Background

We begin by explaining the main concepts and tools that were used in this project. Starting with an introduction of theory of automata and timed automata, as every system analyzed in this paper was treated as a timed automaton. We later jump to the explanation of the tool (UPPAAL) that allows us to represent timed automata as models. Where only the relevant features of UPPAAL that were used in this project’s implementations are discussed. Then we introduce a general explanation of an already existent program that was utilized in order to interact with UPPAAL.

Automaton

Generally speaking, an automaton is a mechanism or machine that automatically executes a sequence of operations. Its representation can be explained as a 5-tuple <Q, Σ, δ,q0,F>where:

* Q is a finite set of states.
* Σ is a finite set of symbols that are part of the alphabet of the automaton.
* δ is the transition function, also representable as: Q x Σ 🡪 Q which can be explained as a 3-tuple (p,o,q) where p is a state that carries the label o, containing Σ and goes to the state q.
* q0 is the initial state.
* F is a set of states of Q.

Timed automaton

A timed automaton is actually an automaton extended with clock variables, that represents an abstract model of a real time system. Based on our previous 5-tuple representation, we need to introduce a finite set of real-valued variables which will be referred as clock variables. Such clock variables are going to be the ones that restraint the behavior of our automaton.

The formal representation of a timed automaton can be done as a 4-tuple <Q, q0, Σ, I> where:

* Q is a finite set of states.
* q0 is the initial state.
* , Σ is a set of edges, also representable as: Σ which can be explained as a 3-tuple (p,o,q) where p is a state that carries the label o, containing Σ and goes to the state q.
* F is a set of states of Q.

UPPAAL

UPPAAL is a modelling tool that is used in this to models and to also interact with the properties of them. It allows to retrieve traces from a model, in order to analyze its behavior. Due to the extended features that this tool offers, we will only discuss the ones that are relevant for this project on the following sections.

UPPAAL can model a system by nesting a set of timed automata, where each automaton represents a state of a system.

Template

Structure that defines an automaton and also allows to declare the parameters or variables of it. The main variable types that are going to be used on this project are:

* Int
* Double
* Clock

Location

In UPPAAL, each state of a system is called a location. Where this last one has the following properties:

* Name
  + The name is used as a unique identifier for each location
* Invariant
  + A Boolean expression that compares the value of a clock variable against an integer. This expression is used as a condition that determines the active time of a location.
* Type
  + Initial
    - Feature of a location that indicates the starting state of a system.
  + Urgent
    - Feature that does not let time pass by on a location
  + Committed
    - Feature that also does not let time pass by on a location, but also restricts the transition of the location to only outgoing edges of the current locations.

Edges

Transitions that allows a location to travel to other locations or to itself. Each transition can be combined with a set of actions or conditions like the followings:

* Guard
  + Boolean expression that has to be satisfied in order to successfully execute the desired transition
* Update
  + Expressions, mostly assignments that can modify or update the values of the declared variables of a template.

Branch Points

A branch point is a structure that is able to connect a location with other locations based on a probability value. The outgoing edges of branch points, which we will call for convenience: “probability edges” have a specific weight value that determines the probability of success of a transition to be executed among all probability edges emerging from the same branch point.

Weighted edges

Query language

UPPAAL has a language that allows a user to interact with the created models and simulate a set of traces that represent the behavior or output of a model given the following parameters of a given command or query:

simulate x [*expression*] {*variables*}

where x is the number of times that the simulation will be executed, expression a Boolean expression that will determine the time of execution of the simulation and variables the set of variables of the system to be observed on the system.

The result of a query with the command simulate outputs x set of traces for each desired variable values from the beginning of the simulation until the specified time of execution.

Trace

simulate 10 [<=300] {Process.x}

Model Generator

Explain from where we’re obtaining the data. (Sascha’s tool) existent tool.