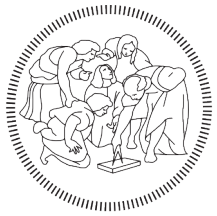

POLITECNICO
MILANO 1863

# FORMAL ANALYSIS OF SEARCH-AND-RESCUE SCENARIOS

Formal Methods for
Concurrent and Real-Time Systems

Lorenzo Bossi 10673045
Matteo Boido 10706985

July 20, 2024

# Contents

# 1 Assumptions

In this section we list all the assumptions we made on aspects not explicitly stated by the homework assignment and explain why we found them reasonable.

- Professionals cannot die and they stay on the scene after they save some civilian. We assumed that first responders are well trained and equipped, so they can stay close to the fire without being injured. Furthermore they do not leave the scene until there in no civilian left to save.
- Civilians and Professional cannot move on a fire cell, but can stay on an exit cell. Drones can stay in every type of cell.
- Civilians leave the scene only when they die or they escape. That means that while the are being helped or helping another civilian, they stay locked in the same position, occupying the cell.
- It is possible that civilians die while being helped by a first or zero responder because their help arrived too late:
  1. if the victim was being assisted by a **first responder**, the first responder is free to move again as soon as the victim die.
  2. if the victim was being assisted by a **zero responder**, the zero responder continue the helping process for the time needed. When the time elapse only the responder is safe. This is to model the fact that the zero responder tries to bring the victim to the exit thanks to the drone indications, but the victim die in the meanwhile, so only the responder manages to get to the exit alive.
- Drones can (try to) communicate with as many entities (professionals and civilians) as possible in a clock cycle even if they move only once per clock cycle. This choice was made because the drone shouldn't move only when it has already handled all the danger situations found in that spot.

# 2 Model Description

In this section we will give an high level view about how we implemented the model and the interactions between the components.
Other than the parameters specified by the homework assignment and the configuration of the grid and entities, we added the configurable parameters:

- **randomness:** knob to tune the exploration-time trade-off
- **targetFireDistance:** target distance for the move to closest fire strategy
- **droneLimitDecisionDistance:** limit distance between civilian and professional for drones to decide weather to suggest to call the closest professional, or to help directly the civilian.
- **disableDrones:** boolean variable usefull to test the scenario without the action of drones.

## 2.1 Controller

The controller template is responsible for ensuring the time flows correctly and for the synchronization of the other components.
The main section of the timed automaton is composed of a cycle where every location is urgent except the first one. The intermediate locations synchronizes with the other components, allowing them to move or take actions. As time is allowed to flow only when every component is in a non urgent location, the time won't flow until every entity has reached a non urgent state. When this happen time will flow until one time unit has passed. In this way the clock ensures that each component start and end its action within a time unit.

## 2.2 Civilian

The civilian template is responsible for modeling people involved in the fire who tries to go to the closest exit. They will eventually enter the location *safe* or *dead*.
The template can be divided in two main logical sections:

1. **not in danger** : the civilian is free to move
2. **in danger** : the civilian is stuck close to a fire

### 2.2.1 Not in danger

When the civilian is not in danger, it starts in a non urgent location called *canMove* where three main events might happen:

1. move command from clock
2. instruction to help another civilian from a drone
3. instruction to call a professionals from a drone

When it receive the command to move, the civilian just move following the chosen moving strategy and then checks if it is close to an exit or a fire.
When it receive instruction to help directly another civilian, it acts as a professional by informing the civilian who is starting to help, and starting a timer. if the civilian in difficulty dies before the timer elapses. In that case the first responder just wait for the timer to elapse and then consider itself *safe* without communicate with the victim.
When it receive instruction to call a professional, it synchronizes with the professional to call and sends information about the civilian in danger. Then it enters a non urgent location and wait for the professional to send the signal that the time has elapsed and so can enter the *safe* location.

### 2.2.2 In danger

When the civilian detects a near fire, it enters the danger section. First of all it starts a timer corresponding to the time after which he dies. Then it enters a non urgent state so that the time can flow, and waits to either be helped or to the timer to be elapsed.

In case a zero or a first responder start to help him, it enters a new location: it will signal to the rescuer if the timer elapses before the rescuing ends. If in this state it receive the signal that the rescuing has completed, it enters the *safe* location, otherwise if the time elapses it enters the *dead* location.

## 2.3 Professional

The Professional template models the behaviors of the first responders. The most important location is the *canMove* location. It represent the idle state of the professional, where it can receive the signals:

- *moveProfessional* from the controller: it means it is the turn of the professional. When this happens, first it checks if there is any civilian in danger to help. If not, it moves according to the specified moving strategy.
- *helpRequest* from a civilian: it means under the suggestion of a drone, a civilian is asking for help. If it happens, it sends a signal that the help request has been accepted and start a timer. If he gets the signal that the civilian in danger died, it still continue the wait for the timer, but it will save only the civilian who asked for help as the drone told him.

## 2.4 Drone

The drone template is mainly composed by a loop divided in two phases:

1. ***Detection phase***: when the drone receive the signal *detect* from the controller, it start scanning the environment for civilians and professionals. Then for each civilian in danger try to match a safe civilian. If a professional is available, the drone chose what action to suggest according to the *droneLimitDecisionDistance* parameter.
2. ***Moving phase***: when the drone receive the signal *moveDrone* it simply move according to the chosen strategy, and goes back into the idle (*canDetect*) location, where time can flow.

## 2.5 Moving patterns

In this section we describe the moving patterns we implemented for the model entities. In particular we implemented the moving pattern by means of functions, and not using Timed Automata, to make our model easily configurable.

- **Random move**: This moving strategy consists in the entity picking pseudo-randomly one of the valid adjacent cells. If there isn't any valid cell the entity simply doesn't move. Most entities will have a well defined moving strategy and make use of the random move only in particular cases as a backup strategy.
- **Move to closest exit**: This strategy is meant to be used by civilians who knows which way is the closest exit. The strategy is just to identify the closest exit, and move towards it along the furthest axis. It the best move is not valid, it tries to move to the exit along the closest axis. If none of them is valid it just follows the random strategy. This strategy

does not try to avoid fires, as it simulates not trained civilians who just try to rush to the closest exit.

- **Move to closest fire**: This strategy is meant for Professionals (first responders) who have been informed about the fire, but don't know anything about where are the civilians in need. This strategy consist in identifying where the closest fire is (we suppose professionals have that information), and then when close enough to that fire, just proceed with random moves hoping to find civilians who needs help. It is possible to regulate how close to the fire they should stay, by tuning the ***targetFireDistance***.

- **Follow straight lines**: This strategy is meant for drones, and simply consist on scanning a row or a column. The trajectory depends on the *droneRadius* parameter, as the drone turns back as soon as it is able to detect until the border of the grid.

- **Square**: This strategy is meant for drones, and consist of following a clock wise square trajectory, with edges of length: *diameter*. This strategy assumes that the position of the drone is initialized in such that the drone can complete the trajectory without exceeding the grid border.

# 3 Stochastic version

In this section we will describe how we built the stochastic version of our model and its main features.
First of all we added two configuration **parameters**:

- **pFail**: probability of a drone detection failure

- **pListen**: probability of a civilian to ignore drone's instructions

In our implementation, in each clock cycle drones have a probability *pFail* of not being able to detect **anything** and *1-pFail* to be able to detect **every** danger situation detectable in its position. This choice is to model some kind of failure in the drone, and not the fact that situations might be harder to detect than others.

In order to make our model stochastic we had to make sure the following **requirements** were satisfied:

1. **All channels must be broadcast**:

   The main difficulty concerning broadcast channels is that transitions can be triggered even if there is no sender. To cope with this problem we applyied the pattern describe in *Uppaal SMC Tutorial chapter 7.1.* When necessary, we added a guard on the broadcast sender which check on a shared variable if the receiver is ready to trigger the transaction receiving that signal.

2. **All delays must be modeled**:

   As timing is managed strictly by the controller, we did not require to make any changes to have the delays correctly modeled.

# 4 Design choices

The first design choice we made was to model the state of the grid using global variables. We define a matrix containing information about every cell, and we used a redundant array for professionals and civilians in order to make information sharing and lookup more efficient.

## 4.1 Controller

We chose to implement the Controller template because it makes much easier to ensure the time flows in a controlled manner and that entities are synchronized in the correct way. In particular in our implementation we sometimes limited the possibility of exploring different transition order. For example, at each round drones will detect emergency situations in the same order and professionals will scan the neighbour cell in the same order. This choice reduces the model complexity and the time of property verification, with the cost of exploring less possibilities. Of course we could let entities chose freely the action order, but the complexity would increase exponentially, making simulation of interesting scenarios unfeasible. To cope with this, we tried to increase the possible random moves explored by our model, by using the randomness variable explained below.

In conclusion we found the use of a controller a good way to simplify our implementation and get it more robust, with the cost of a less general model. Furthermore the use of a controller and the choice or a stricter synchronization makes our model more scalable to large scenarios.

## 4.2 Randomness

This section is very important, as we decided for a stricter synchronization, we need to make sure to cover enough cases in our simulation.

The easiest way to create the randomness needed to make random moves, is by using the *select* functionality of the uppaal timed automata, before every random move. However by doing so we found the computational complexity explodes soon, making the verification of larger models unfeasible. To cope we this we implemented a tunable pseudo-random generator, good enough to explore different move sequences. The main parameter called *randomness* defines how many different cases to test in the simulation. The randomness is created by initializing $randomseed \in [0, randomness - 1]$ at the beginning of the simulation, using just one *select*. Every time an entity utilize the random seed to get randomness, it updates it using the formula:

$$randomSeed = ((randomSeed + id) \oplus time) \mod randomModulus$$

Where $randomModulus = max(4, randomSeed)$.
We chose to implement randomness in this way to allow us to have a better control on the trade-off between the exploration of every possible random decision and the computational cost to verify properties in each possible scene modeled.

# 5 Grid layouts

In this section we show the most interesting scenarios we tested and why we found them interesting. In the following scenarios we choose to assign to civilans the moving strategy move to closest exit, and for professionals move to the closest fire.

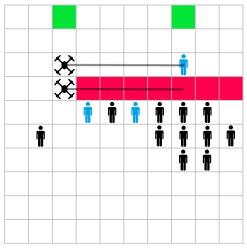

Figure 1: Test Layout

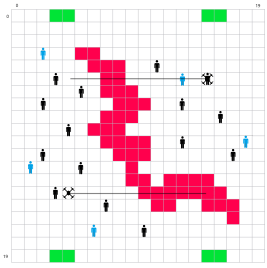

Figure 2: Layout A

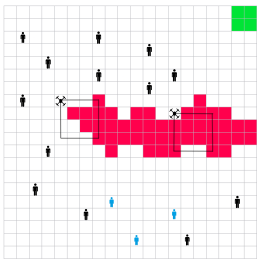

Figure 3: Layout C

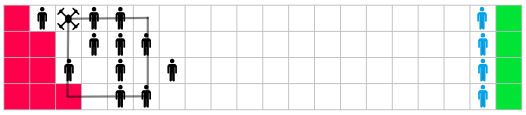

Figure 4: Layout B

## 5.1 Test Layout

The first scenario we present is an example of how we built different layout to make sure to test every corner case. This is not meant to infer knowledge on the problem, but just to test the correctness of our implementation.

## 5.2 Layout A (big grid)

The first layout we implemented represent a generic situation where civilians and professionals are randomly spread across the grid. The fire is also well spread, and there emergency exits on the top and bottom of the grid.
This scenario is meant to test the entities behaviour in the more general situation, and to better understand the computational limits of our model, as the layout is pretty complex. There are two drones that scan horizontally the grid with a *droneRadius* that allows the to scan almost all the grid.

## 5.3 Layout B (narrow grid)

The second scenario is meant to model a narrow and crowded area. Moreover in this situation, first responders are arriving on the scene from the only available exits. In this scenario we expect the civilians who tries to rush to the closest

exit to be hindered by the presence of other civilians. More over in this scene professionals will try to reach the fire area, clashing with civilians trying to reach the exit.
In this case we expect the presence of a drone to be more beneficial, as it would encourage civilians collaboration while professionals struggle to reach the scene.

### 5.4 Layout C (fire in the middle)

In this layout there is a wide fire that almost divide the grid in two halves, where only one side has an emergency exit. While civilians and spread in both of the sides, *professionals are only on the side without any exit.*
We expect professionals will mainly help civilians on the side of the fire as unlikely will be able to get to the other side of the fire. In this case we hope the presence of drones will be beneficial for two main reasons:

- encourage civilians collaboration in the area that won't be reached by the professionals.
- enable civilians to get in touch with professionals across the fire, as drones can detect and communicate with both hovering on the fire

## 6 Property Verification

In this section we explained how we utilized the verification functionality of Uppaal to obtain knowledge from our model.

Note that we will use the terms *is guaranteed* and *is possible*, in the sense that is guaranteed or possible in every situation covered by our model in that specific simulation. Even if this may not be the same as in every possible situation, it is enough to infer knowledge in many cases.

When not specified otherwise, the chosen settings are:
$N_v : 4,\ T_v : 30,\ T_{fr} : 5,\ T_{zr} : 10$
$targetFireDistance : 2,\ \ droneLimitDecisionDistance : 5$

### 6.1 Model correctness

First of all we tried to verify the correctness of our model. To make sure to capture each corner case we developed other scenarios (test layout is one of those). The main corner cases we made sure to cover are:

- More drone can detect the same danger situation
- More professionals can help the same civilian
- Civilian die while being helped

After testing the model using the simulator, we used the verifier to check some properties. in particular we made sure to check:

- no deadlocks (not in stochastic version)
- no zeno paths
- eventually will enter the finish state

$$A\Box \ not\ deadlock$$

$$Controller.moveTimer == 0 \rightarrow Controller.moveTimer == 1$$

$$Controller.starting \rightarrow Controller.finish$$

## 6.2 Mandatory properties

To verify the mandatory properties in the non-stochastic model we introduced some integer variables between 0 and 100 representing an approximation of the percentage of survivors and victims.

$$survivorsPerc = fint(survivors/Ncivilians * 100)$$

Each time the survivors count is updated, even the percentage is updated. This is done because in non stochastic model double variables cannot be used in the property verification.

The implemented mandatory properties are:

$$E\Diamond(globalTime \leq T_{scs} \wedge surivorsPerc \geq N_{\%})$$

$$A\Diamond(globalTime \leq T_{scs} \wedge surivorsPerc \geq N_{\%})$$

### 6.2.1 Layout A (big grid)

We chose for this scenario a randomness value equal to 100, as it is a pretty big simulation.

1. **With drones enabled:**
   - NOT possible to have $\geq$ 90% survivors in 50 units of time
   - guaranteed to have $\geq$ 85% survivors in 50 units of time
2. **With drones disables:**
   - NOT possible to have $\geq$ 90% survivors in 50 units of time
   - guaranteed to have$\geq$ 85% survivors in 50 units of time

In this scenario the use of drones is not much effective, because the civilians are to spread and the drones hover on a wide area, and are not able to detect other civilians to help the one in danger. In particular the problem is that when drones arrive close to civilians in danger, it is too late, because all the other civilians which could help have already reached the closest exit. The solution to this problem would be to allow drones to contact first responders without the help of a civilian.

### 6.2.2 Layout B (narrow grid)

In this scenario professionals needs time to get close to the fire as they run into the civilians rushing to the exit. We set the randomness to 1000, because the scenario is not too big, but we expect there will be many random moves, as entities won't often be able to perform the better move. In particular we verified:

1. **With drones enabled:**
    - possible to have $\geq 70\%$ survivors in 15 units of time, but no $\geq 75\%$
    - guaranteed to have $\geq 100\%$ survivors in 30 units of time
2. **With drones disables:**
    - possible to have $\geq 35\%$ survivors in 15 units of time, but no $\geq 40\%$
    - guaranteed to have $\geq 70\%$ survivors in 30 units of time, but no $\geq 75\%$

These figures shows drones can substantially increase the number of survivors and also in some cases reduce the rescue time.

#### 6.2.3 Layout C (fire in the middle)

In this scenario we used a randomness value of 200, as it is a bigger scenario, requiring more time. However the choice of smaller randomness does not affect much the simulation because there are going to be less situation requiring random moves. In this scenario we also increase the droneRadius to 5, as it is needed to hover on a bigger fire.

1. **With drones enabled:**
    - possible to have $\geq 100\%$ survivors in 30 units of time
    - guaranteed to have $\geq 90\%$ survivors in 30 units of time
2. **With drones disables:**
    - not possible to have $\geq 85\%$ survivors in 30 units of time
    - guaranteed to have $\geq 80\%$ survivors in 30 units of time

This shows that with drones, the layout C has a percentage of survivors $\geq 90$, while without drones the percentage is bounded between $perc \in [80, 85)$. So we can conclude that drones has a great positive effect, but they need to have a bigger radius to be able to communicate across the sides of the fire.

### 6.3 Stochastic properties

In the stochastic model, the two properties collapses in one, showing the probability of validity of the disequality:

$$Pr[\leq T_{scs}](\Diamond(survivors/Ncivilians) \geq N_{\%})$$

#### 6.3.1 Layout A (big grid)

As shown in the previous chapter, in this scenario drones are not effective. In fact we verified that in some corner cases, the drone is actually employed, but the rescuer is to late to actually save the civilian.

$T_{scs} : 50$ $N_{\%} : 85\%$

- pFail: 0.1, pListen: 0.9 , **probability**: $\geq 95\%$

- pFail: 0.9, pListen: 0.1 , **probability**: $\geq 95\%$

As expected the tuning of the probabilities doesn't affect significantly the outcome of the simulation, as human-drone interaction is not effective in this case.

#### 6.3.2 Layout B (narrow grid)

$T_{scs} : 30$ $N_{\%} : 95\%$

- pFail: 0.1, pListen: 0.9 , **probability**: 80.9%
- pFail: 0.3, pListen: 0.9 , **probability**: 62.2%
- pFail: 0.1, pListen: 0.7 , **probability**: 58.6%

In this case drones effective communication success is fundamental, and probabilities decrease rapidily. In fact if drone-civilian communication first attempts fails, then all civilians get far from the fire, and don't have any chance to try again.

#### 6.3.3 Layout C (fire in the middle)

$T_{scs} : 30$ $N_{\%} : 95\%$

- pFail: 0.1, pListen: 0.9 , **probability**: 96.4%
- pFail: 0.3, pListen: 0.9 , **probability**: 83.6%
- pFail: 0.1, pListen: 0.7 , **probability**: 82.3%

The probabilities decreases slower in this case. Even if communication fails at first, there are more opportunities to retry as the civilians are well spread and stay close to the fire area for a longer time.

## 7 Conclusions

In conclusion we found the action of drones is beneficial in most cases. In particular drones are more effective when:

- The area is very crowded and professionals struggle to reach the fire area (Layout B)
- When professionals can reach the fire area, but struggle to find the civilians to help (Layout C). However this situation might needs drones with a bigger detection radius.

More over some critical situation highlighted by the stochastic analysis are:

- Drones must be able to detect civilians in danger before all the other people has escaped, otherwise the drone won't find anyone to call for help.
- Drone mulfunctions and civilian hesitation is most problematic in scenarios where everyone is moving away from the fire, leaving alone the ones struggling to escape. In other cases it less problematic because the drone has more opportunity to retry and insist on hesitating civilians.