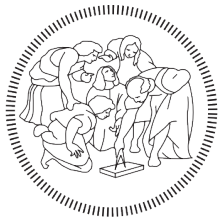

POLITECNICO
MILANO 1863

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

Part I
title

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 the 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 civilians, first-responders or drones (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 type value 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 1.3.4). 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 and the cell that they once occupied is now free.

General Movement and Detection Entities can move following their respective moving policy inside the map. 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 the Manhattan distance. This choice was made to simplify the moving policy without loss of generality, since the model can be extended to work with movement in all eight directions 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 idealistic 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 need to communicate in a particular and 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 will have its own inbox containing a single message and at the end all of them are grouped into a single array of inboxes, 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 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 or die 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.

1.3.1 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.

1.3.2 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*: victim 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.

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 logics and to avoid many problems related to miss-communication or multi-communication.

A civilian is characterized by the following features:

- *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.

1. A civilian starts off by going through two initial committed states in which it is checked if it is *safe* or *victim*, if neither of those are true it ends up in an *Idle* state. The committed nature of these first two states is fundamental, since no other entity can move apart for civilians (can interleave their actions) 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. From the *Idle* state, a *civilian* can be contacted by a drone or it can move randomly (*civilian moving policy*).

4. 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 transactions, while the non-committed ones are needed to let time pass.

   (a) In the *zero-responder* case, the civilian will read the message sent by the drone to know which *victim* to assist. After that it will perform the rescue as a responder. If the *victim* becomes a *casualty* before the end of the rescue, the *zero-responder* will still end its own rescue.

(b) In the *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 for the *first-responder* to be free and then communicate who needs assistance. After that, it will be saved by the first-responder.
It might happen that while the civilian is reaching the first-responder or is waiting for the first-responder 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. In addition, it is based on the decision policy of the drone (described in section 1.3.4). Finally, the first-responder does not assist the contacting survivor even if it is waiting (near the first-responder), because the contacting survivor is not in need of assistance and the rescue has not yet started. So the first-responder should prioritize rescuing other victims, while (in a real scenario) just instructing the contacting survivor to go to the nearest exit.

Stochastic Version TODO

1.3.3 First-Responder

A first-responder can be:

- *FIRST_RESPONDER*: default first-responder.
- *CONTACTED_FIRST_RESPONDER*: first-responder who is being contacted by a survivor.

This differentiation was made to ensure that: no two drones could detect the same first-responder (according to the decision policy of the drone (described in section 1.3.4)) and to give a higher priority to the assisting a contacting survivor and the corresponding victim, because this action can save two civilians instead of one.

A first-responder is characterized by the following features:

- *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.

1. A first-responder starts off in an *Idle* state. From here, respecting the precedence of actions earlier discussed, it can check its surroundings to look for a *victim* to help. If there is more than one *victim* detected, it will try to save the one with 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 victim with the highest possibility to live. The model could be extended to work with variable *tfr*, depending on different factors to simulate this aspect of an emergency rescue (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 and the first-responder returns to the *Idle* state.

(b) If a first-responder does not detect a *victim* to help, it will move randomly (*first-responder moving policy*). 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, 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).

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 *victim* communicated by the survivor. 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.

1.3.4 Drone

As discussed in section 1.1, drones are not represented in the map, so they do not have any value type.
Drones have been modelled as if they would be all deployed from one side of the map (without loss of generality, 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 (not all) communication conflicts between them.

A drone is characterized by the following features:

- *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. 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). 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; 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 (and in a reasonable distance), 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 its *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 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. 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, since no other entity should be able to detect the detected *survivor*, *victim* or *first-responder* (e.g. other drones) and act upon that detection. Moreover, entities should not move and change state (e.g. civilians) while the detection and decision is taking

place. The *Patrolling* state has been also marked as committed, even tough not strictly mandatory, in order to convey the idea of the atomic action that a drone makes during a scenario.

Stochastic Version TODO

2 Properties

2.1 Non-stochastic Scenarios

In order to comprehensively evaluate the effectiveness and adaptability of our emergency response system, we have designed three distinct scenarios that simulate a range of real-world emergency situations. These scenarios are carefully crafted to test various aspects of the system's performance under different conditions, each presenting unique challenges and constraints.

The scenarios we have developed are:

- Urban High-Rise Fire
- Large-Scale Natural Disaster
- Industrial Complex Emergency

Each scenario is characterized by a specific set of parameters that define the operational environment, available resources, and the nature of the emergency. These parameters include:

- Map dimensions
- Number and capabilities of drones
- Number and efficiency of first responders
- Number and vulnerability of civilians

By varying these parameters across scenarios, we aim to assess how our system performs under different conditions, such as:

- Dense urban environments vs. widespread disaster areas
- Limited vs. advanced drone capabilities
- Varying levels of civilian vulnerability and self-rescue ability

The following sections will detail each scenario, explaining the rationale behind the chosen parameters and the specific aspects of system performance they are designed to evaluate.

2.1.1 Scenario 1: Urban High-Rise Fire

This scenario simulates a fire in a multi-story urban building, testing the system's efficiency in a compact, high-stakes environment.

Parameters:

- Map size: 8x10
- Drones: 2 (high visibility: $n_v = 2$)
- First Responders: 2 (moderate rescue time: $t_{fr} = 2$-3)
- Civilians: 3 (varied vulnerability: $t_v = 2$-11, moderate zero-responder rescue time: $t_{zr} = 1$-2)

Rationale:

- The compact map size (8x10) represents a dense urban environment, typical of a high-rise building floor plan. This layout challenges the system to navigate tight spaces and make quick decisions.
- Two drones with high visibility ($n_v = 2$) simulate advanced thermal imaging capabilities, crucial for locating civilians through smoke and obstacles in a building fire. Their effectiveness in this enclosed space is key to rapid response.
- The moderate number of first responders (2) with varied rescue times ($t_{fr} = 2$-3) reflects the reality of initial emergency teams entering a burning building. Their slightly different rescue times account for varying levels of experience or equipment.
- Three civilians with diverse vulnerabilities ($t_v = 2$-11) represent a mix of building occupants, from those in immediate danger to those with more time to be rescued. The moderate zero-responder rescue times ($t_{zr} = 1$-2) indicate that some civilians might attempt self-rescue or assist others, typical in a fire scenario where immediate evacuation is crucial.

This scenario tests the system's ability to prioritize rescues in a time-sensitive, spatially constrained environment where every decision is critical.

2.1.2 Scenario 2: Large-Scale Natural Disaster

This scenario simulates a widespread natural disaster, such as a flood or earthquake, testing the system's ability to handle a complex, large-scale situation with advanced but limited drone resources.

Parameters:

- Map size: 20x14
- Drones: 2 (high visibility: $n_v = 3$)
- First Responders: 5 (high rescue time: $t_{fr} = 8$)
- Civilians: 10 (high vulnerability: $t_v = 15$-25, high zero-responder rescue time: $t_{zr} = 6$)

Rationale:

- The expansive map size (20x14) represents a large affected area, typical of a natural disaster zone. This vast space challenges the system to efficiently allocate resources across a wide area.
- Two drones with high visibility ($n_v = 3$) simulate the deployment of advanced, long-range reconnaissance drones equipped with sophisticated sensors and imaging technology. This high visibility allows for more effective area coverage despite the reduced number of drones, testing the system's ability to maximize the utility of limited but highly capable aerial assets. The scenario explores how well the system can leverage superior information gathering to compensate for fewer aerial units.
- The increased number of first responders (5) with uniformly high rescue times ($t_{fr} = 8$) reflects the difficulties of navigating a disaster-stricken area. The consistent high rescue time represents the challenges posed by debris, damaged infrastructure, or flood waters that hinder rapid movement.
- Ten civilians with high vulnerabilities ($t_v = 15$-25) and uniformly high zero-responder rescue times ($t_{zr} = 6$) simulate a population caught unprepared by a sudden, widespread disaster. The high vulnerability scores indicate immediate danger from environmental hazards, while the high zero-responder times suggest that self-rescue or civilian-to-civilian assistance is difficult due to the scale and nature of the disaster.

This scenario evaluates the system's capacity to manage multiple rescue operations simultaneously across a large area, prioritizing limited resources effectively in a situation where time is critical.

2.1.3 Scenario 3: Industrial Complex Emergency

This scenario simulates an emergency in a medium-sized industrial complex where a lot of drones with low visibility are available.

Parameters:

- Map size: 18x12
- Drones: 4 (low visibility: $n_v = 1$)
- First Responders: 5 (high rescue time: $t_{fr} = 8$)
- Civilians: 10 (varied vulnerability: $t_v = 15$-25, high zero-responder rescue time: $t_{zr} = 6$)

Rationale:

- The medium map size (18x12) represents an industrial complex with multiple buildings and open areas. This layout challenges the system to navigate both enclosed spaces and exposed regions, simulating the varied terrain of an industrial site.

- Four drones with low visibility ($n_v = 1$) reflect the challenges of operating in an environment with potential chemical hazards, smoke, or electromagnetic interference from industrial equipment. This tests the system's ability to make decisions with limited aerial visibility.

- Five first responders with high rescue times ($t_{fr} = 8$) simulate the challenges of operating in a hazardous industrial environment. The high rescue time accounts for the need to use specialized equipment, navigate complex industrial structures, and potentially deal with hazardous materials.

- Ten civilians with varied vulnerabilities ($t_v = 15$-25) represent a mix of workers in different parts of the complex, from those in immediate danger near the source of the emergency to those in less affected areas. The high zero-responder rescue times ($t_{zr} = 6$) indicate that self-rescue or peer assistance is difficult due to the hazardous nature of the emergency, requiring professional intervention in most cases.

This scenario evaluates the system's ability to manage rescue operations in a complex, hazardous environment where both the layout and the nature of the emergency pose significant challenges to traditional rescue methods.

2.2 Stochastic Scenarios

2.3 Verification Results

3 Conclusions