible for all monsters to move, you can ignore these situations.

2.5 Gates

Every now and then a gate will open in one of the buildings. (This happens on predefined times, specified in the structure). Gates can be closed by the investigator. they stay closed, unless they are specifically opened again. When closed gates reopen they no longer count as closed. Equivalently open gates stay open until they are closed.

The investigator can only close a gate if there is an open gate in his location and he has at least two clues.

2.6 Clues

The investigator receives one clue per monster he kills and must use two clues to close a gate, i.e. if he has one clue at time t and kills two monsters, he will have three clues at time t+1, equivalent when closing gates. The investigator must have two clues to be able to close a gate. The investigator has a maximum of ten clues, any extra clues gained after that are lost.

3 Part 1: Modelling the game in IDP

You are **obliged** to model this assignment in the LTC formalism! In order to model this, you use (an extension of) the following vocabulary:

```
LTCvocabulary Arkham {
    type Time isa nat
    Start : Time
    partial Next(Time) : Time

    type location
    type district isa location
    type building isa location

    type number isa nat
```

```
    Move_To(Time, location)
    Close_Gate(Time)
    Attack(Time)

    Location(Time) : location
    I_Location : location

    type monsternb = {0..3} isa nat
    Monsters_In(Time, location) : monsternb
    Exit_Gate(Time, building)

    Clues(Time) : number

    Open_Gate(Time, building)
    C_Open_Gate(Time, building)
    Closed_Gate(Time, building)

    Has_Connection(location, location)

    type state constructed from {player_turn, monster_turn,
        won, lost}
    GameState(Time) : state
    Closed_To_Win : number
    Open_To_Lose : number

}
```

It is permitted, and advised, to extend this vocabulary with new predicates or functions! You **cannot** change the names, arity, types, ...of the given predicates, types and functions, since our automatic verifications are based on these properties.

The meaning of the given symbols is as follows

Time represents the time in your domain;

Start is the initial time point;

Next(t) is the successor time of t;

location represents the locations in the town;

district is the set of districts in Arkham;

building is the set of buildings in Arkham;

number is a range of natural numbers used for counting clues;

Move_To(t, l) is true if and only if the investigator moves to location l at time t;

Close_Gate(t) is true if and only if the investigator closes a gate at time t;

Attack(t) is true if and only if the investigator attacks at time t;

Location(t) is the location of the investigator at time t;

I_Location is the starting location of the investigator;

monsternb is the set of numbers used to represent how many monsters are in a location;

Monsters_In(t, l) is the amount of monsters present in location l at time t;

Exit_Gate(t, b) is true if and only a monster comes through the gate in building b at time t;

Clues(t) is the number of clues the investigator has at time t;

Open_Gate(t, b) is true if and only if there is an open gate in building b at time t;

C_Open_Gate(t, b) is true if and only if a gate opens in building b at time t, i.e. it will be open at time $t + 1$;

Closed_Gate(t, b) is true if and only if there is a closed gate in building b at time t;

Has_Connection(l1, l2) means location $l1$ is directly connected to location $l2$, and vice versa. The information retained in this predicate is complete enough to know which locations are directly connected, i.e. if $l1$ is directly connected to $l2$, either Has_Connection(l1,l2) or Has_Connection(l2,l1) is true. It might be a good idea to define a new relation, which does represent this symmetrical aspect. (**Tip:** it really is a good idea)

state is the set of all states the game can be in;

GameState(t) is the state of the game at time t;

Closed_To_Win is the number of gates which must be closed at one time to win the game.

Open_To_Lose if there are this many gates open at one time you lose the game.

On Toledo, you can find a skeleton and a couple of test-structures that test whether your theory correctly models certain aspects of the assignment. You can run all tests we provide by executing the method "check" in `instances.idp`. These test scenarios are non-exhaustive, so please read the assignment carefully to make sure that you have modelled every aspect of this game.

Make sure not to write useless constraints. For example, if f is a total functions, the constraints $\forall x : \exists y : f(x) = y$ and $\forall x : \#\{y : f(x) = y\} \leq 1$ are useless. They follow from the fact that f is a total function. Adding useless constraints to your specification often makes it less readable, and less readable means less points.

The IDP IDE supports autocompletion (CTRL+SPACE), use this to save yourself some time.

3.1 Verifications

Please, verify the following claims and answer the questions (Use the 7 verification procedures in the skeleton for this):

1. Check whether the investigator can reach all locations in Arkham, i.e. there is a path from each location to every other location.
2. For the structure `Impossible` check that it is impossible to win.
3. For the structure `Unlosable` check that it is impossible to lose.
4. How long can you avoid losing in the `BoxedIn` structure, assuming the monsters move optimally for you?
5. Check that for the structures `Easy` and `Harder` it is possible to win and possible to lose.
6. Verify that once the game is won or lost, it stays that way.
7. Verify that it is impossible to win if the investigator does not attack.

If any of the claims are untrue explain why this is the case. The structures are provided in the `Structures.idp` file.

3.2 Visualisation

A procedure to visualise the game is provided (`show(struc)`). You can use this to check if your model of the game behaves as you would expect. Additional information needed for Visualisation added to the structures, you can ignore this information.

If you would like to add your own structures, remember to add the visualisation information. You can do this by simply copying it from the given structures.

4 Practical

The output of this part of the project consists of

1. A *concise* report in which you discuss design decisions.
   - For each predicate symbol `p` and function symbol `f` you introduce, the report should contain a detailed description of its intended interpretation in the following format (which we also used above):
     - `p(x,y)` is true if and only if ⟨some condition on x and y here⟩,
     - `f(x,y,z)` is ⟨some description here⟩.
   - The report should also contain the time you spent on this part of the project. The expected time needed is 10 to 15 hours.
2. A file "solution.idp", containing your IDP specification.
3. About the code:

- Your system should give the expected answer to all tests we provide (without changing the instances). If this is not the case, you should clearly argue why your specification fails. We will perform extra verifications, so it does not suffice that your theory captures all provided examples.
- You are obliged to start from our skeletons, and you are **not allowed** to change the given symbols or their interpretation in the structures. Remember that you **are allowed** to add new symbols to the vocabulary.
- Use well-chosen names for introduced symbols and variables.
- Make sure there are no warnings, except for the *Verifying and/or autocompleting structure* warnings!
- Choose your ontology wisely. An important part of our quotation consists of checking whether your theory is a good and readable representation of the domain knowledge.
- Write comments! Clearly specify the meaning of every line in your theories!
- Use good indentation.

4. Grading of this part of the project is divided into three parts:
   - 1/2 is based on whether your solution satisfies our test cases. You lose 1 point per failed test case, there are less points available than there are test cases, e.g. if you pass half of the test cases, you will receive less than half of the points.
   - 1/4 is based on the readability of your code. This includes the existence of any warnings, code duplication, lack of comments, very long lines, complex sentences, too many quantifications, etc.
   - 1/4 is based on the checks you were asked to implement in Section 3.1. **HINT**: There are several subtleties in the verifications, take care.

You are allowed to discuss this project with other people, but are not allowed to copy any code from other groups. This is not an open source project, i.e., you are also not allowed to put your code openly available on the web. If you want to use git, do not use public GitHub repositories (we will find them).

For any questions, do not hesitate to contact one of the assistants.

This project is made in teams of two students. You submit the result on Toledo before the 22th of november 2015, 23.59. **You only submit this once per group.**

Good luck!