POLITECNICO MILANO 1863

Formal Analysis of Search-and-Rescue Scenarios

Formal Methods for Concurrent and Real-Time Systems
a.y. 23/24

Prof. Pierluigi San Pietro
Dr. Livia Lestingi

1 Introduction

Rescue services increasingly exploit autonomous systems as a support tool during mass emergencies. For example, fleets of drones can be deployed to monitor an enclosed area and interact with human subjects to foster cooperation and ease the evacuation process.

This project adopts a framework that is currently under research, envisaging civilian survivors (i.e., referred to as *zero-responders*) helping those in need or asking the *first-responders* (i.e., professionals such as nurses and firefighters) to intervene. Specifically, when a drone surveying the area detects a potential zero-responder in the vicinity of a person in need, it must **decide** whether to instruct them to help directly or ask a first-responder to intervene [3].

As a highly safety-critical setting, ensuring the drones' decision-making policies guarantee certain properties is fundamental to maximizing life-savings. Therefore, you are in charge of developing a formal model of search-and-rescue scenarios featuring drones, civilians, and first-responders. The work has two main outputs: an Automata-based model of the scenario and a formal verification experimental campaign highlighting relevant features of the system.

The document is structured as follows:

Section 2 explains the mandatory entities to be modeled, how they are supposed to synchronize and their main characteristics.

Section 3 explains the mandatory properties to be verified.

Section 4 provides instructions on how to deliver your project.

2 Model

The modeling approach for the entities for the search-and-rescue scenario described in the following is inspired by the dynamics of Cellular Automata[1] [4].

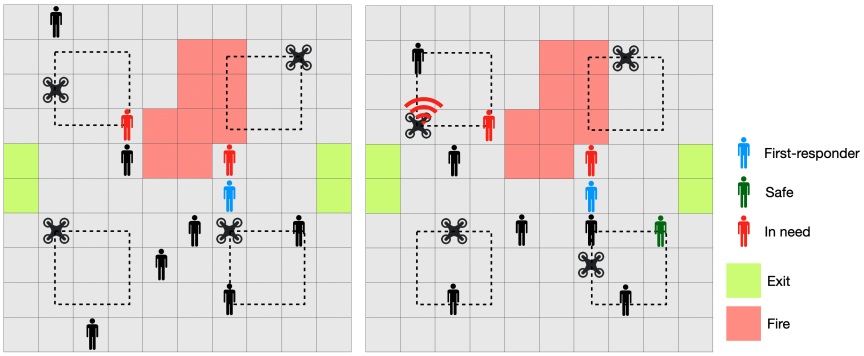
Layout

Assume that the layout (i.e., the geometry of the area surveyed by the drones) is discretized into an $m \times n$ grid such that $m, n \in \mathbb{N}$ and (potentially) $m \neq n$ can hold. The values of m and n must be *configurable* (i.e., the model cannot be tied to a specific layout geometry). A 10×10 example is shown in Figure 1.

Emergency exits are placed on the boundary of the layout. Positions of boundary cells corresponding to exits must be configurable. Fires are also present on the scene. Fires can occupy blocks of adjacent cells, and their position must also be configurable. Assume that their position does not change throughout the scenario; that is, they are not put out, and no new fire is generated mid-way through a scenario.

Agents (drones, civilians, and first-responders) occupy one cell at a time (each cell is occupied by only one human agent simultaneously, while drones can hover over cells occupied by a human). Agents' positions refresh each time unit (assume that all agents move by only 1 cell/time unit) with rules described in their corresponding sections.

[1] Conway's Game of Life is a typical example of cellular automaton.



(a) (b)

Figure 1: Snapshot of example layout in two successive time units with four drones and one first-responder deployed on the scene.

Drones

Drones survey the area and make a decision when they detect a situation requiring intervention.

You can assume that drones patrol with a pre-determined movement pattern that you design (e.g., each drone circles around a fixed area as in Figure 1).

The drone is programmed to communicate with civilians when it detects both a survivor and a person in need of assistance within $\mathsf{N_v}$ tiles (i.e., $\text{dist}(\text{drone}, \text{survivor}) \leq \mathsf{N_v}$ and $\text{dist}(\text{drone}, \text{victim}) \leq \mathsf{N_v}$ hold), where $\mathsf{N_v}$ is a configurable parameter (different drones may have different $\mathsf{N_v}$ values).

Stochastic Version: The drones' vision sensors are not faultless. When in the proximity of a survivor and a victim, the drone detects it with probability $1 - \mathsf{p_{fail}}$. Parameter $\mathsf{p_{fail}}$ is configurable (different drones may have different $\mathsf{p_{fail}}$ values).

You design the decision-making policy. The simplest option is a purely non-deterministic choice between instructing to help and contacting the closest first-responder. Potential refinements take into account the distance between the survivor and the victim and the survivor and the first-responder.

!! What happens if two drones detect the same survivor-victim pair (especially if they send conflicting instructions)?

First-Responders

First-responders move (and change their state) with the following rules:

1. If a person needing assistance is within a 1-cell range, the first-responder will assist them for $\mathsf{T_{fr}}$ time units. After such time elapses, the assisted civilian is automatically considered safe (thus, they effectively leave the scene).

2. Otherwise, the first-responder can move. You design the moving policy. The simplest option is a purely non-deterministic choice between adjacent cells.

The ordering matches the rule's priority.

Civilians

Civilians move (and change their state) with the following rules:

1. If an exit is within a 1-cell range, they are considered safe and, effectively, leave the scene (thus, they no longer move and are not involved in the drones' decisions).
2. If a tile occupied by a fire is within a 1-cell range, they are considered in need of assistance (either by a zero-responder or a first-responder). They are considered a casualty if not brought to a safe state within $\mathsf{T_v}$ time units.
3. If the survivor has been instructed to help, they are busy acting as a zero-responder for dist(survivor, victim) + $\mathsf{T_{zr}}$ time units (i.e., the time it takes for the zero-responder to reach the victim plus a configurable overhead to perform the assistance). Different subjects may have different $\mathsf{T_{zr}}$ values. After such time elapses, both the zero-responder and the victim are automatically considered safe (you do not need to model them reaching the exit explicitly).
4. If the survivor has been instructed to contact the first-responder, they are busy enacting this instruction for dist(survivor, first-responder) + dist(first-responder, victim) + $\mathsf{T_{fr}}$ time units (i.e., the time it takes for the zero-responder to reach the first-responder, then the victim, plus a configurable overhead for the first-responder to intervene). After such time elapses, both the survivor and the victim are automatically considered safe (you do not need to model them reaching the exit explicitly).

   !! If the first-responder is already busy assisting another civilian, this rescue action must be completed before they can begin a new one.

   Stochastic Version: Civilians are not professionals trained to tackle emergencies. When instructed by the drone, they acknowledge the instruction and enact with probability $\mathsf{p_{listen}}$ and ignore it (or miss it) with probability $1 - \mathsf{p_{listen}}$. Different subjects may have different $\mathsf{p_{listen}}$ values.
5. If none of the above apply, the survivor can move. You design the moving policy. The simplest option is a purely non-deterministic choice between adjacent cells. Potential refinements may take into account the direction towards the closest exit.

The ordering matches the rule's priority (i.e., if a person is adjacent to an exit, they are considered safe even if a fire is equally distant).

3 Properties

The project will be carried out using the Uppaal[2] tool [1]. The verification experimental campaign must be carried out in two phases.

[2]Uppaal.org

3.1 Phase 1: Exhaustive Model Checking

You are required to formally model the search-and-rescue scenario as in Section 2 without the stochastic features. The system must be modeled as a Network of **Timed Automata** (NTA) through the Uppaal GUI. Unless explicitly stated, you decide what to model as a TA feature and what via a variable for each described entity.

Given the features presented in Section 2, the scenario evolves differently based on:

- the geometry (i.e., where exits and fires are located);
- the number of agents in the scene (i.e., how many drones, civilians, and first-responders are present);
- the drones' visibility range N_v;
- time-related parameters T_{fr}, T_{zr}, and T_v;
- the drones' decision-making policy;
- the drones' and humans' moving policy.

You are required to formally express in **TCTL** logic and check the following properties, given a scenario configuration:

1. it is possible for a percentage $N_{\%}$ of all civilians to reach a safe state within time T_{scs};
2. a percentage $N_{\%}$ of all civilians is always guaranteed to reach a safe state within time T_{scs}.

Parameters $N_{\%}$ and T_{scs} must be tuned based on the specific configuration (e.g., find the maximum $N_{\%}$ and minimum T_{scs} such that properties 1 and 2 are satisfied). Verifying additional properties is welcome (indeed, encouraged) but not mandatory.

Analyze and report on **at least three** scenario configurations that highlight different behavioral aspects. Configurations must not be trivial (e.g., only one survivor or a 2x2 layout).

3.2 Phase 2: Statistical Model Checking

You are required to extend the model used for Phase 1 with the stochastic features [2]. Formally express using Uppaal's `Pr` operator the **probability** of Phase 1 properties holding within specific time bounds. The time-bound used for verification must also be properly tuned to avoid trivial results (e.g., 0% deadlock probability within 1 time unit). Verifying additional properties (e.g., the estimated loop duration for a single piece to support the time-bound tuning task) is welcome also for Phase 2.

Given the stochastic features presented in Section 2, the scenario evolves differently based on:

- the drones' sensor failure rates;

- the humans' instruction acknowledgment rates.

Analyze and report on **at least three** non-trivial scenario configurations (these can be stochastic versions of one or more configurations analyzed during Phase 1), highlighting relevant behavioral aspects.

4 Delivery Instructions

It is **mandatory** to deliver:

- a .pdf report (max 10 pages excluding front cover and bibliography, no constraint on the template) describing:
  - the model (emphasis on **critical** modeling choices you have made);
  - the system configurations you have chosen;
  - verification results;
- the .xml Uppaal model. If you implement the stochastic features, the standard non-stochastic version of the model must still be evaluated (you can deliver two .xml files, one for the standard version and one for the stochastic one). Make sure:
  - it is the same as what you present in the report
  - it is test-ready (i.e., it includes the queries you ran for the experiments and system configurations are easily selectable).

Delivery deadline: TBA (approx. mid July).

To submit your work, send an email to livia.lestingi@polimi.it with the report, Uppaal files, and potential supplementary material to be evaluated.

Bibliography

[1] Behrmann, G., David, A., Larsen, K.G.: A tutorial on uppaal. In: Formal Methods for the Design of Real-Time Systems. Lecture Notes in Computer Science, vol. 3185, pp. 200–236. Springer (2004), https://doi.org/10.1007/978-3-540-30080-9_7

[2] David, A., Larsen, K.G., Legay, A., Mikucionis, M., Poulsen, D.B.: Uppaal SMC tutorial. Int. J. Softw. Tools Technol. Transf. **17**(4), 397–415 (2015), https://doi.org/10.1007/s10009-014-0361-y

[3] Gavidia-Calderon, C., Kordoni, A., Bennaceur, A., Levine, M., Nuseibeh, B.: The idea of us: An identity-aware architecture for autonomous systems. ACM Transactions on Software Engineering and Methodology (2024)

[4] Wolfram, S.: Cellular automata as models of complexity. Nature **311**(5985), 419–424 (1984)