Analysis of RPG models with UPPAAL.

Stefaan Kenis

Department of cumputer science
Universiteit Antwerpen
stefaan.kenis@student.uantwerpen.be

Abstract

UPPAAL is a tool for modelling, simulation and verification of Real-Time systems that is developed at the universities of Aalborg and Uppsala. This tool is mostly used for systems that require communication and timing aspects and now we will analyse role-playing game models with this tool. The creation of the game is done in UPPAAL itself and we also export models from AToMPM to metaDepth. Then these models are transformed into valid xml files that can be used in UPPAAL.
The current version of UPPAAL is available on `http://www.uppaal.org`.

Keywords: UPPAAL, role-playing game, deadlock, reachability, modelling, simulation, verification, AToMPM, metaDepth

1. Introduction

In the previous report we described what UPPAAL is and how can use it to do analysis. In this paper we will use UPPAAL to analyse role-playing game models. In section 2 we explain the RPG models that we want to analyse. In section 3 we explain how we created RPG models in UPPAAL. In section 4 we explain how we exported RPG models from AToMPM, transform them to a valid xml with EGL and imported them into UPPAAL. In section 5 we explain the analysis that is done on these models. In section 6 we give a conclusion about analysing RPG models with UPPAAL.

2. Role-playing game models

The game that we want to model consists of one or multiple scenes. Each scene consists of tiles of which adjacent tiles are connected in both directions. There should be exactly one hero in the game. There also is another character: the villain. Hero and a villain can not stand on the same tile and they can attack each other when they are on adjacent tiles. Other items in the game are goals and traps. A goal is an item that the hero wants to pick up and a trap does damage when the hero steps on it. The game finishes when the hero has collected all goals or when the hero has died. To make the models easier to make and understand we limit the number of villains to one. The reason for this will be explained in the next section. There can be multiple goals and traps in the game. And we only look at one scene, because this does not change the analysis much. In the original RPG models it was also possible to have an obstacle. If we do not draw the tiles with obstacles, then we do not have to worry about that either. The goal was to make the models easy to make and understand, without changing the analysis too much.

3. Creation of models in UPPAAL

In UPPAAL we have the possibility for creating elements that are combined to a system. These elements can communicate to each other by using channels. The elements that our game needed are Hero, Villain and Turn. We explain each of them separately.

3.1. Hero

The most important element in our game is the hero. The hero should be able to walk to all reachable tiles. So we started by creating locations for every tile and connecting them with edges in both directions between two adjacent tiles. For each location we needed a boolean that showed if the hero is standing on that tile or not. So always exactly one of these booleans should be true. We also had to define the initial location of the hero. Then we needed a variable that represented the health of the hero. Initially it is set to 100. Now for every edge we had to include guards, synchronisation and updates. The guards are the conditions for when the hero is allowed to take that edge. In our game the hero was not allowed to walk to a tile where a villain is standing. So we also needed data variables for where the villain is

standing. If the tile we want to go to is empty the hero can walk there. When the hero walks it will send a message: "HeroTurn!". The update section will update the data variables about where the hero is standing. When the hero wants to walk to a trap the guard also includes information about whether he has already stepped on the trap or not. If it is the first time, then the hero loses 10 health and the boolean for the trap is set to true. This way the next time that the hero moves to this tile, it is the same as walking on an empty tile. For a goal we do something similar, we set the boolean for that goal to true.
Then we still need to be able to fight the villain. This is done by creating self-edges. The guard makes sure that the villain stands on an adjacent tile. We also send the message "HeroTurn!" and decrease the health of the villain with 10. An example of the hero looks like this:

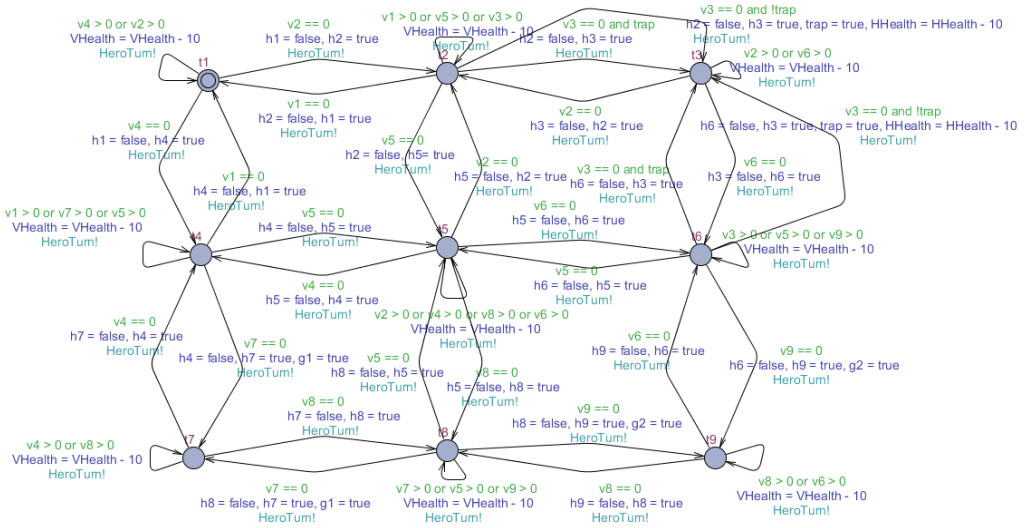

Figure 1: Hero (9 tiles, 1 trap, 2 goals)

3.2. Villain

The part of the villain is similar to that of the hero. The villain needs the same structure of locations and edges. It has to check that he does not walk to a tile where the hero is standing. It sends the message "VillainTurn!" and updates the variables that show where he is standing. The villain can fight the hero if he is on an adjacent tile. The villain does not need to know anything about goals and traps. The part of the villain looks like this:

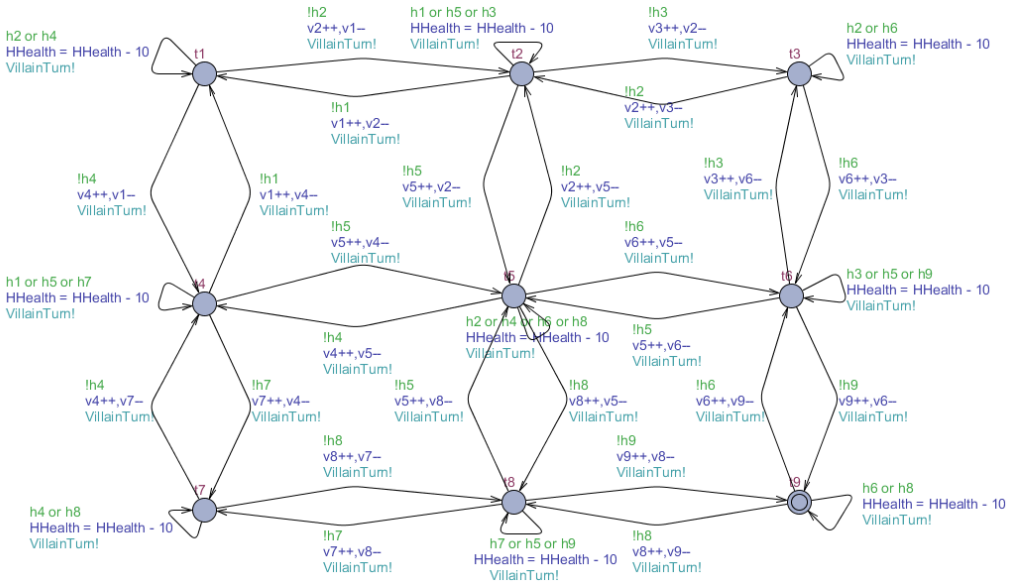

Figure 2: Villain

3.3. Turn

The last part is Turn. This keeps track of whose turn it is. It receives the messages from both hero and villain and makes sure the right character can do an action. If both goals are picked up, their booleans are true, then the game is finished. This results in a deadlock state called "Won". If the hero has no more health the game also stops in the state called "Died". It is also possible that the villain is dead. Then the hero can keep moving until he dies or has picked up all goals. The turn part looks like this:

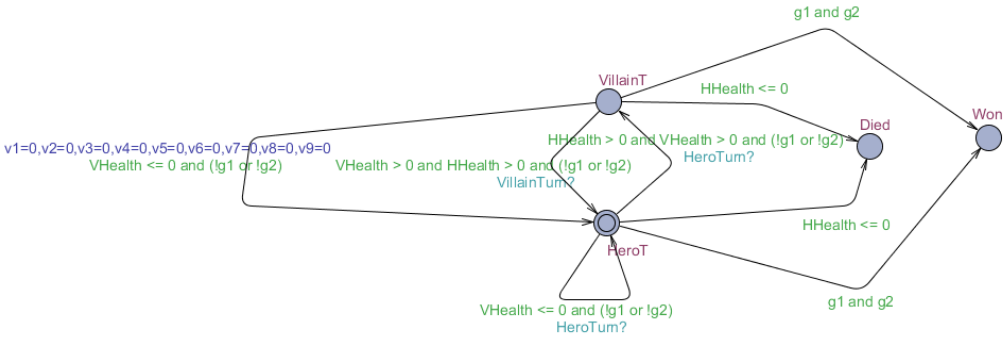

Figure 3: Turn

4. Export models from AToMPM

We could already export models from AToMPM to metaDepth. These metaDepth models can be transformed in a valid xml file that can be used in UPPAAL. This transformation is done with EGL (Epsilon Generation Language). We have restricted our model again to have one villain. We also did not include fighting and stepping on traps and goals. The reason for this is because it is more complicated to adapt the right variable. We don't know what goal we are on, we only know we step on a goal. Also fighting is not easy, because we need to know all outgoing links for one tile and check if the other character is on one of them. So both hero and villain can walk and they can not both stand on the same tile. To complete this kind of model to a model with traps, goals and fighting is easy in UPPAAL. If we import such a model in UPPAAL it looks like this (Note that all labels are on the same place, because an association in AToMPM does not have a position attribute):

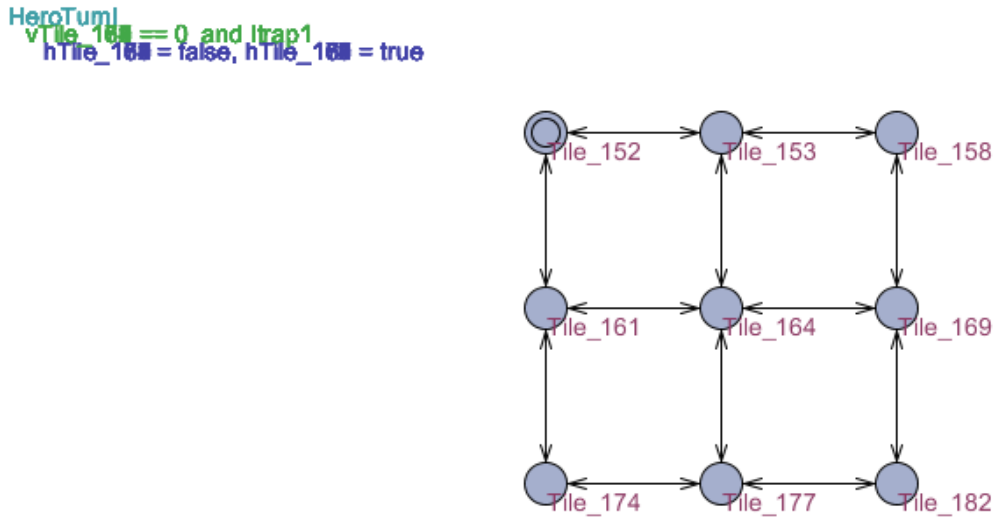

Figure 4: Hero (Exported model from AToMPM to UPPAAL)

5. Analysis

The analysis was mostly done on the models created in UPPAAL itself, because they are an extension of the models that were made by exporting AToMPM models. UPPAAL had both a simulator and a verifier to analyse models. The simulator is used more in the implementing phase, because this way we can see very fast if the game does what it should do. But if we want analysis for all possible executions we need to use the verifier. UPPAAL is known for analysing deadlocks and reachability. This are important properties for our RPG models. We now explain the checks that we did on the model explained in section 3 one by one.

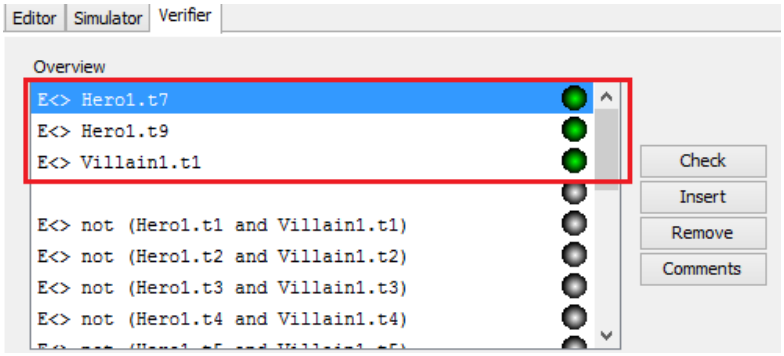

Figure 5: Reachability checks.

Maybe the most important question is that the hero can reach all goals in the game. Otherwise the hero can never win the game and maybe the game will go on forever. So for every goal we have included a check. These are the two first checks, because we have two goals in this model. It literally tells us if there exists a path from the initial state to the goal state. Then we have also included a check that the villain can reach the hero (and thus can reach all tiles that the hero can reach). Otherwise the villain is just walking in another group of tiles, which is useless and not what we want.

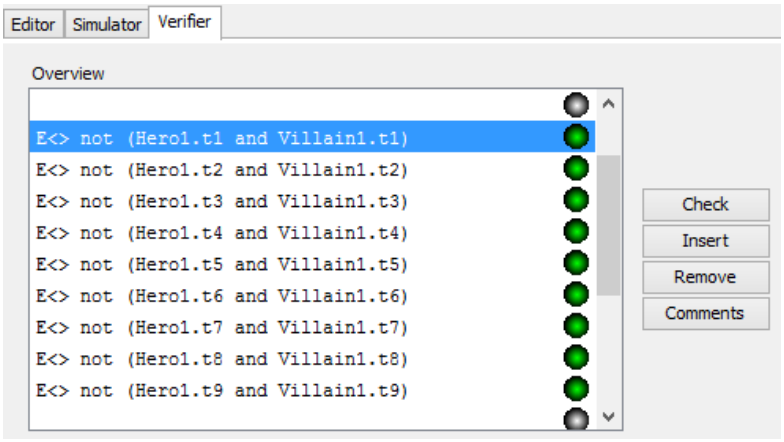

Figure 6: Check at most one character on tile.

These checks are rather trivial. For every tile we check if the hero and villain never stand on the same tile.

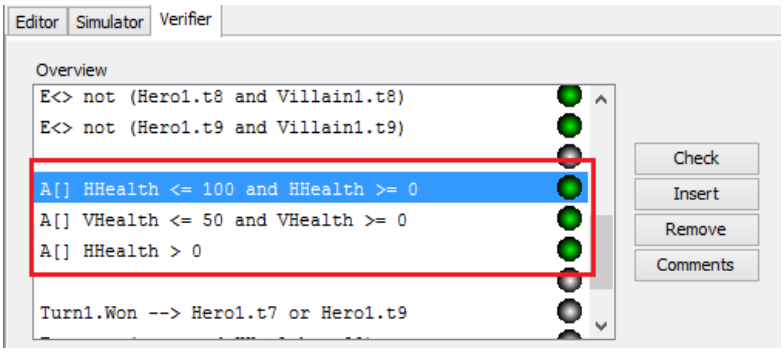

Figure 7: Check health of characters.

This kind of checks are invariants. For every state the system is in, this property should be true. The first two lines make sure the health of both characters is always in the right range. The last is maybe not that intuitive. This actually says that the hero never dies. This is true, because the hero has more health than the villain and both characters can only attack each other in turn. So the villain always dies before the hero and there are not enough traps in the game to let the hero die. If we change the initial health of the characters (hero smaller than villain), then this invariant does not hold any more.

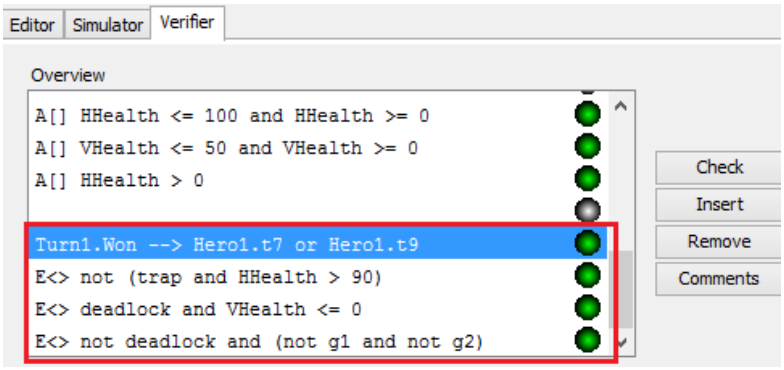

Figure 8: General checks.

The first one shows that the hero must stand on a goal tile when the game is finished. It has just picked up the last goal. The second check is that the hero must have lost health, when he has stepped on a trap. His health can not be larger than 90 any more. The third line shows that the game can still finish if the villain has died. Of course this check is not enough, because we want that our game only finishes when all goals are picked up. This is checked in the last line. It says that the game will not deadlock if at least one of the goals is not picked up yet. Remember that the hero can not die in this specific model, otherwise the game should also end when he dies.
All checks that we have discussed passed. For other models we can do the same kind of checks again.

6. Conclusion

UPPAAL is a good tool for creating and analysing RPG models. The tool is easy in use and very efficient in analysing. The communication channels between different parts of the system are useful for alternating turns, which we need in role-playing games. UPPAAL can analyse important properties for RPG. It is important to analyse reachability to know which states of the game are reachable. It is also very useful to know the deadlock states of the game. They tell us when the game can stop. In an RPG this is when the hero has picked up all goals or dies.