MCS Project Part 2: Saving Arkham again

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

December 8, 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

- In this second part, you must model the 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, some of the assumptions and rules have changed to simplify the assignment. Section 3 explains what you have to do for this 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

The game, for this part, consists of a number of key concepts:

- The town
- The investigator
- Monsters
- Gates
- Game turns

These will be 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.

A context is provided which models this, so you do not have to worry about the town being modeled correctly.

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 over it stays that way. The game is won if there are at least the specified amount of gates closed at a certain time. The game is lost if there are at least the specified amount of gates open at a certain time.

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, i.e. he can only act once per 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. 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. When there are open gates, a single monster can come through that gate each monster turn, however gates do **NOT** have to spawn monsters each turn, it should simply be possible (if there are enough monsters in the monster pool).

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.

In the previous assignment monsters were only counted, i.e. they did not have an identifier. In this part we identify each monster by a specific number. Additionally the restriction that at most three monsters can be present in a single location no longer applies.

2.5 Gates

Every monster turn a single gate *can* open in a **building**, where no gate is currently open. A gate does not have to open each turn. 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. There is never a closed and open gate in the same building. I.e. gates are either open, closed or none existent.

The investigator can only close a gate if there is an open gate in his location.

2.6 Clues

Clues are no longer present in the game, so you do not have to model them. Specifically this means:

- Attacking simply makes the monsters in your location disappear.
- Closing an open gate is possible whenever the investigator is in the same location.

3 Part 2: Modelling the game in Event-B

You are **obliged** to model this assignment in Event-B! In order to model this, you complete the skeleton present in the archive file on toledo. This second part is split up into four smaller parts.

- Complete the `Saving_Arkham_0` machine
- Complete the `Saving_Arkham_1` machine
- Perform LTL/CTL checking on the `Saving_Arkham_1` machine using ProB.
- Do a data refinement on the `Saving_Arkham_1` machine.

The skeleton is offered as a zip archive. To import it into Rodin, you need to create a new project and import the archive into that project.

For the first and second part you **cannot** add to or change the variables and contexts, or change or remove invariants, or remove or rename events.

3.1 Complete the `Saving_Arkham_0` machine

For the first part of this assignment you have to complete the `Saving_Arkham_0` machine. To do this you have to add the proper guards and actions to the events of the machine.

The machine consists of three variables:

game_over is a boolean variable which is true if and only if the game is over.

open_gates is a set of buildings. A building is in this set if and only if there is an open gate in that building.

closed_gates is a set of buildings. A building is in this set if and only if there is a closed gate in the building.

DO NOT CHANGE OR ADD ANYTHING TO THESE VARIABLES.

This machine consists of the following four events, apart from the INITIALISATION event:

end_game happens when the game comes to an end, i.e. the game is in play and either the win or lose condition applies. After this event the game must be over.

game_over Is only enabled when the game is over. It indicates the game is over and makes sure the machine can run forever.

open_gate Opens a gate in the given location. This can only happen during the game and in a building where there is no **open** gate. After this event there is an open gate in the building given as the parameter.

close_gate closes the gate in the building passed as parameter. This can only happen during the game and if there is an open gate in the specified building.

Initially the game the game is not over and there are no open or closed gates.

DO NOT ADD ANY EVENTS, only add guards and actions to the supplied events, and potential invariants.

3.2 Complete the `Saving_Arkham_1` machine

For the second part of this assignment you have to complete the `Saving_Arkham_1` machine. To do this you have to, again, add the proper guards and actions to the events of the machine. This machine is a refinement of the `Saving_Arkham_0` machine, so you must make sure you uphold the correct refinement relation between the two machines.

The machine consists of eight variables:

location is the location of the investigator, it is an element of the locations set, defined in the `Arkham` context, additionally this context also specifies the connected location relation.

state is the state the game is in, this variable can have any of the values specified in the states set, defined in the `GameStates` context. Two subsets of this states set are defined, `over`={won, lost} and `playing`={player_turn, monster_turn}. The open_to_lose and closed_to_win constants are also defined in this context.

turn_end is a boolean variable, which is true if and only if it is the end of the current turn.

monster_locations is a function mapping monsters to locations. It maps each monster currently in Arkham to its location.

monsters is the set of monsters currently in Arkham, it is a finite subset of the monsterpool, which is defined in the `Monsters` context.

acted is the boolean variable, which is true if and only if an action happened this turn, i.e. a gate was opened or closed, or the investigator attacked. Only one act can happen during each turn, player or monster turn.

open_gates is a set of buildings. A building is in this set if and only if there is an open gate in that building.

closed_gates is a set of buildings. A building is in this set if and only if there is a closed gate in the building. The result is that the gate is closed.

DO NOT CHANGE OR ADD ANYTHING TO THESE VARIABLES, only add guards and actions to the supplied events, and potential invariants.

This machine consists of the following 10 events, apart from the INITIALISATION event:

move happens when the investigator moves. The investigators location is changed to the new location, this marks the end of a player turn. Moving can only happen to a connected location.

monster_move happens when the monsters move. This marks the end of a monster turn. The parameter is the new function mapping monsters to their locations. The movement should happen in such a way that each monster moves to a connected location and they never move to a building.

add_monster happens when a monster exits an open gate in the specified location. The parameter monster is the identifier of the new monster, this must be an element of the monster pool and different from all other monsters. The location must be a building where there is an open gate. A monster can only be added to a location where there is not already a monster.

attack happens when the investigator attacks. This can only happen once during a player turn and before the investigator moves.

next_turn happens when a playing turn, i.e. player or monster turn, ends and the win and lose conditions are not satisfied. It marks the start of the next turn, you can use the next_turn function for this.

lose_game happens when the lose condition is satisfied, too many gates open, at the end of a playing turn. The game is lost after this event.

win_game happens when the win condition is satisfied, enough gates closed, at the end of a playing turn. The game is won after this event.

game_over Is only enabled when the game is over. It indicates the game is over and makes sure the machine can run forever.

open_gate Opens a gate in the given location. This can only happen during a monster turn and in a building where there is no **open** gate. After this event there is an open gate in the building given as the parameter. This counts as an action during a monster turn.

close_gate closes the gate in the building passed as parameter. This can only happen during a player turn and if there is an open gate in the specified building. The result is that the gate is closed.

Initially the game the game is in the start of a player turn, there are no open or closed gates, and there are no monsters.

3.3 Perform LTL/CTL checking on the `Saving_Arkham_1` machine using ProB.

To do LTL/CTL checking on the `Saving_Arkham_1` machine follow the following steps:

1. In the `SEES` section of the machine change the `Monsters` context to the `Monsters_For_LTL_CTL` context. This context reduces the size of the monsterpool to allow for more efficient checking.
2. Right-click the `Saving_Arkham_1` machine and under the `Prob Classic...` sub-menu, select `Open in Prob classic`. A prompt will appear saying the file exists and ask whether your want to overwrite it. Select Yes.
    - IF, you get a prompt saying ProB can not be found, go to the windows/preferences menu and under ProB/Prob Classic, select the path to your ProB installation. This is not the ProB plugin, but the actual application, as used in the exercise session on ProB.
3. Your machine will open in ProB in read-only mode, allowing you to specify and check LTL and CTL formulas as seen in the exercise sessions on ProB.
4. Use LTL and/or CTL to specify the following statements.

LTL/CTL

Check the following statements on your machine using LTL and/or CTL statements. **Add these statements to your report so we can verify them!** Remember that B syntax can be used in {curly brackets} in an LTL and CTL formula. The occurrence of an event is expressed as [E] for event E.

- When it is a player turn it is always possible to eventually win.
- When the game is lost it stays lost forever.

- It is possible that the game goes on forever, i.e. it never reaches the won or lost state.
- The game stays in the playing state (player or monster turn) until either the win or lose condition, i.e. too many gates open or closed, holds.
- The win and lose condition, i.e. too many gates, cannot hold at the same time.
- The game is always in exactly one of the playing states until at least the end of a turn.
- It is always true that if you eventually win, you eventually have more closed gates than open gates.
- If there is a monster in the game, it stays in the game until the investigator attacks. Remember that for the purpose of verification, there is only one monster in the monsterpool.
- To win you have to eventually close two gates.

3.4 Do a data refinement on the `Saving_Arkham_1` machine

In the last part of this assignment you are asked to refine the `Saving_Arkham_1` machine. Specifically you are asked to do a data refinement. This part is a challenge and is more complex than the three previous parts. It counts towards less points than the other parts, so do not invest too much time in completing this part.

The challenge for this part is as follows. Refine the `Saving_Arkham_1` machine so that the `turn_end` and `acted` are replaced with a single variable `phase`. I.e., the phase variable should describe which phase of a turn the game is in. When inspecting the game it is clear that during a player turn there are actually three phases: the act phase when the investigator can close a gate or attack, the movement phase and the end of the turn. Similarly there are four phases in a monster turn: the act phase, monster add phase, movement phase and end of turn. Note that in some phases something must occur, e.g. movement phase, while others may be skipped, e.g. act phase. Take this into account.

While doing the data refinement try to keep the behaviour of the game the same as in the `Saving_Arkham_1` machine.

For this part you are expected to:

1. Refine the `Saving_Arkham_1` machine
2. Create a context for the phases
3. Replace the variables `turn_end` and `acted` by `phase`, and add the needed linking invariants
4. Change and/or add events in accordance to these changes.

4 Practical

The output of this part of the project consists of

1. A *concise* report in which you discuss design decisions.
   - The report should contain the time you spent on this part of the project. The expected time needed is 10 hours, this depends on your preparation during the exercise sessions. The first three parts should be completable by everyone within those 10 hours. The last part is an extra challenge, take this into account when planning your assignments.
2. A zip-file "solution.zip" containing all files in your Rodin project. Create this using Rodin's export option, which works the same as in Eclipse. Please make sure we can import your project from that .zip before submitting.
3. About the code:
   - You are obliged to start from our skeletons, and you are **not allowed** to change the given symbols or their interpretation. Remember that you **are allowed** to add new invariants to the intermediate machines, and new variables to the last refined machine.
   - Use well-chosen names for introduced symbols and variables, invariants, guards, and actions.
4. Grading of this part of the project is divided into three parts:
   - Just under half of the points are based on the first two parts of the assignment, i.e. the completion of the first two machines.
   - Just under half of the points are based on the third part, i.e. the LTL and CTL verifications.
   - The rest is based on the last part, i.e. the extra data refinement.

4.1 Exporting your project

To export your project from Rodin, follow the following steps:

1. Right-click your project and select `Export...`
2. Select `Archive File`
3. Deselect your project folder, on the left (otherwise an additional folder is included in the zip, making it difficult to import)
4. Select **all** the files in your project, on the right. This makes sure only the files are present in the zip file.
5. Press finish
6. Create a new project and make sure you can import the archive file into it.

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 the same team as the last assignment. This project is expected to be completable in 10 hours or less, thus you should be able to complete it before the end of December. To give you more flexibility in planning your different assignments you are only expected to submit the result on Toledo before the 10th of January 2016, 23.59. This does not mean we expect you to work on the project during the "blok" or during the holidays, we want to give everyone the chance to complete the project without it impeding their chance to complete other projects or study for the exams. **You only submit this project once per group.**

Good luck!