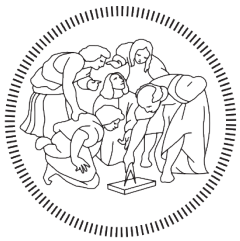

POLITECNICO
MILANO 1863

Formal analysis of search-and-rescue scenarios

Homework Project

Course name

Formal Methods For Concurrent and Real-Time Systems
A.Y. 2023/2024

Instructors

Prof. Pierluigi San Pietro
Dr. Livia Lestingi

Authors

Alberto Noris
Lorenzo Prosch

Contents

0 Abstract

This document presents an UPPAAL model for search-and-rescue scenarios, the design choices behind it, meaningful properties and their formal verification, with the final objective of highlighting its strengths and weaknesses. The modelled problem is the search and rescue of civilians during an emergency, carried out by trained professional assisted by autonomous drones. In this document it is assumed that the reader has full access to the described UPPAAL model and the problem description.

1 Model Description

From here after the word *entity* refers to a civilian, first-responder or drone (and all of their subcategories), while *object* refers to fire and exit. Each entity is uniquely identified among other entities of the same type by an id. While going through the different aspects and sections of the model, the corresponding design decisions will also be directly highlighted and explained.

1.1 Layout

Map and Positions The modelling of the emergency area is achieved by a 2D array, called for simplicity map. In addition, the position of each entity is specified in its respective 1D array, where the particular entity is identified by its index and each element of the array is a pair of coordinates (referring to the map).
Each map cell contains only one value, and each cell value type has a different meaning. The most meaningful and critical decisions to explain are two. The first one is the absence of a drone-value in the map, deemed superfluous and a complicating factor due to the following considerations: drones can fly over other entities and objects, and the drone moving policy adopted (explained in section 1.4.4). The second point is the values of *SAFE* and *CASUALTY*, which are different in principle (to distinguish the two states inside model logic), but then are written in the map as the same, *FREE*. This last choice was done because when civilians are saved or die, they are no longer part of the scenario, so the cell that they once occupied should be now free and their position in the respective array removed.

General Movement and Detection Entities can move following their respective moving policy inside the map, updating it and their position arrays. The following *simplifying assumption* was made: entities can only move in the four cardinal directions. From this point stems the fact that distances between cells is calculated via Manhattan distance. This choice was made to simplify the moving policy without loss of generality, since the model can be easily extended with eight-direction movement without impacting the overall model and logic. On the other hand, entities can detect other entities and objects in all directions. This choice was made to simulate the realistic behaviour of entities, which could not be simplified like for the moving policy. This decision places greater emphasis on the detection policies rather than on the moving policies.

1.2 Communication

As it will be clearly explained in future sections, entities have the need to communicate in a specific fashion.

Channels A given entity need to synchronize with a specific entity over a channel, to achieve this binary correspondence channel arrays were utilized, where the specific target entity for that channel is identified by its id. Each channel is clearly named according the communication that it represents.

Message Passing In addition to synchronization, entities also need to pass information between each other and to solve this problem a communication mechanism was implemented. At the core of the system there are messages (single or double, depending on the number of values contained), then each entity has its own inbox containing a single message and finally all of them are grouped into a single array of inboxes (identified by their id), contained in a global structure (communication purpose specific). The model simulates a postman carrying letters to different recipients, managing all of the message passing and cleaning the inboxes whenever necessary (if not strictly necessary an inbox is not cleaned).

1.3 Termination

In a real-life scenario a rescue operation cannot continue indefinitely, drones will run out of battery, first-responders will get tired and civilians who are not rescued in time will become missing civilians. To model this important aspect, every entity has its own ”tiredness meter” implemented by an integer in their respective indexed arrays (properly named). The single counter measures how much an entity is ”tired” and not how much energy is left (for ease of implementation without clarity loss). Whenever, an entity does a meaningful amount of work (which also is worth to be accounted for), its counter will increases until it reaches its ”max tiredness threshold” after which the entity will not be able to do any more work. This aspect will not be further discussed in the templates, if not to point out interesting and meaningful details. Each entity category has its own single threshold, this could be expanded into a new parameter for each entity.

1.4 Templates

The model is based on the following 4 templates:

- *Initializer*: helping template to initialize the scene and the various used data structures.
- *Civilian*: person in need of rescue inside the emergency area, who can be saved, die or missing and who can receive instructions by a drone.
- *First-Responder*: trained professional rescuing civilians inside the emergency area.
- *Drone*: autonomous flying entity, which has the objective of instructing civilians in the best way possible to maximizing rescue rate.

1.4.1 Initializer

Simple automaton to correctly populate the map with entities and to initialize all the global data structures. Following this initialization all the other automata are signalled to start.

1.4.2 Civilian

A civilian can be:

- *CIVILIAN*: default civilian, also referred to as survivor.
- *SAFE*: saved civilian.
- *VICTIM*: civilian near a fire.
- *CASUALTY*: civilian no longer alive.
- *IN_ASSISTANCE*: civilian who is being assisted by a responder.
- *ZERO_RESPONDER*: civilian who was instructed by a drone to help a victim.
- *IN_CONTACT*: civilian who was instructed by a drone to contact a first-responder.
- *EXHAUSTED_CIVILIAN*: civilian who is too tired to move.

All these different subcategories were used to explicitly model the different conceptual states in which a civilian can be in and their evolution during a scenario. In addition, this clear distinction enables the model to utilize more precise, easier and cleaner logic and to avoid many problems related to miss-communication, miss-detection and multi-communication conflicts.

A civilian is characterized by the following parameters:

- *civilian_id*
- *tv*: time in seconds before a victim dies, becoming a casualty.
- *tzr*: time in seconds needed for a zero-responder to rescue a victim.

Behaviour What follows is the general behaviour and the most important aspects of the civilian template, not all transitions, states and details will be discussed.
A civilian can check its surroundings up to 1-cell distance and it is has no knowledge about the layout of the emergency area (*simplifying assumption*).

1. A civilian starts off by going through three initial committed states in which it is checked if it is *safe*, *victim* or *exhausted civilian*, if none of those are true it ends up in an *Idle* state. The committed nature of these first states is fundamental, since no other entity can move apart for civilians and this assures that the model behaves as intended. Because at each step of a civilian, before any other entity can do anything, that civilian will correctly update its state, not misleading any other entity by being in a wrong state. In addition, this also works perfectly for the initialization phase of the civilians in the map.

2. When a civilian becomes a *victim*, it will become a *casualty* after *tv* time. However, if in this period of time it is assisted by a responder, it can be saved. If the civilian is *in assistance* and becomes a *casualty* this event will be synchronized with the responder. What happens when a civilian becomes *safe* or *casualty* inside the map has been already described in section 1.1.

3. When a civilian becomes an *exhausted civilian*, it will be unable to move any further. If a first-responder sees it, the civilian can still be rescued and saved. However, if all drones and first-responders have run out of energy, the rescue is considered completed and the civilian will become a missing civilian. There is no value type for this last status, since it is not needed in the map.

4. From *Idle* state, a *civilian* can be contacted by a drone or it can move randomly (*civilian moving policy*).

5. When it is contacted by a drone, it can be instructed to become a *zero-responder* or an *in contact* civilian. The committed states are fundamental to create atomic transitions.

   (a) In *zero-responder* case, the civilian will read the message sent by the drone to know which *victim* to assist. Then, it will perform the rescue as responder, assisting itself and the communicated victim. If the *victim* becomes a *casualty* before the end of the rescue, the *zero-responder* will still end its own rescue, simulating the fact that it now knows where the exit is and it can head there on its own.

   (b) In *in contact* case, the civilian will read the message sent by the drone to know which *first-responder* to contact and which *victim* is in need of assistance. Firstly, the civilian enacts the moving towards the *first-responder* to contact it, taking *distance between itself and the first-responder* time. When it arrives, it waits until the *first-responder* is free and then communicates who is in need of assistance. After that, the contacting civilian will be assisted by the first-responder.
   It might happen that while the civilian is reaching the first-responder or is waiting for it to be free, the *victim* is no longer in need of assistance (be it already saved, a casualty or in assistance) and in this case the civilian will not contact the first-responder. This choice was made not to waste the precious time of the first-responder. Finally, the first-responder does not assist the contacting survivor even if it is waiting near itself, because the contacting survivor is not in need of assistance in that moment and the rescue had not yet started. So, the first-responder should prioritize rescuing other civilians who are in need (in a real-life scenario the first-responder could simply tell the contacting survivor where the nearest exit is).

Stochastic Version To account for the stochastic features the civilian template has been expanded:

- New civilian parameter *plisten*: probability expressed by an integer [0, 100], that a civilian when given an instruction by a drone follows it.

- New probabilistic transition: when a civilian is contacted by a drone, now there is a branch point. A civilian will follow the instruction (accepting the communication) only with probability *plisten*, while it will ignore it with probability *100 - plisten*. This simulates the unpredictability of untrained civilians under stress.

- New channel *civilian_noncompliance_msg* and global communication structure *global_drone_comm_-civilian*: these two additions (and the necessary ones that follow) are necessary to make the drone aware that the civilian did not follow the given instruction and to act accordingly.

1.4.3 First-Responder

A first-responder can be:

- *FIRST_RESPONDER*: default first-responder.
- *CONTACTED_FIRST_RESPONDER*: first-responder who is being contacted by a survivor.
- *FATIGUED_FIRST_RESPONDER*: first-responder who is too tired to continue the rescue.

This differentiation was made to ensure that: no two drones could detect the same first-responder and no drone could detect a *fatigued first-responder* (according to the decision policy of the drone described in section 1.4.4), and to give a higher priority to the assisting of a contacting survivor and the corresponding victim, since this action can save two civilians instead of one.

A first-responder is characterized by the following parameters:

- *first_responder_id*
- *tfr*: time in seconds needed for a first-responder to rescue a victim.

Behaviour What follows is the general behaviour and the most important aspects of the first-responder template, not all transitions, states and details will be discussed.
A first-responder can check its surroundings up to 1-cell distance and it has knowledge about the objects layout in the emergency area (*conceptually* needed to assist others and not die).

1. A first-responder starts off in an *Idle* state. From here, respecting the precedence of actions earlier discussed, if it is not fatigued it can check its surroundings to look for a *victim* or *exhausted civilian* to assist. If there is more than one civilian detected, it will try to save the one with the largest *tv* (*first-responder detection policy*). This decision stems from the observation that in a real-life scenario the first-responder must make a decision regarding who to help first and since the *tfr* is fixed (every assistance takes the same amount of time), it makes sense that the first-responder will try to save the civilian with the highest possibility to live. The model could be extended to work with variable *tfr* depending on different factors (different rescue types should require different *tfr*s) or it could support different priorities for different civilian types (e.g. children, men, women, elderly, etc.).

   (a) If a first-responder detects a *victim* to help, it will start the rescue and complete it in *tfr* time. If the *victim* dies during the rescue becoming a *casualty*, then the rescue is aborted.

   (b) If a first-responder does not detect a *victim* to help, it will move randomly (*first-responder moving policy*), it has no knowledge about civilians' positions. It is worth pointing out, that the first-responder also has as a valid option to remain in the same position. This decision was made for consistency purposes, because in the model a first-responder can assist civilians also in impossible situations (e.g. exit completed surrounded by fire) and so a first-responder should still be able to assist civilians even if it is trapped (unable to move), and to do so, it needs to transition from *Wandering* state to *Idle* state without moving.

   The fact that after the detection of a *victim* there is a committed state is fundamental to ensure atomic actions and avoid conflicts, since no other entity should be able to detect it as a *victim* (e.g. other first-responders or drones) and act upon that detection. This models a possible and plausible rescue coordination mechanism among entities during an emergency.
   Moreover, it is also fundamental the distinction between this committed state and the *Wandering* state, in order to avoid possible deadlocks (also prevented by the discussed moving policy).
   No other interaction can happen when a first-responder is fatigued, because it cannot be detected by drones. To achieve this, it was mandatory to use the function *fEnergyUsed* when detecting victims, even tough is not completely true in real-life, since the first-responder should get tired after performing the action and in this way the first-responder is already fatigued when it will make the last rescue or step.

2. When a first-responder is being contacted (*contacted first-responder*), after finishing its previous rescue or movement, it will assist the *contacting survivor* and the communicated *victim*. It will help both of them in *tfr* + *distance between itself and the victim* time. However, if in this period of time the *victim* dies becoming a *casualty*, the first-responder will still finish rescuing the *contacting survivor*, since the rescue has already started and completely aborting it would mean losing all the progress already made, wasting precious time.

1.4.4 Drone

As discussed in section 1.1, drones are not represented in the map (no value type). Drones have been modelled as if they would be all deployed from one side of the map (without loss of generality, from the top) with the aim of maximizing the covered area, so they are meaningfully spaced apart. Not only this design choice increases their rescue capabilities, but it also prevents most communication conflicts between them.

A drone is characterized by the following parameters:

- *drone_id*
- *nv*: range in cells in which a drone can detect other entities.

Behaviour What follows is the general behaviour and the most important aspects of the drone template, not all transitions, states and details will be discussed.

1. A drone starts off in an *Idle* state. From here it can scan its surrounding area to detect meaningful entities, if it is not energy drained. The *drone detection policy* is as follows: it will scan within a square radius of *nv* cells its surroundings, looking for *survivor*s, *victim*s and *first-responder*s; if there is more than one entity for a specific category, it will choose the closest one. The following assumption was made: a drone does not have the capability to choose the victim with the highest possibility to live (like a first-responder can). The drone can only detect *victim*s and not *exhausted civilian*s, since it cannot distinguish between them and a normal *civilian*s. In addition, if the model would be extended to support variable first-responder's *tfr*, the drone could have the capability to use this additional information to make better decisions; the same consideration applies with the addition of civilian's priorities.

   (a) If a drone detects a *survivor* and *victim* pair, it will start the communication with the survivor and then give instructions according to the *drone decision policy*. If the drone during its scanning also detected a *first-responder*, then it will instruct the survivor to become an *in contact* civilian, giving it the additional information needed about which first-responder to contact and which victim needs assistance. On the contrary, if there is no near *first-responder*, the drone instructs the survivor to become a *zero-responder*. This design choice ensures the utilization of first-responders' expertise to help two civilians, instead of one, whenever possible, while also offering the option of helping the victim in need by providing instructions to a fellow civilian to assist it.

   (b) If a drone does not detect a *survivor* and *victim* pair, it will move following the *drone moving policy*. When drones are deployed into the map, they will start from the top edge and move downwards in a straight line. Then to change direction and come back, the drones do not fly until the limit of the map, but they take into consideration their *nv*s in order to cover more efficiently the map. This refinement is achievable by using simple sensors, that should already be present on the drones. This moving policy has been chosen for its simplicity and practicality in real-life scenarios, because this drone fleet configuration is much easier to control, it covers the most area without using more complex policies and it helps avoiding conflicts between drones, which would create additional confusions in an already difficult and chaotic situation.

The following are some additional considerations worth pointing out:

- When a drone decides to instruct a survivor to contact a first-responder, the drone itself sets the first-responder value type to *contacted first-responder*. This is done to ensure no conflicts among drones,

and could be achieved in real-life by drones communicating which first-responder has been assigned (recognizing the first-responder by a chip tag). This could also be extended to support quasi-real-time monitoring of first-responders.

- The drone's committed states are fundamental to ensure atomic actions and avoid conflicts, since no other entity should be able to detect the detected *survivor*, *victim* or *first-responder* (e.g. other drones or first-responders) and act upon that detection. Moreover, entities should not move and change state (e.g. civilians or first-responders) while the detection and decision is taking place. The *Patrolling* state has been also marked as committed, even tough not strictly mandatory. This was done to convey the idea of the atomic action that a drone makes during its internal processing.

Stochastic Version To account for the stochastic features the drone template has been expanded:

- New drone parameter *pfail*: probability expressed by an integer [0, 100], that the detection of entities done by the scanning of a drone does not work.
- New probabilistic transition: when a drone detects a *survivor* and *victim* pair, now there is a branch point. A drone will start the communication only with probability *100 - pfail*, while it will ignore the detection with probability *pfail*. This simulates the defect affecting drones' vision sensors.

2 Properties

2.1 Scenarios

We have designed three scenarios to evaluate our emergency response system: Urban High-Rise Fire, Large-Scale Natural Disaster, and Industrial Complex Emergency. The following table summarizes their key parameters:

Parameter	Scenario 1	Scenario 2	Scenario 3
Environment	Urban High-Rise	Natural Disaster	Industrial Complex
Map Size	8x10	20x14	18x12
Drones (visibility)	2 ($n_v = 2$)	2 ($n_v = 3$)	4 ($n_v = 1$)
First Responders (rescue time)	2 ($t_{fr} = 2$-3)	5 ($t_{fr} = 8$)	5 ($t_{fr} = 8$)
Civilians (vulnerability, zero-responder rescue time)	3 ($t_v = 2$-11, $t_{zr} = 1$-2)	11 ($t_v = 15$-25, $t_{zr} = 6$)	10 ($t_v = 15$-25, $t_{zr} = 6$)

Table 1: Summary of Emergency Response Scenarios

These scenarios assess system performance in different conditions, such as dense urban vs. widespread disaster areas, limited vs. advanced drone capabilities, and varying civilian vulnerability.

2.1.1 Scenario 1: Urban High-Rise Fire

Tests efficiency in a compact, high-stakes environment. The compact map simulates a dense urban environment with advanced drones with for example thermal imaging capabilities. Limited responders reflect initial teams, while varied civilian vulnerabilities test prioritization in time sensitive situations.

2.1.2 Scenario 2: Large-Scale Natural Disaster

Tests handling of complex, large-scale situations with limited but advanced resources. The large map represents a widespread disaster area. Advanced drones test efficient resource allocation over a vast area. Similar high rescue times reflect environmental challenges, while high civilian vulnerability simulates a population caught unprepared.

2.1.3 Scenario 3: Industrial Complex Emergency

Tests operations in a hazardous environment with low-visibility drones. The medium-sized map simulates varied industrial terrain. Low-visibility drones reflect operational challenges in potentially hazardous conditions. High rescue times account for the complex environment, while varied civilian vulnerabilities represent different risk levels across the industrial complex.

2.2 Verification Results (Non-Stochastic)

This section discusses the results of two queries on three systems to assess the efficiency and reliability of civilian rescue operations under various constraints. The key variables were $N\%$ (percentage of civilians to be rescued) and T_{scs} (time constraint for rescue).

The queries were:

- `E<> n_safe >= n_percentage && global_time < t_scs && initializer.Initialized`
- `A<> n_safe >= n_percentage && global_time < t_scs && initializer.Initialized`

These queries evaluate the possibility (`E<>`) and inevitability (`A<>`) of rescuing a certain percentage of civilians within a time frame.

For the tables 'V' indicates a successful query, while '∞' means that the query was running for a long time meaning the result could be successfull or not (but is very likely it will eventually turn not sucessfull).

2.2.1 System 1 Results

The second query, checking if all paths lead to rescuing at least $N\%$ of civilians within T_{scs}, always returned false, indicating no guaranteed rescue within the time constraints. The first query, checking if at least one path allows rescuing $N\%$ of civilians within T_{scs}, gave more varied results, summarized below:

$T_{scs} \backslash N\%$	25%	50%	75%	100%
2	∞	∞	∞	∞
3	V	∞	∞	∞
4	V	V	V	∞
5	V	V	V	V

Table 2: System 1 Query 1 Results

These results show that significant rescues are possible given enough time, highlighting the importance of quick response and efficient resource allocation early in rescue operations.

2.2.2 System 2 Results

For the second system, our analysis revealed distinct patterns in the rescue operation's dynamics. The outcomes for the first query are summarized in the following table:

$T_{scs} \backslash N\%$	25%	50%	75%	100%
1	V	∞	∞	∞
2	V	∞	∞	∞
5	V	∞	∞	∞
10	V	∞	∞	∞
100	V	V	∞	∞

Table 3: System 2 Query 1 Results

For the first query, the results show that it returns true if and only if the number of people to be saved is 1, regardless of the time constraint.

These findings suggest that System 2 has a more constrained rescue capability compared to System 1. It provides a consistent level of performance for small-scale rescues but faces significant challenges in scaling up to larger percentages of the civilian population.

2.2.3 System 3 Results

System 3 exhibits characteristics similar to System 2, with the difference that query 2 returns always false.

Key observations for System 3:

1. The system maintains the possibility of saving up to 25% of civilians across all time frames.
2. As with System 2, increasing the time constraint does not improve rescue potential for larger groups.
3. The false result for Query 2 suggests that even saving a single civilian is not guaranteed under all circumstances.
4. The system shows no successful scenarios for rescuing 50% or more of the civilians.

System 3 presents a more challenging rescue environment compared to Systems 1 and 2. While it retains the potential for medium-scale rescues, it lacks any guaranteed successful outcomes, indicating a higher level of uncertainty in rescue operations.

2.3 Verification Results (Stochastic)

In addition to the deterministic queries, we decided to conduct the stochastic analysis on the second scenario using the following query:

```
Pr[<=t_scs](<> n_safe >= n_percentage && initializer.Initialized)
```

Key findings include:

- **Guaranteed Minimal Rescue:** (trivial) When the target was to save just one civilian, the query returned a 100% pass rate, regardless of the time constraint.
- **Time-Dependent Success Rate:** For saving 25% of civilians:
  - With $t_{scs} = 10$, the success probability was 30%.
  - With $t_{scs} = 2$, the success probability decreased to 17%.

These results indicate that while single-civilian rescue is highly reliable, the success rate for larger-scale rescues (25% of civilians) is significantly time-dependent, with longer time frames improving the chances of success.

3 Conclusions

Our analysis of search-and-rescue scenarios using UPPAAL has yielded several important insights:

1. The effectiveness of rescue strategies varies significantly across different emergency situations, underscoring the necessity for adaptable approaches.
2. Time constraints emerge as a critical factor in rescue operations. Extended timeframes generally correlate with higher success rates, particularly when attempting to rescue larger groups of civilians.
3. The stochastic analysis reveals a stark contrast between single-civilian and large-scale rescues. While individual rescues demonstrate high reliability, operations involving a larger percentage of civilians (e.g., 25%) are considerably more time-sensitive and less predictable.

4. Notably, none of the modelled systems achieved a consistent 100% rescue rate, highlighting the inherent complexities and uncertainties in emergency response scenarios.

5. The model effectively illustrates the intricate interplay of various factors, including environmental conditions, resource availability (responders and drones), and civilian vulnerability levels.

6. Potential areas for improvement include optimising resource allocation, enhancing the strategic capabilities of drones and responders, and developing more versatile systems capable of adapting to diverse emergency contexts.

In conclusion, this study demonstrates the significant value of formal modelling techniques in analysing and enhancing emergency response systems. While providing valuable insights into current capabilities, it also illuminates promising avenues for future research and development in this critical field.