A formal framework to specify and test systems with fuzzy-time information*

Juan Boubeta-Puig[1], Azahara Camacho[2], Luis Llana[2], Manuel Núñez[2]

[1] Department of Computer Science and Engineering, School of Engineering
University of Cadiz, Spain
Email: juan.boubeta@uca.es
[2] Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid, Spain
Email: mariaazc@ucm.es,llana@ucm.es,manuelnu@ucm.es

Abstract. We specify the behavior of a sensor network with different sensor stations distributed all along the region of Andalusia (South of Spain). The main goal of this network is the measure of air quality taking into account the maximum levels of certain pollutants. The problem that we try to solve with this formalization is the management of time inaccuracies between the components of a network with the aim of avoiding the malfunctions that are derived from them. We present the formal syntax and semantics of our variant of fuzzy-timed automata and define all the automata corresponding to the different parts of the netowrk.

1 Introduction

The use of formal methods for the specification and analysis of network systems and of the protocols running over them is not new [3,4]. However, formal methods are not widely used to guide the development of computer systems. This is an anomalous situation in engineering, where the simplest project includes a detailed design of the artifact to be built. Therefore, formal methods should use blueprints to guide the *construction* of computer systems [8], in particular, to formally implement the validation process via testing [7,5].

In this paper we present a formal design of a non-trivial system: the measure of air quality through a sensors network, with sensor stations distributed all along Andalusia (a region of Spain). First, we had to find a formalism to represent the network. The problem with well-established formal methods is that they are oriented to systems where uncertainty usually does not play a relevant

* The research presented in this paper has been partially supported by the Spanish MINECO/FEDER project DArDOS (TIN2015-65845-C3-1-R and TIN2015-65845-C3-3-R), the Comunidad de Madrid project SICOMORo-CM (S2013/ICE-3006) and the University of Cádiz project (PR2016-032). The second author is supported by a Universidad Complutense de Madrid - Santander Universidades grant. The first author would like to thank the hospitality of the Design and testing of reliable systems research group, at Universidad Complutense de Madrid, during his stay when this research was carried out.

role. In fact, they are good to cope with time information, parallelism and non-determinism, features required in our study, but approximate reasoning is out of the scope of these languages. Specifically, they are not good to deal with *fuzzy time*. Suppose that a certain signal should be received every 10 minutes and there is a lapse of 10 minutes and 0.001 seconds since the reception of the last message then we should not raise an alarm because this *error* might be due to several reasons, none of them showing a failure of the device sending the signal. For example, the clock of the receiver might be wrong or the message might have been delayed while being retransmitted in a point between the sender and the receiver. We decided to *adapt* an existing formalism, timed automata [1], to consider fuzzy time. Although there are many proposals to define fuzzy automata [2,6,11,12], we considered timed automata because it is much easier to reuse its existing tools to deal with fuzzy time, most notably UPPAAL [9], than build tools from scratch. Technically, our proposal is a hybrid between timed automata [1] and our proposal of fuzzy automata [2].

We introduce a formalism to formally design complex systems where inaccuracies in the measure of time are managed via fuzzy logic. In addition, we introduce an implementation relation so that we can match the expected behavior (the one given by the specification) and the one of the studied system. The idea is to analyze mismatches in order to detect potential malfunctioning. In order to show the usefulness of our formalism, we will formally specify a sensors network formed by a central server connected to 61 sensor stations, each of them conformed by 6 sensors, used to capture the amount of different pollutants in the air of Andalusia.

The rest of the paper is structured as follows. In Section 2 we present some basic concepts of fuzzy logic and introduce our fuzzy adaption of timed automata. In Section 3 we present our case study and formally specify the sensors network. Finally, in Section 4 we present our conclusions.

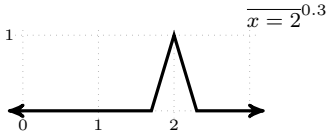
2 Introducing fuzzy-timed automata

In this section we introduce some basic concepts of fuzzy logic that will be used afterwards to define our *fuzzy* version of timed automata.

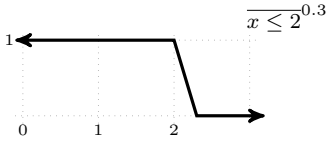
In ordinary logic, a set or a relation is determined by its characteristic function: a function that returns true if the element is in the set (or if some elements are related) and false otherwise. In the fuzzy framework we have a complete range of values in the interval $[0, 1]$; the larger is the value, the more confidence we have in the assessment. We consider relations over the set of real numbers $\mathbb{R}$. Therefore, a fuzzy relation is a mapping from the Cartesian product $\mathbb{R}^n$ into the interval $[0, 1]$

In this paper, we consider the fuzzy relations given in Figure 1. These relations depend on a non negative real number $\lambda \geq 0$. We use these functions to define fuzzy-timed automata. The only property we use, which simplifies the writing, is the commutativity of equality $\overline{x = y}^{\lambda} = \overline{y = x}^{\lambda}$, viewed as a binary operation over $\mathbb{R}$.

$$\overline{x = y}^{\lambda} = \begin{cases} 0 & \text{if } x \leq y - \lambda \text{ or } x > y + \lambda \\ 1 + \frac{x-y}{\lambda} & \text{if } y - \lambda < x \leq y \\ 1 - \frac{x-y}{\lambda} & \text{if } y < x \leq y + \lambda \end{cases}$$



$$\overline{x \leq y}^{\lambda} = \begin{cases} 1 & \text{if } x < y \\ 0 & \text{if } x > y + \lambda \\ 1 - \frac{x-y}{\lambda} & \text{if } y \leq x \leq y + \lambda \end{cases}$$



$$\overline{x \geq y}^{\lambda} = \begin{cases} 1 & \text{if } x > y \\ 0 & \text{if } x \leq y - \lambda \\ 1 + \frac{x-y}{\lambda} & \text{if } y - \lambda < x \leq y \end{cases}$$

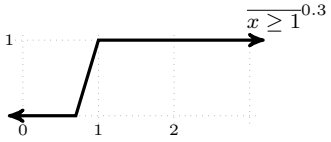


Fig. 1. Fuzzy Order Relations

A triangular norm (abbreviated t-norm) is a binary operation used in fuzzy logic to generalize the *conjunction* in propositional logic. Therefore, we require a t-norm to satisfy similar properties. We also require an extra property: monotonicity. Intuitively, the resulting truth value does not decrease if the truth values of the arguments increase.

Definition 1. *A* t-norm *is a function* $T : [0,1] \times [0,1] \mapsto [0,1]$ *which satisfies the following properties:*

- *Commutativity:* $T(x,y) = T(y,x)$.
- *Monotonicity:* $T(x,y) \leq T(z,u)$ *if* $x \leq z$ *and* $y \leq u$.
- *Associativity:* $T(x,T(y,z)) = T(T(x,y),z)$.
- *Number 1 is the identity element:* $T(x,1) = x$.
- *Number 0 is nilpotent:* $T(x,0) = 0$.

Since t-norms are associative, we can generalize them to take as parameter a list of values:

$$T(x_1, x_2, \ldots, x_{n-1}, x_n) = T(x_1, T(x_2, \ldots, T(x_{n-1}, x_n) \ldots))$$

The following t-norms are often used:

Łukasiewicz t-norm:
$T(x,y) = \max(0, x + y - 1)$. We represent this t-norm with the symbol $\curlywedge$.

Gödel t-norm:
$T(x,y) = \min(x,y)$. We represent this t-norm with the symbol $\overline{\wedge}$.

Product t-norm:
$T(x,y) = x \cdot y$ (real number multiplication). We represent this t-norm with the symbol $\bigstar$.

Hamacher product t-norm:

$T(\delta_1, \delta_2) = \dfrac{\delta_1 \cdot \delta_2}{\delta_1 + \delta_2 - \delta_1 \cdot \delta_2}$. We represent this t-norm with the symbol $\divideontimes$.

Next, we define *fuzzy-timed* automata. In classical timed automata theory [1], time is expressed in the time constraints. Hence, we need to modify these constraints in order to be able to introduce fuzziness. In ordinary timed automata theory, the time constraints consist of conjunctions of inequalities. Instead of ordinary crisp inequalities we use their fuzzy counterparts appearing in Figure 1. We could have more freedom in allowing general convex fuzzy sets, but we have preferred to keep our constraints close to the original ones so we can use the theory developed for timed automata. The role of a conjunction in Fuzzy Theory is played by t-norms. There is not a *canonical* t-norm: we have presented 4 of the more used t-norms and the designer can specify which one is more appropriate in each situation.

In order to define fuzzy-timed automata we need some additional ingredients. First, we need a set of *variables* denoted by Vars. We denote the elements of Vars by $x, y, z, \ldots$. These variables will take values in real numbers. We assume that there is a special variable ts that represents a clock. This variable can only take non negative values. An *environment* is a function $e : \mathsf{Vars} \mapsto \mathbb{R}$ that maps variables and clocks into real numbers. We denote the set of environments by Env. We extend the environments to sequences of variables in the natural way: $e(x_1, \ldots, x_n) = (e(x_1), \ldots, e(x_n))$. The modification of a variable is denoted by $e[\overline{v}/\overline{x}]$, meaning that the variables in $\overline{x}$ take the values indicated in $\overline{v}$. We consider a set of *channel labels*, denoted by A. From this set we define the set of *output channels*: an output channel is defined by $a!(id, ID, \overline{v})$, where a is a channel label, id is the identifier of the sender of the message, ID is the set of identifiers of the receivers, and $\overline{v}$ is the tuple of sent data. Formally, $A!$ is the set

$$\{a!(id, ID, \overline{v}) \mid a \in A, id \in \mathbb{N}, ID \subseteq \mathbb{N},\ \overline{v} \in \bigcup_{i=1}^{|\mathsf{Vars}|} \mathbb{R}^i\}$$

We will also consider a set of *input channels* $a?(id, \overline{x})$ where $a \in A$ and id is the identifier of the component receiving the value. Formally, $A?$ is the set

$$\{a?(id, \overline{x}) \mid a \in A, id \in \mathbb{N},\ \ \overline{x} \in \bigcup_{i=1}^{|\mathsf{Vars}|} \mathsf{Vars}^i\}$$

Finally, we consider the set of actions $\mathsf{Acts} = A \cup A! \cup A?$. Next we define the formal syntax of our fuzzy constraints using Backus-Naur Form (BNF).

Definition 2. *A* fuzzy constraint *is a formula built from the following BNF:*

$$C ::= \mathsf{True} \mid C_1 \triangle C_2 \mid \overline{x \bowtie n}^{\lambda} \mid \overline{x_1 - x_2 \bowtie n}^{\lambda}$$

where $\triangle$ is a t-norm, $\bowtie \in \{\leq, =, \geq\}$, $x, x_1, x_2 \in \mathsf{Vars}$, $\lambda \in \mathbb{R}^+$, and $n \in \mathbb{N}$. We denote the set of fuzzy constraints by $\mathcal{FC}$.

In timed automata theory, constraints are used to decide if the automata can stay in a location and to decide if a transition can be executed. All this is done by checking if a valuation satisfies the corresponding constraint. In fuzzy theory the notion of satisfaction is not crisp: we do not have a boolean answer but a value in the interval $[0, 1]$. Therefore, we do not have that a constraint is true or false but a *satisfaction grade* of a constraint.

Definition 3. *Let $e \in \mathsf{Env}$ be an environment and $C \in \mathcal{FC}$ be a fuzzy constraint. We inductively define the* satisfaction grade of C in e, *written $\mu_C(e)$, as*

$$\begin{cases} 1 & \text{if } C = \mathsf{True} \\ \overline{e(x) \bowtie n}^{\lambda} & \text{if } C = \overline{x \bowtie n}^{\lambda},\ \bowtie \in \{\leq, =, \geq\} \\ \overline{e(x_1) - e(x_2) \bowtie n}^{\lambda} & \text{if } C = \overline{x_1 - x_2 \bowtie n}^{\lambda},\ \bowtie \in \{\leq, =, \geq\} \\ \triangle\,(\mu_{C_1}(e), \mu_{C_2}(e)) & \text{if } C = C_1 \triangle C_2 \end{cases}$$

Let us remark that $\mu_C(e) \in [0, 1]$.

Definition 4. *A* fuzzy-timed automaton *is a tuple (L, l_0, n, E, I) where:*

- *L is a finite set of locations.*
- *$l_0 \in L$ is the initial location.*
- *$n \in \mathbb{N}$ is the identifier of the automata.*
- *$E \subseteq L \times \mathsf{Acts} \times \mathcal{FC} \times L$ is the set of edges; we write $l \xrightarrow{a,C} l'$ whenever $(l, a, C, l') \in E$.*
- *$I : L \mapsto \mathcal{FC}$ is a function that assigns invariants to locations.*

Next we are going to define the operational semantics of fuzzy-timed automata. We need it to obtain the fuzzy traces that are used for the conformance relations. This operational semantics is given in terms of transitions, which are enabled when time constraints hold. Since we do not have crisp time constraints, transitions must be decorated with a real number $\alpha \in [0, 1]$. This number indicates its certainty. In order to define the operational semantics we need a t-norm. Let us explain the reason. In the definition of the operational semantics of an ordinary timed automaton, action transitions require the condition "$e \vDash C$ and $e[r] \vDash I(l')$". This conjunction must be transformed into its fuzzy version: a t-norm. Our operational semantics is a labeled transition system. The states are triples (l, e), where l is a location of the automaton and e is the environment. This operational semantics has two kind of transitions: timed transitions and action transitions. The former represents the passing of time whereas the later represents the execution of an action. The actions are basically those of the set Acts with one exception: when an input action is performed we need to know the data transmitted in order to update the environment. So we consider the set:

$$\mathsf{Acts_v} = A \cup A! \cup \{a?(id, \overline{v}) \mid a \in A,\ \overline{v} \in \bigcup_{i=1}^{|\mathsf{Vars}|} \mathbb{R}^i\}$$

We also consider the initial environment e_{None} where all variables are set to 0.

Definition 5. *Let $fA = (L, l_0, id, E, I)$ be a fuzzy-timed automaton and $\triangle$ be a t-norm. The* $\triangle$-operational semantics *of fA is the probabilistic labeled transition system whose set of states is $L \times \mathsf{Env}$, the initial state is (l_0, e_{None}), and the set of transitions is given by the following two rules:*

1. $(l, e) \xrightarrow{d}_{\alpha} (l, e[ts + d/ts])$, $\mu_{I(l)}(e[ts + d/ts]) = \alpha$.
2. $(l, e) \xrightarrow{a}_{\alpha} (l', e')$, *being* $\alpha = \triangle (\mu_{I(l')}(e'), \mu_C(e))$, *and such that one of the following conditions hold:*
   (a) $l \xrightarrow{a,C} l'$ *with* $e' = e$.
   (b) $l \xrightarrow{a!(id,ID,\overline{v}),C} l'$ *with* $a = a!(id, ID, \overline{v})$ *and* $e' = e$.
   (c) $l \xrightarrow{a?(id,\overline{x}),C} l'$ *with* $a = a?(id, \overline{v})$ *and* $e' = e[\overline{x}/\overline{v}]$.

We are going to represent systems in which internal clocks are not synchronized. In fact, it is very likely, and realistic, that there will be either delays or advances in the internal clocks. Thereby, in order to define transitions that allow the communication we need to remove timed transitions. So, we define a new kind of *untimed* transition.

Definition 6. *Let $(l, e), (l', e') \in L \times \mathsf{Env}$, $\alpha \in [0, 1]$ and $a \in \mathsf{Acts_v}$. We write $(l, e) \stackrel{a}{\Longrightarrow}_{\alpha} (l', e')$ if there exist two consecutive transitions $(l, e) \xrightarrow{d}_{\alpha_1} (l_1, e_1)$ and $(l_1, e_1) \xrightarrow{a}_{\alpha_2} (l', e')$ such that $\alpha = \alpha_1 \cdot \alpha_2$*

The transitions $(l, e) \stackrel{a}{\Longrightarrow}_{\alpha} (l', e')$ are defined for the labeled transition system that is derived from an automaton A.

Next we are going to consider a network of automata.

Definition 7. *An* automata network *is given by the following BNF.*

$$N ::= (A, l, e) \mid \|_S (N_1, \ldots, N_k)$$

where $k \geq 2$ is a natural number, $S \subseteq A$ is the synchronization alphabet, e is an environment, and l is a location of the fuzzy automaton A.

Given Δ a t-norm, the operational semantics of an automata network is given by the rules in Figure 2.

Once we have defined the operational semantics of an automata network, we can define a conformance relation. In a conformance relation we have a specification, that in our case is an automata network, and an implementation. The implementation is a real system. However, we make a (realistic) assumption to compare specifications and implementations: the latter can produce sequences of actions belonging to the same set of actions as the automata network representing the specification.

Definition 8. *Let N be an automata network, and $\triangle$ be a $t-norm$. We say that $(w, p) \in \mathsf{Acts_v}^* \times [0, 1]$ is a* fuzzy trace *of N according to $\triangle$, denoted by $(w, p) \in \mathsf{ftr}_{\triangle}(N)$, if there exist a sequence of transitions*

$$N = N_0 \stackrel{a_1}{\Longrightarrow}_{p_1} N_1 \ldots \stackrel{a_k}{\Longrightarrow}_{p_n} N_k$$

$$\frac{(l,e) \overset{\beta}{\Longrightarrow}_{\alpha} (l',e')}{(A,l,e) \overset{\beta}{\Longrightarrow}_{\alpha} (A,l',e')}\ \beta \in \mathsf{Acts_v}$$

$$\frac{a \in S,\ N_{i_0} \overset{a!(id_{i_0},ID,v)}{\Longrightarrow}_{\alpha_{i_0}} N'_{i_0}, \forall j \in ID : N_j \overset{a?(j,v)}{\Longrightarrow}_{\alpha_j} N'_j, \forall j \notin ID \cup \{i_0\} : N'_j = N_j}{\|_S(N_1,\ldots,N_k) \overset{a}{\Longrightarrow}_{\triangle\{\alpha_i \mid i \in ID \cup \{i_0\}\}} \|_S N'_1,\ldots,N'_k}$$

$$\frac{N_i \overset{\beta}{\Longrightarrow}_{\alpha} N'_i, \beta \in \{b, b!(id, ID, \overline{v}), b?(id, \overline{v})\}, b \notin S}{\|_S(N_1,\ldots,N_i,\ldots N_k) \overset{\beta}{\Longrightarrow}_{\alpha} \|_S(N_1,\ldots,N'_i,\ldots N_k)}$$

Fig. 2. Automata network semantics

such that $w = a_1 \ldots a_k$ *and* $p = \triangle (p_1, \ldots p_k)$.

Let s *(the specification) and* i *(the implementation) be two automata networks and* $\triangle$ *be a t-norm. We say the* i conforms to s *under the t-norm* $\triangle$*, denoted by* $i\ \mathsf{conf}_{\triangle}\ s$*, if and only if for all trace* $(w,p) \in \mathsf{ftr}_{\triangle}(i)$ *there exists a trace* $(w,p') \in \mathsf{ftr}_{\triangle}(s)$ *such that* $p \geq p'$.

3 Case Study: a sensors network controlling air quality

In this section, we present our case study about a sensors network for controlling air quality. First, we present the main indicatives of air quality used during our study. Next, we give the formal specification, using the automata introduced in the previous section, of all the components of the network. The final goal of our research is to use the specification to derive expected properties of the system and check whether the real system fulfill them.

3.1 Measuring air quality

Our case study is the sensor network implemented by the Andalusian regional government to control air quality across the region. The total area of the region is 87,268 Km^2 (33,694 square miles) and it currently has a population of around 8.4 million people. The network has 61 sensor stations located all around the region. Each station measures the key air pollutants $PM_{2.5}$, PM_{10}, CO, O_3, NO_2 and SO_2 every 10 minutes. In addition, some of them measure other pollutants not relevant for this case study, such as NO and SH_2. In order to validate our proposed system, we use all the real sensing data collected during December 2016.

In order to establish the bounds to determine *good* air quality, we use the index proposed by the U.S. Environmental Protection Agency (EPA) [10] because this agency provides complete information about how every air quality level affects different risk groups. EPA provides the value interval for detecting a

Table 1. AQI categories.

Air Quality Category		**Pollutants**					
Name	Level	NO_2 (ppb) 1 h.	SO_2 (ppb) 1 h.	CO (ppm) 8 h.	O_3 (ppm) 8 h.	$PM_{2.5}$ ($\mu g/m^3$) 24 h.	PM_{10} ($\mu g/m^3$) 24 h.
Good	1	0 - 53	0 - 35	0.0 - 4.4	0.000 - 0.054	0.0 - 12.0	0 - 54
Moderate	2	54 - 100	36 - 75	4.5 - 9.4	0.055 - 0.070	12.1 - 35.4	55 - 154
Unhealthy for Sensitive Groups	3	101 - 360	76 - 185	9.5 - 12.4	0.071 - 0.085	35.5 - 55.4	155 - 254
Unhealthy	4	361 - 649	186 - 304	12.5 - 15.4	0.086 - 0.105	55.5 - 150.4	255 - 354
Very Unhealthy	5	650 - 1249	305 - 604	15.5 - 30.4	0.106 - 0.200	150.5 - 250.4	355 - 424
Hazardous	6	1250 - 2049	605 - 1004	30.5 - 50.4	>0.200	250.5 - 500.4	425 - 604

specific air quality level for every isolated pollutant (see Table 1). As it can been seen in this table, an average of the last 1, 8 or 24 hour-values must be calculated depending on the considered air pollutant. For example, the average value in 1-hour period is required for NO_2, while 8-hour period is needed for CO. Once the average has been calculated, the air quality level can be reported by choosing the range to which the value belongs to. EPA defines the so-called *Air Quality Index* (AQI), a general level for air quality based on the maximum level of each pollutant. There are six possible air quality levels: *Good*, *Moderate*, *Unhealthy for Sensitive Groups*, *Unhealthy*, *Very Unhealthy*, *Hazardous*. The corresponding health risks are given for each air pollutant and level. For example, having a NO_2 1 hour-avg value of 55 ppb (parts per billion) and a CO 8 hour-avg value of 13.7 ppm (parts per million) implies the detection of the following air quality levels: *Moderate* for NO_2 and *Unhealthy* for CO. As the *Unhealthy* level is more dangerous than the *Moderate* one, the air quality level (AQI) for this location will be *Unhealthy*.

3.2 Formal specification

In this section, we present the specification, using our fuzzy-timed automata, of all the components of the network. We have three main entities. First, *sensors* are the small devices collecting information about a particular pollutant. These sensors are associated with *sensor stations*, containing one sensor per pollutant (in our case, 6 different sensors). Our network has 61 sensor stations that communicate with a *server* where the information is processed and alarms, if needed, are produced. Servers and stations also communicate to deal with potential errors, both malfunctioning of a station and wrong adjustments of local clocks. First, we present the variables used in these automata to store and manage the collected data:

- u: variable for saving the daily report of a sensor station.
- v: variable for saving the value obtained by the sensor of one air pollutant.

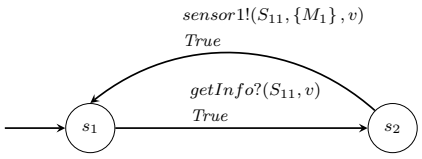


Fig. 3. Sensor S_{11} automaton.

- x: 6-tuple for saving the value of each air pollutant measured in a sensor station.
- y: 3-tuple for saving the information obtained from checking the clocks of the stations.
- z: variable to store the mean of delays of the clock of a sensor station during an hour.
- w: variable to store the *state* of a sensor station; 1 denotes *erroneous* state while 0 denotes *normal* state.
- ts: variable for storing the time value of the clock.
- ts_{aux}: auxiliary variable for storing time that will be helpful to the trigger of some actions.
- ts_{rep}: auxiliary variable for storing the last time when a report was sent to the stations of the network.

Next, we describe each of the three components that we mentioned before. The most basic elements of the system are *sensors*. We have a different type of sensor for each air pollutant that they measure: $PM_{2.5}$, PM_{10}, CO, O_3, NO_2 and SO_2. Since our network has 61 sensor stations, we have a total of 366 sensors. The functionality of sensors includes two actions:

- Data Capture: Register the current level of the pollutant that the sensor measures. This is performed by the action $getInfo$.
- Data Sending: Send to the sensor station the info collected about the amount of pollutants. Depending on the measured pollutant, the action performed is $sensor1$ for $PM_{2.5}$, $sensor2$ for PM_{10}, $sensor3$ for CO, $sensor4$ for O_3, $sensor5$ for NO_2 and $sensor6$ for SO_2.

In Figure 3 we give a graphical representation of the fuzzy-timed automata corresponding to the first sensor (S_{11}) of the first sensor station (M_1). The definition of all the sensors follow the same pattern.

Sensors are collected in group of 6 (one per every pollutant) and they conform *sensor stations*. A total of 61 stations are distributed along the Andalusian territory for measuring air quality. The main functionality of stations is implemented with the following actions:

- Data Capture: Receive the current level of the pollutant from the sensor and save this information for sending it to the central server every 10 minutes. This is performed by the actions $sensor1$ to $sensor6$, receiving an input with the information of each sensor.

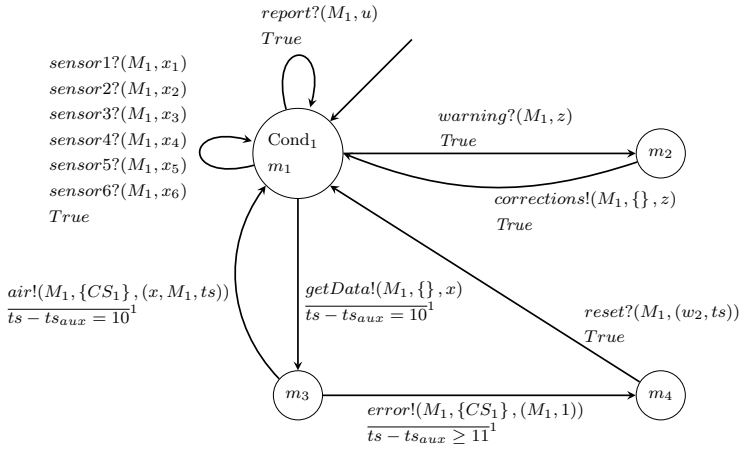


Fig. 4. Sensor station M_1 automaton.

- Data Sending: Send to the central server the information collected, from all its associated sensors, about the amount of pollutants. This is performed by the action *air*, but firstly, data is accumulated in the station with the action *getData*.
- Report an error: Send to the central server the abnormal state of the station because the clock is not set to the right time. This is performed by the action *error*.
- Recover from an error: Apply the solution sent by the server in order to recover the behavior of the station. This is performed by the action *reset*, giving the value of *ts* from the server to the *ts* of the station.
- Save delay: Receive the mean of delays that suffer the clock of the station every hour. This is obtained from the central server and it is performed by the actions *warning*, capturing the information from the server, and *corrections*, saving the delay that should be corrected.
- Save daily report: Receive the daily report generated by the central server with the information obtained from the station. This is performed by the action *report*.

In Figure 4 we present a graphical view of the automaton corresponding to the sensor station M_1. Again, the automaton corresponding to other sensor stations are similar.

The *central server* collects the information captured by all the sensors of the sensor stations. In particular, it controls the delays of the clocks of the sensor stations. The functionality of this element is implemented with the following actions:

- Data Reception: Receive the information of each sensor station and process it. This is performed by the actions *air*, receiving an input with the infor-

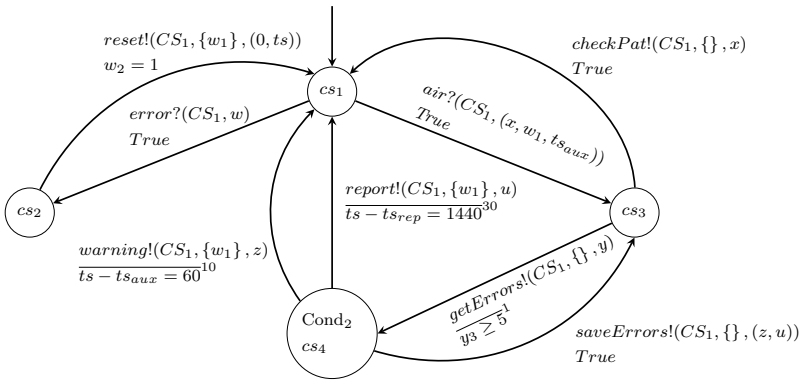


Fig. 5. Central server CS_1 automaton.

mation of the station, and the action *checkPat*, applying the patterns each time that information is received.

- Sending of information: Send to the corresponding sensor station the information obtained from the evaluation of their clocks. Firstly, we save the errors of the clocks with the action *getErrors* when the delay is greater or equal to 5 and they are saved with *saveErrors*. Then, with the action *warning* we send the errors detected every hour and with *report*, we send the information collected of the station during a day.
- Reset from an error: Receive an abnormal state of a sensor station and the server sends the measure to apply with the aim of recovering that station from an error state. This is performed by the actions *error* and *reset*.

In Figure 5 we present a graphical representation of the automaton defining the central server CS_1.

The network requires the synchronization of all the elements in two levels. First, we define the synchronization of the sensors with their corresponding station. The system corresponding to the sensor station i, where $1 \leq i \leq 61$, is:

$$SS_i = ||_A(S_{i1}, S_{i2}, S_{i3}, S_{i4}, S_{i5}, S_{i6}, M_i)$$

where $A = \{sensor1, sensor2, sensor3, sensor4, sensor5, sensor6\}$. Second, we define the synchronization of the sensor stations with the central server. This is formally defined as:

$$N = ||_B(SS_1, SS_2, \ldots, SS_{61}, CS_1)$$

where $B = \{air, error, reset, warning, report\}$.

4 Conclusions

We have introduced a formalism to specify systems with uncertain information. We have defined the syntax and operational semantics of our *fuzzy* version of

the classical timed automata formalism. In order to show the usefulness of our formalism, we have used it to formally represent a real system. Specifically, we present the formal specification of a sensor network conformed by a central server, 61 sensor stations and 366 sensors devoted to the study of air quality. The measurement of these levels are performed periodically and some reports and warnings are generated according to some timing requirements. The main problem, concerning time, is that lacks of synchronization between local clocks and the clock of the server may provoke the malfunction of the network. Although the measure of the quality of the air is the first goal of the network, it is essential to check that the time in this scenario is correct in order to assure that the timing triggers of the actions are accurate.

References

1. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
2. C. Andrés, L. Llana, and M. Núñez. Self-adaptive fuzzy-timed systems. In *13th IEEE Congress on Evolutionary Computation, CEC'11*, pages 115–122. IEEE Computer Society, 2011.
3. B. S. Bosik and M. Ü. Uyar. Finite state machine based formal methods in protocol conformance testing. *Computer Networks & ISDN Systems*, 22:7–33, 1991.
4. A. Cavalli, J.P. Favreau, and M. Phalippou. Standardization of formal methods in conformance testing of communication protocols. *Computer Networks and ISDN Systems*, 29:3–14, 1996.
5. A. R. Cavalli, T. Higashino, and M. Núñez. A survey on formal active and passive testing with applications to the cloud. *Annales of Telecommunications*, 70(3-4):85–93, 2015.
6. M. Doostfatemeh and S. C. Kremer. New directions in fuzzy automata. *International Journal of Approximate Reasoning*, 38(2):175–214, 2005.
7. R. M. Hierons, K. Bogdanov, J.P. Bowen, R. Cleaveland, J. Derrick, J. Dick, M. Gheorghe, M. Harman, K. Kapoor, P. Krause, G. Luettgen, A.J.H Simons, S. Vilkomir, M.R. Woodward, and H. Zedan. Using formal specifications to support testing. *ACM Computing Surveys*, 41(2), 2009.
8. L. Lamport. Who builds a house without drawing blueprints? *Communications of the ACM*, 58(4):38–41, 2015.
9. K.G. Larsen, P. Pettersson, and W. Yi. Uppaal in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1-2):134–152, 1997.
10. D. Mintz. Technical Assistance Document for the Reporting of Daily Air Quality - the Air Quality Index (AQI). Technical Report EPA-454/B-16-002, U.S. Environmental Protection Agency, 2016.
11. J. N. Mordeson and D. S. Malik. *Fuzzy automata and languages: theory and applications*. Chapman & Hall/CRC, 2002.
12. W. G. Wee and K. S. Fu. A formulation of fuzzy automata and its application as a model of learning systems. *IEEE Transactions on Systems Science and Cybernetics*, 5(3):215–223, 1969.