POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Formal Analysis of Search-and-Rescue Scenarios

Project for Formal Methods for Concurrent and Realtime Systems course

Authors: **Alberto Nidasio, Federico Mandelli, Niccolò Betto**

Advisor: Prof. Pierluigi San Pietro
Co-advisors: Dr. Livia Lestingi
Academic Year: 2023-24

Abstract

This document presents a formal model implemented with Uppaal of search-and-rescue scenarios. Inside a rectangular map of arbitrary size, civilians have to be brought to safety by either reaching an exit or being assisted by first-responders. Drones surveys the area and coordinate the rescue efforts by instructing civilians on what to do. The model then undergoes formal verification to highlight key behavioral aspects and identify optimal configurations for maximizing civilian safety.

1 High Level Model Description

The model adopted for the search-and-rescue mission involves 3 different types of agents: Civilians, First-responders and Drones. They are placed in different numbers inside a rectangular map, where exits (i.e. safe zones reached by civilians to get to safety) and fires are fixed in placed from the beginning of the scenario.

The key characteristics of the agents are these:

- **Civilians**: Can be in 3 different states, depending whether they find themselves near a fire or if they are following instructions
  - **In-need** (i.e. near a fire): They cannot move and needs to be assisted. After T_v time units near a fire, they became a casualty
  - **Busy**: The civilian is following an instructions and can be either assisting directly or contacting a *first-responder* to get help
  - **Moving**: When civilians are not near a fire or busy enacting some instruction, they can move towards an exit to get to safety following some *moving policy*
- **First-responders**:
  - **Assisting**: When a civilian *in-need* is within a 1-cell range, the *first-responder* will assist them for T_{fr} time units. After that, the assisted civilian is considered safe
  - **Moving**: When free from other tasks, the *first-responder* can move following some *moving policy*
- **Drones**: They survey their surroundings, limited by the field of view N_v of the sensors, and following a pre-determined path moving 1 cell at each time step. When two civilians, one *in_need* and one free, are detected the drone can instruct the free civilian to assist the *in_need* directly or to contact a *first-responder*

1.1 Model Assumptions

To simplify the model described in the assignment, the following assumptions have been made:

- The map is a 2D grid with a fixed number of rows and columns, and fires and exits are static (i.e. they won't change during simulation)
- Movements from one cell to another allows for diagonal movements. Therefore, the distance can be esily computed as the maximum between the difference of the x and y coordinates
- Movement of *civilians* and *first-responders* towards a human target (e.g. a *survivor* goes to a *first-responder*), are modeled with a wait state, where the agents remains idle for the duration of the movement, and with a change in coordinates. This is done to reduce the complexity of the model
- Drones know the global position of all the *first-responders* and their status, at any given time. This allows them to instruct *civilians* to contact the nearest *first-responder*
- All *civilians* know the location of the exits and can determine the nearest one
- *survivors* cannot start the simulation inside a fire cell

2 Model Description and Design Choices

2.1 State and Parameters Representation

Each agent type (*civilian*, *first-responder*, *drone*) is represented by an automaton, called **template** in Uppaal. These templates are characterized by many different parameters, and are implemented in Uppaal in the following way:

- The template signature (the parameters list) contains only one constant parameter, the agent id, annotated with a custom type defined as an integer with the range of possible ids (e.g. `typedef int[0, N_DRONES-1] drone_t;`). This way, by listing the template names in the *System declaration* (`system Drone, Survivor, FirstResponder;`), Uppaal can automatically generate the right number of instances of each template;
- The agents' parameters are defined in constant global arrays. Each template instance can then access its own parameters by using the agent id as an index for these arrays (e.g. `const pos_t drones_starting_pos[drone_t] = {{1, 3}, {6, 3}};`).

This setup allows to easily define the simulation parameters all inside the *Declaration* section, thus without modifying neither the templates nor the *System declaration* section, and to easily assign different parameters to each agent instance.

2.1.1 Map Representation

Desipite each agent holding internally its own position, a global representation of the map is needed for agents who require to know the state of other agents (e.g. drones need to know the position of *first-responders* to instruct *civilians* to contact them).

The map is represented as a 2D grid of cells, with each cell indicating which type of human agent is within (drones positions are not neeed, so they are not included in this representation). This choice is made to avoid each agent holding a reference to all other agents, which would make the model more complex and harder to maintain.

```
// Map cell status enumeration
const int CELL_EMPTY =      0;
const int CELL_FIRE =       1;
const int CELL_EXIT =       2;
const int CELL_FIRST_RESP = 3;
const int CELL_SURVIVOR =   4;
const int CELL_ZERO_RESP =  5;
const int CELL_IN_NEED =    6;
const int CELL_ASSISTED =   7;
const int CELL_ASSISTING =  8;

typedef int[0, 8] cell_t;

// Map array
cell_t map[N_COLS][N_ROWS];
```

When one agent changes position, it updates the map accordingly. For example, when a *civilian* moves, it empties the cell it was occupying and fills the new cell with its type.

```
  void move(int i, int j) {
    set_map(pos, CELL_EMPTY);
    pos.x += i;
    pos.y += j;
    set_map(pos, CELL_SURVIVOR);
}
```

2.2 Civilian

The civilian agent is the actor in danger that needs to get to safety.

At the beginning of the simulation the civilians position themselves in the map; if they are near a fire they become *in_need* otherwise they are considered *survivors*.

1. *in_need*: The civilian cannot move and needs to be assisted. After T_v time units near a fire, they became a casualty, if assisted in time they are considered safe. In both cases they leave the simulation freeing the map cell they were occupying.
2. *survivors*: The civilian moves towards an exit following a *moving policy*. If they are within a 1-cell range from an exit they are considered safe and leave the simulation freeing the map cell they were occupying. In this state they can receive instructions from the drone to either assist a *in_need* civilian or to contact a *first-responder* to get help.
   - Assisting a *in_need* civilian: The civilian "moves" towards the *in_need* civilian by staying in the same cell for the time needed to reach the *in_need* (the distance between the two). The Civilian "assist" the *in_need* by waiting T_{zr}. After that the civilian is considered safe and leaves the simulation.
   - Calling a First responder: The civilian "moves" towards the *first-responder* by staying in the same cell for the time needed to reach the *first-responder* (the distance between the two). The Civilian "calls" the *first-responder* that will assist the *in_need* civilian. When the *first-responder* ends the assist the civilian is considered safe and leaves the simulation.

At the end of the simulation all civilians are either safe or casualties.

2.3 First-responder

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut.

2.4 Drone

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut.

2.5 Initializer

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut.

2.6 Design Choices

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et.

3 Properties

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut.

4 Conclusion

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut.