UppaalTD: a Formal Tower Defense Game

Formal Methods for Concurrent and Real-Time Systems

A.Y. 2024 - 2025

Andrea Manini and Pierluigi San Pietro

Politecnico di Milano
firstname.lastname@polimi.it

1 Introduction

A *Tower Defense Game* is a strategy game in which the player aims to protect valuable assets from an enemy invasion. In this project, we refer to the main asset to protect as the *Main Tower*. The player has *turrets* at its disposal, which can be placed on a Map in order to stop the invasion. An invasion is typically referred to as a *wave*. To win the game, the player must place turrets to ensure the Main Tower survives either to all the enemies in a wave or for a given timeout.

As a Quality Assurance Engineer in an AAA game development studio, your task is to formally verify the correctness of the specification for a new Tower Defense Game under development. You are required to provide a model of the game as a Network of Timed Automata using Uppaal [1,3] and to verify some relevant formal properties of your model.

This paper is structured as follows: Section 2 describes the game and all the necessary entities that must be considered in your model. Section 3 describes the analysis to be performed on a simplified version of the game. Section 4 describes the analysis to be performed on a stochastic version of the game. Section 5 provides the necessary information for the final delivery of the project. Section 6 contains other details that must be considered while developing the project.

2 Game Description

This section describes the main components of the Tower Defense Game you must take into consideration while developing your Uppaal model.

Map. A Map on which enemies move, turrets are placed, and the Main Tower to be defended is located. The Map, shown in Figure 1, is a grid of 16×8 cells. In particular, the blue cell (*Main*) is the the Main Tower location. Green cells (T) indicate the positions over which turrets can be placed. Red cells (E) constitute the path over which enemies move to reach the Main Tower.

Main Tower. The tower to be defended, characterized by its life points. If the life points of the Main Tower drop to 0 or below, the player loses.

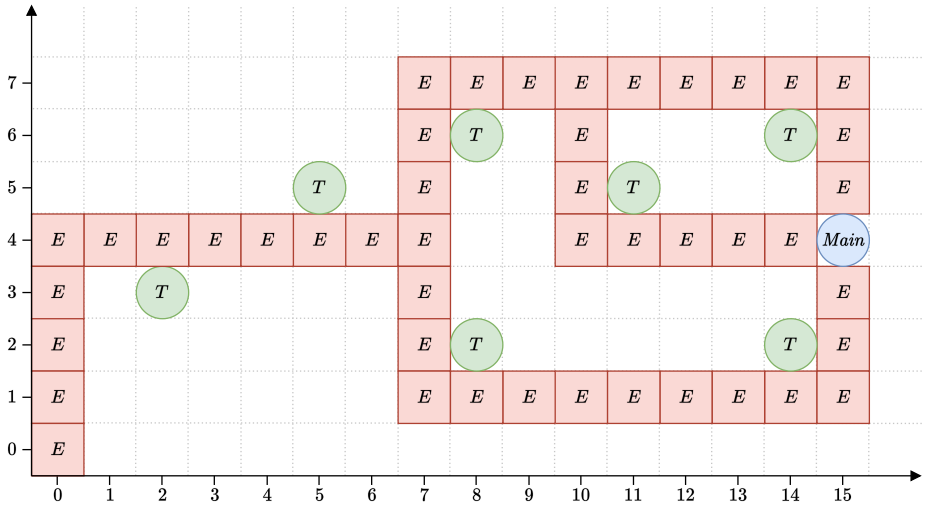

Fig. 1: The Map over which the game is played.

Enemies. Enemies are classified into two distinct types: Circles and Squares. They are characterized by several parameters: speed, health, and damage.

- **Speed**: describes how fast an enemy moves on the Map. In particular, for an enemy to move, a delay exactly corresponding to its speed must first elapse;
- **Health**: represents the maximum life of an enemy;
- **Damage**: denotes the attack power of an enemy.

Enemies can move only on red cells. They cannot deviate from their predefined path and can only move one cell at a time, between adjacent cells. Additionally, enemies cannot move backward; they always proceed in a single direction towards the Main Tower. When a branching point is reached on the Map, an enemy non-deterministically chooses which path to follow. The possible directions that enemies can follow are depicted in Figure 2. Multiple enemies can simultaneously occupy the same cell. If an enemy's health drops to 0 or below, the enemy dies (after being killed it cannot be targeted by turrets anymore). Upon reaching the Main Tower, i.e., the enemy's position coincides with the Main Tower position, the enemy attacks exactly once. After attacking the Main Tower, if not killed beforehand, the enemy leaves the Map after a delay equal to its speed (after leaving the Map it cannot be targeted by turrets anymore).

Turrets. Turrets are classified into three distinct types: Basic, Cannon, and Sniper. Turrets are characterized by several parameters: shooting range, firing speed, and damage inflicted to enemies.

- **Shooting range**: only enemies in this range can be targeted by a turret. The shooting range r of a turret can be imagined as a square, centered in the turret position, and having a side length equal to $1 + 2r$. For instance,

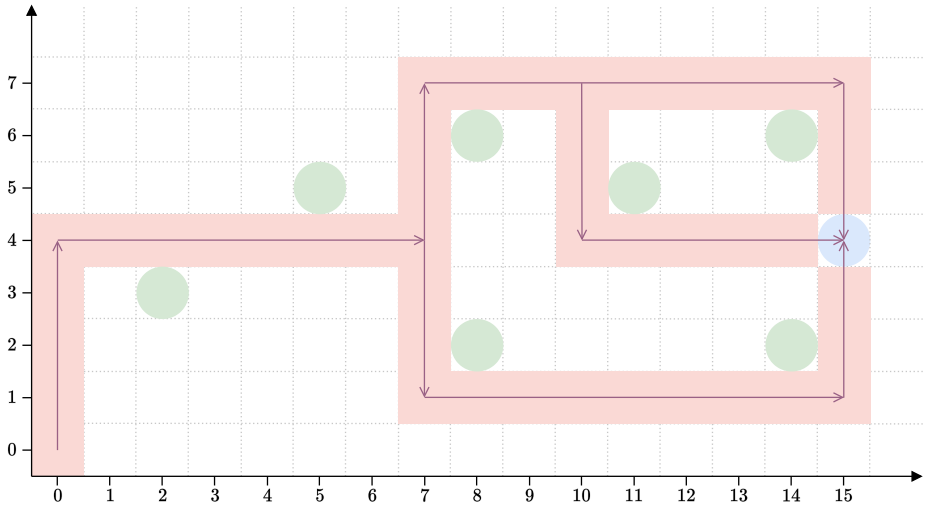

Fig. 2: The possible enemies directions, indicated with arrows. Bifurcation points are located in positions $(7, 4)$ and $(10, 7)$.

if a turret is located in position $(2, 3)$ and has a shooting range $r = 1$, the cells that fall under its range are the ones located in positions $(1, 2)$, $(2, 2)$, $(3, 2)$, $(1, 3)$, $(2, 3)$, $(3, 3)$, $(1, 4)$, $(2, 4)$, and $(3, 4)$;

- **Firing speed**: the frequency at which a turret fires. After a turret fires, it must wait a delay exactly equal to its firing speed before it can fire again;
- **Inflicted damage**: the attack power of a turret towards enemies.

Turrets can be placed only on green cells. At most one turret can be placed on a single green cell. Turrets can target only one enemy at a time, within their shooting range. Turrets target the enemy closest to them. If multiple enemies are at the same (minimum) distance from a turret, the enemy that has been present on the Map for the shortest time is targeted. If both Circles and Squares can be targeted, turrets prioritize attacking Squares.

Game Execution. At the start of the game, only the Main Tower is placed on the Map. The player's objective is to place turrets before an enemy wave begins. There is no limit to the number of turrets that can be placed, provided there are still available green cells for placement. The arrangement of turrets before a wave starts is referred to as a *configuration* of the game. There are no restrictions on the types of turrets that can be used. For example, a configuration consisting of seven Basic turrets is perfectly valid. When the wave begins, enemies spawn on the Map at regular time intervals (e.g., Circles spawn every x time units, while Squares every y time units, see the next section for values of x and y). Enemies always spawn in the cell located at the *start of the Map*, i.e., in position $(0, 0)$. DO NOT USE THE *SPAWN* KEYWORD TO MODEL THIS BEHAVIOR, AS IT IS BROKEN IN THE CURRENT VERSION OF UPPAAL! Upon appearing,

Parameter	Value
Enemies Spawning Time	
Circle Spawning Time	2
Square Spawning Time	3
Parameters of Different Enemy Types	
Circle Speed	1
Circle Health	10
Circle Damage	2
Square Speed	3
Square Health	20
Square Damage	4
Parameters of Different Turret Types	
Basic Shooting Range	2
Basic Fire Speed	2
Basic Inflicted Damage	2
Cannon Shooting Range	1
Cannon Fire Speed	7
Cannon Inflicted Damage	5
Sniper Shooting Range	4
Sniper Fire Speed	20
Sniper Inflicted Damage	8
Parameters of the Main Tower	
Main Tower Life Points	10

Table 1: Game Parameters.

they start moving after a delay equal to their speed. After an enemy moves, it has to wait a delay exactly equal to its speed before it can move again (it moves as soon as this delay elapses). Recall that enemies can be targeted only when they are alive and on the Map (they can be targeted as soon as they spawn on the Map). Turrets shoot instantly as soon as an enemy enters in their shooting range (if not waiting for the delay equal to their firing speed to elapse). The wave ends when either all enemies have been killed, the Main Tower is defeated, or a timeout expires. The following sections provide further details on the game parameters and ending criteria, depending on the considered version of the game.

3 Vanilla Version

In this version, we consider a simplified game in which only a single wave takes place, consisting of 6 enemies: 3 Circles and 3 Squares. In this version, the speed of enemies and the firing speed of turrets are constant, each depending on the specific enemy or turret type, respectively. The parameters to be used in this

version of the game are contained in Table 1. You are required to formally model the game without any stochastic Uppaal features. The game must be modeled as a Network of Timed Automata through the Uppaal GUI. You decide what to model as a Timed Automata feature and what as a variable.

Properties to verify. You are required to formally express in TCTL logic and verify the following properties of your model, by creating suitable queries in the Uppaal verifier. In the following, the length of a path between two cells a and b is defined as the total number of moves required for an enemy to travel from a to b. For example, the length of the path $(0,0) \rightarrow (0,1) \rightarrow (0,2) \rightarrow (0,3)$ is 3.

Properties to be verified WITHOUT TURRETS:

I) The game never reaches a deadlock state.
II) All enemies can reach the Main Tower.
III) Circles reach the Main Tower in no more than $n \cdot c$ time units, where n is the length of the longest path from the start of the Map to the Main Tower, and c is the speed of the Circles.
IV) Squares reach the Main Tower in no more than $n \cdot s$ time units, where n is the length of the longest path from the start of the Map to the Main Tower, and s is the speed of the Squares.
V) All enemies never leave the red path.

Properties to be verified WITH TURRETS:

VI) Verify whether the following configuration is winning or losing:
- Basic in cell $(5,5)$;
- Cannon in cell $(8,2)$;
- Cannon in cell $(8,6)$;
- Cannon in cell $(14,2)$;
- Cannon in cell $(14,6)$;
- Sniper in cell $(2,3)$;
- Sniper in cell $(11,5)$.

VII) The game, under the configuration described in point (VI), never reaches a deadlock state.
VIII) Provide at least two other significant configurations using at least 4 turrets and verify whether they are winning or losing.

4 Stochastic Version

In this version, a wave has a total of 700 enemies: 400 Circles and 300 Squares. You are required to extend your previous model with stochastic features. In particular, the firing speed of turrets is now modeled as an exponential probability distribution having a rate equal to $1/\mathit{fireSpeed}$ (where *fireSpeed* is the original firing speed of a turret). The speed of enemies is also modeled as an exponential probability distribution having a rate equal to $\mathit{speed}/10$ (where *speed* is the

original speed of an enemy). Furthermore, the Main Tower now has at least 100 life points. All other parameters are the same as the ones reported in Table 1.

Exponential Probability Distribution. The following is an extract of the Uppaal documentation regarding exponential probability distributions [4].

The rate of exponential is a ratio expression which specifies the rate of an exponential probability distribution. The rate expression can be a simple integer expression or two integer expressions separated by a colon like `r:q`, where the rate is determined as the ratio r/q.

The rate of exponential is used in statistical model checking. If a location does not have an invariant over time, then it is assumed that the probability of leaving the location is distributed according to the following exponential distribution: $Pr(\textit{leaving after } t) = 1 - e^{-\lambda t}$, where $e = 2.718281828\ldots$, t represents time, and λ is the fixed rate. The probability density of the exponential distribution is $\lambda e - \lambda t$, where λ is the probability density of leaving at time zero, i.e., as soon as some edge is enabled. The smaller the rate, the longer the delay is preferred.

The generation of exact delay relies on pseudo random number generator and on 32-bit architectures the longest possible delay is: $\ln(231)/\lambda \sim 21.49/\lambda$.

Properties to verify. You are required to extend the model you developed for the Vanilla version of the game and to formally express in TCTL logic using Uppaal's stochastic operators [2] and verify the following properties of your model, by creating suitable queries in the Uppaal Verifier.

Properties to be verified WITH TURRETS:

I) Simulate the system for a maximum timeout of 200 time units and estimate how long the Main Tower remains undamaged in the configuration provided in the query (VI) in the previous section.

II) Determine the probability that, for a maximum timeout of 200 time units, the Main Tower survives in the configuration provided in the query (VI) in the previous section.

III) Simulate the system for at least two other significant configurations and provide an analysis of other interesting behavioral aspects of the game.

5 Delivery Instructions

It is mandatory to deliver:

- A PDF report (max 10 pages excluding front cover and bibliography, no constraint on the template) describing:
  1. The model (emphasis on critical modeling choices you have made);
  2. The game configurations you have chosen;
  3. Verification results.

- An XML file containing the Uppaal model of the Vanilla version of the game and an XML file containing the Uppaal model of the Stochastic version of the game. Make sure:
  1. Each file is the same as what you present in the report;
  2. Each file is test-ready (i.e., it includes the queries you ran for the experiments and game configurations are easily selectable).

Beware: I will re-run the queries you describe in your report, so make sure the model is experiment-ready and queries are saved in the verifier section!

Delivery deadline: also for students participating in the exchange program with UIC, is July 24 (June 30 if you want the grade registered in the first call). RECALL THAT YOU MUST BE ENROLLED TO A VALID CALL TO GET YOUR GRADE REGISTERED!

To submit your work, send an email to `andrea.manini@polimi.it` with the report, Uppaal files, and potential supplementary material to be evaluated. Students under the UIC program must explicitly specify it in the email.

6 Minor Things

- ALL parameters and variables must not be hardcoded in templates ad declarations, but should be generalized as much as possible (this does not apply to queries written in the verifier). If their names are not sufficiently self-explanatory, they must be accompanied by meaningful comments.
- ALL functions created in the declarations must have meaningful comments (à la Javadoc), i.e., they must describe input parameters, output results, and give a brief overview of how the function works.
- Templates must not be (graphically) confusing.
- Each template must contain at least one significant comment briefly explaining its functioning.
- ALL queries must have meaningful comments.
- Files must be structured to facilitate the execution of queries (e.g., by including comments containing game configurations).

Grading criteria. Up to 2 bonus points will be awarded for projects with clean, well-documented code and easily readable, non-confusing templates.

- Maximum grade without stochastic features: 26 (with the bonus points the grade can increase up to 28).
- Maximum grade with stochastic features: 30L.

A sufficient grade (up to 18 without considering bonus points and up to 20 considering bonus points) is still obtainable by developing an excellent model of the system but with no queries at all.

If some queries cannot be verified within a reasonable amount of time on your machine (say, less than two hours) due to the complexity of your model, it is recommended for you to simplify the model. If the model is still too complex

even after simplifications, queries that take too long to be formally verified will still be evaluated based on the correctness of their specification. Whether to assess full points for those queries will be considered for each specific case. For this reason, if you encounter this issue, please document it in your final report. In this case, your report must also include details on the machine specifications (processor and RAM) on which the queries were run and the maximum time limit reached before terminating the verification process.

References

1. Behrmann, G., David, A., Larsen, K.G.: A Tutorial on Uppaal, pp. 200–236. Springer Berlin Heidelberg, Berlin, Heidelberg (2004), https://doi.org/10.1007/978-3-540-30080-9_7
2. David, A., Larsen, K.G., Legay, A., Mikuăionis, M., Poulsen, D.B.: Uppaal smc tutorial. Int. J. Softw. Tools Technol. Transf. **17**(4), 397–415 (Aug 2015). https://doi.org/10.1007/s10009-014-0361-y, https://doi.org/10.1007/s10009-014-0361-y
3. Larsen, K.G., Pettersson, P., Yi, W.: Uppaal (2025), https://uppaal.org/, accessed: 2025-03-28
4. Larsen, K.G., Pettersson, P., Yi, W.: Uppaal documentation (2025), https://docs.uppaal.org, accessed: 2025-04-02