Specifying the Behaviour of Building Automation Systems

João Aguiam

IST - Instituto Superior Técnico

Abstract. Building automation systems have known an upsurge of interesting due to their energy savings potential and for creating smart environments. However the development of this type of systems consists essentially of embedded systems in hardware controllers written in *C* which are difficult to program. In situations where the actuation must vary continually based on the sensor inputs, development and testing becomes even harder. This work proposes a framework based on a declarative language to specify the behaviour of a building automation system. This language allows the specification of the behaviour without worrying about hardware details. This language and framework proposed herein use the notions of fuzzy logic and temporal logic. The specifications developed using the proposed language describe fuzzy control systems that interface with actuators through a different defuzzification process, where we suggest an intermediate approach less demanding than the Mandani fuzzy system but more flexible than the Tsukamoto and Sugeno models. We validate our proposal by developing a simulator of the language semantics and animating a case study specification of the control system of an automated office room. For further validation we used sensor reading traces and observed the behaviour of our animated specification, analysing graphic representations of the actuators along the simulation.

1 Introduction

Building Automation has its origins in the 1980s however in the last decade there has been an upsurge of interest in this thematic. This interesting is essentially related with the potential energy savings that a building can have if it is controlled automatically. However, in a residential environment there are also other research developments focused on converting the home space into a more comfortable place for its inhabitants. In both residential and non residential buildings the area of Artificial Intelligence and Smart Environments have spurred the interest by the research community and of individual users.

To address this situation most automation hardware manufacturers have created suites of hardware modules that came with pre-packaged embedded applications that have to be configured. In either case, to develop an automation system, the user must have considerable expertise regarding hardware details. Besides this issue, the proprietary systems and the intercommunication between different manufacturers devices is also a problem in building automation systems.

In the same way, the majority of building automation systems are based on *C* programs embedded in hardware controllers. With this approach, modifications and adjustments to the system are a difficult and sometimes impossible task. To modify the logic of the programs it is necessary to reprogram these controllers which must be done by an expert, making it very difficult or impossible for the end-user to adapt the system to suit its needs. The development and testing of these system are also time consuming and error prone. Ideally, there should be a clear separation between the hardware controller and the program logic. The specification of the behaviour of the system should be done by the end-user in a friendly way. To this aim, we propose a declarative language for specifying the behaviour of Building Automation Systems. By using a declarative language with a formally declarative language, the focus of this proposal focuses on *what* the user wants the system to do instead of *how* to do it. This has the advantage of approximating the specification task to the domain of the end-user of the system due to the simplicity of this specification, allowing it to be specified by non-expert users. Declarative languages also enable the automatic validation of behaviour specification.

The remaining content in this article is organized as follows. Section 2 presents the related works and concepts with this work like the concepts of building automation, fuzzy logic, temporal logic and the concept of context-aware systems and its fuzzy context aware applications. In the section 3 we describe the proposed declarative language while in the section 4 the framework which use this language is specified. To validate this work we defined a simple case study and discuss its results, which is described in the validation section 5. Finally we finish this article with the listing of possible future works in section 7 and the conclusions we have achieved in the section 6.

2 Concepts and Related Work

This section addresses the main concepts related with this work as well as list some of the most relevant works in this domain.

2.1 Building Automation

The concept of Building Automation (BA) is a very wide topic that touches the subjects of control, automation, energy management, learning and intelligence applied in buildings. Sometimes BA is also equated with Smart Buildings and Smart Homes, as all these concepts refer to automating and controlling buildings. However, these two concepts are rather general concepts while BA refers to concrete technological solutions. We can divide BA into two different domains, the domain of homes and the domain of facilities. This distinction is important because in homes the automation system is more focused in the comfort of the inhabitant, while in the buildings the aim of its automation system lies on the efficiency and reduction of costs of the building operations [9]. Despite having different goals, BA systems aims at controlling electric devices such as luminaries,

HVAC systems (heating, ventilation and air-condition), window blinds, among other in order to reproduce the desired behaviour in the building. All types of Building Automation Systems (BAS) have in common three components: the *sensors*, *controllers* and the *actuators*. Both sensors and actuators are devices installed in the building. The formers give information to the controller in order for it to reason about the building state and perform actions in the actuators, which will define the behaviour of the building.

In this work we will focus our attention in the controller component as it is the core of the specification of the system's behaviour. The controller of a BA system specifies the logic behind the system. In this work we define the components of this system using the concepts of fuzzy logic and temporal logic, which will be introduced on what follows.

2.2 Fuzzy Logic

The notion of fuzziness was introduced by Zadeh with the aim of creating fuzzy sets [19]. These sets tries to simulate the propositions used by humans which are not completely true neither completely false, they have a degree of truth. Zadeh defined a fuzzy set A as a set of pairs formally defined as $A = \{(x, \mu_A(x))|x \in X\}$. where $\mu_A(x)$ is the membership function of the fuzzy set which return the degree of truth of an object x. A membership function can be any function from $X \rightarrow [0, 1]$ that characterizes the fuzzy set. The domain X is also referred as *universe of discourse* or in fuzzy logic terminology as *linguistic variable*. A linguistic variable can have associated with it one or more fuzzy sets that describe a certain interval of the universe. These fuzzy sets are called a *linguistic value* [20,21]. The operations of classical logic theory can be generalized to fuzzy logic by defining the common logic operations in turns of membership functions.

$$\begin{aligned} A \wedge B &= min(\mu_A(x), \mu_B(x) \\ A \vee B &= max(\mu_A(x), \mu_B(x) \\ \neg A &= 1 - \mu_A(x) \end{aligned}$$

Besides the notion of fuzzy sets, in fuzzy logic the notion of fuzzy rules were also introduced. These rules usually assumes the form

$$if\ x\ is\ A\ then\ y\ is\ B,$$

where A and B are linguistic values defined by a fuzzy set in X and Y respectively. These rules are often written as $A \rightarrow B$, to describe a binary fuzzy relation R over the product space $X \times Y$ between the linguistic values A and B, which are represented by the variables x and y. These two linguist values are in different spaces being necessary to transform the fuzzy sets into a fuzzy set over the product space $X \times Y$, which is done through the *cylindrical extension*.

In the fuzzy reasoning process, to calculate the generated fuzzy set of the system one firstly has to calculate the degrees of compatibility of each fuzzy set in the antecedent part of each rule, then the firing strength of each one which

will be propagated to the consequent part generating a fuzzy set for that rule and finally aggregate all the resultant fuzzy sets into a single one. The resultant fuzzy set of a rule (C'), which is a relation $A \times B \rightarrow C$, is achieved through the equation 2.1, where w_1 and w_2 are the degrees of compatibility of A and B respectively, while $w_1 \wedge w_2$ is the firing strength of the rule.

$$\begin{aligned}
\mu_{C'}(z) &= \vee_{x,y}[\mu_{A'}(x) \wedge \mu_{B'}(y)] \wedge [\mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z)] \\
&= \vee_{x,y}[\mu_{A'}(x) \wedge \mu_{B'}(y) \wedge [\mu_A(x) \wedge \mu_B(y)] \wedge \mu_C(z)] \\
&= \{\underbrace{\vee_x[\mu_{A'}(x) \wedge \mu_A(x)]}_{w_1}\} \wedge \{\underbrace{\vee_y[\mu_{B'}(y) \wedge \mu_B(y)]}_{w_2}\} \wedge \mu_C(z) \\
&= \underbrace{(w_1 \wedge w_2)}_{\text{firing strength}} \wedge \mu_C(z)
\end{aligned} \tag{2.1}$$

Fuzzy reasoning process is mostly applied in controller systems which are denominated as *fuzzy control systems*. These systems have the particularity of returning a single crisp value in order to control a certain object. Due to this requirement, an additional component to transform the resultant fuzzy set into a single crisp value is needed. This process is called defuzzification and there are many different techniques to do so. A fuzzy control system is composed of four different components: (1) a fuzzification component which transforms the crisp input into a fuzzy one, recurring to the membership functions defined in the system; (2) a rule-base which contains the fuzzy rules of the system; (3) an inference mechanism that performs the fuzzy reasoning and (4) a defuzzification component to convert the fuzzy output into a single crisp value.

There are three main models to build a fuzzy control system in literature with differences in the defuzzification phase. The Mamdani fuzzy inference system [10] has the particularity of receiving crisp inputs instead of a fuzzy set, so the membership function of each fuzzy set must be used with these values as input. The rest of the fuzzy inference process follows the usual procedure. As a defuzzification schema, Mamdani suggest to use the centroid of area method. The Sugeno fuzzy model was originally proposed by Takagi, Sugeno and Kang [16, 17]. It is characterized by defining the consequent part of a rule as a crisp function in the fuzzy region space specified by the antecedent of the rule. The deffuzification procedure used by this model is a simple weighted average. This method avoids the time-consuming process of defuzzification, however it can lead to the loss of membership linguistic meanings. The Tsukamoto fuzzy model [18] is a model that can be placed in between the Mamdani system and the Sugeno model, as it allows the propagation of the fuzziness from the antecedent to the consequent of each rule, however it also avoids the heavy computation effort to deffuzy the output into a crisp value. This is possible because in this model the consequent membership function has to be monotonical, so the propagation of the firing strength into these membership functions only generate a single value, though the deffuzzification technique applied can be the weighted average [7].

2.3 Temporal Logic

Another topic related with the logic behind the controller component of our BA system is the Temporal Logic (TL). Prior introduced the TL with the aim of reasoning about time, where the operators *eventually* and *always* were suggested [14]. With these operators we may write formulas to express the proposition *"The light will always be turned on"* or *"The room will eventually be occupied"*. Writing *"always p"* means that p is true in a given state s and in all states reachable from s. Writing *"eventually p"* means that p will be true in some state reachable from s. Over the years, TL has been extended with other operators such as *until* or *since* [8].

Temporal Logic can be classified into two categories considering the model of time flow: *Linear Temporal Logic* (LTL) and *Branching Temporal Logic* (BTL). In LTL the model time flow is ordered in a linear sequence, i.e. each state can only leads to a single next state. On the other hand, in BTL, there may be several distinct future states, which model a non-determinism, i.e. we do not know a priori how the system will behave. Usually, this kind of logic is structure as a tree. [4–6].

The common interpretation of LTL formulas is a linear path of states, where each state correspond to an observation of the system in a certain instant t. A path is represented as $\pi = s_0, s_1, \ldots, s_n$, where s_t is the state observed in the system at the instant t, $0 \leq t \leq n$. Reasoning in LTL is made through the analysis of the states in a certain path π according to a specified *formula*. A formula φ can be defined by the following grammar: $\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \circ\varphi \mid \varphi_1 \mathcal{U} \varphi_2$, where the symbols $\mathcal{U}$ and $\circ$ denotes the operators *until* and *next* respectively. The operator next ($\circ\varphi$) says that φ holds in the next state to be observed in the path. The formula $\varphi_1 \mathcal{U} \varphi_2$, using the temporal operator until, means that φ_1 holds since now until the moment that φ_2 holds, which can be either now or somewhere in the future.

Temporal logic is frequently used in the formal verification of the correctness of programs. TL allows the reasoning about past, present and future observations of an observed state which allows us to formally verify the correctness of a program by specifying its properties in this logic [13]. A specification of the behaviour of the program is translated into a graph of state transition. Then, properties about the desired program behaviour written as temporal logic formulas are evaluated over this model. This is known as model-checking [3].

2.4 Fuzzy Context-Aware Systems

Besides theses concepts, some related works must be analysed. These works can be inserted in the domain of Context-Aware Systems (CAS) which are systems that adapt themselves to the surrounding environment. However these related systems have the particularity of reasoning based on fuzzy logic, which is the main topic of our work. Acampora et al. developed a system to control certain aspects of the environment (such as lighting, temperature and blinds) which relates multiagents paradigm and fuzzy logic [1]. Mantyjarvi and Seppanen, in their

work, introduce the concept of fuzzy context in the development of a context-aware system, more specially in handheld devices [11]. The definition of fuzzy context aims at modelling the humans' perceptions of the environment, which are also fuzzy. Park et al. developed a context-aware music recommendation system which is based on fuzzy Bayesian networks with utility theory [12]. They gather context informations from sensors and from the internet which are then pre-processed in a fuzzy system to generate a fuzzy membership vector. This vector is then input in the Bayesian network in order to infer the contexts of the system. Finally, to calculate the music score having into account the user preference, the utility theory is applied. Ranganathan et al. proposed a system for dealing with uncertain contexts [15]. They have developed a model that uses probabilistic logic, fuzzy logic and Bayesian networks in their reasoning with the aim of facilitating the task of incorporating uncertainty in their context-aware applications. Cao et al. also introduce the fuzzy logic theory into context-aware applications, more specially in the adaptation model which is concerned in the reconfiguration of the services of such a system according to the user's current context [2].

3 Context Description Language

In this section we propose a language that can be used in the description of a context. This is a declarative language based on fuzzy logic and temporal logic. We define a context as a combination of different gathered informations from the environment which is represented by a value between 0 and 1. To calculate this value, a context can be defined through a formula expressed by the operators defined in this language. These operators are adaptations of the operators from the fuzzy logic and temporal logic among others defined by us. Some of these operators have different meanings and definitions their first introduction in the logic proposal. A context can be compared with the notion of proposition used in the fuzzy and temporal logic, which can be either fuzzy or crisp, although, in this case, a context is a fuzzy proposition.

A formula that specifies a context can be defined by the following grammar: $C ::= c \,|\, f(r_t(s)) \,|\, \uparrow C \,|\, \downarrow C \,|\, \neg C \,|\, C_1 \wedge C_2 \,|\, C_1 \vee C_2 \,|\, C_1 \wedge_b C_2 \,|\, C_1 \widetilde{\mathcal{U}} C_2$, where c is a constant between 0 and 1, $f(r_t(s))$ a function $f : S \rightarrow [0, 1]$ where $r_t(s)$ is a sensor reading at instant t of the sensor s and S its domain; and the symbols $\uparrow$, $\downarrow$, $\wedge_b$ and $\widetilde{\mathcal{U}}$ the operators *Raise Transition*, *Fall Transition*, *Combination* and *Until* respectively. The operator raise verifies if the context value has increased from the previous instant t while the fall checks a decreasing in the context. The operator combination can be seen as a weighted multiplication between two context by a certain factor b between 0 and 1. Finally, the operator until defined in this work holds (in a fuzzy way) since the value of the first context argument holds until the value of the second one holds. This definition is different than the one defined for the until operator in TL. Let $[\![C]\!]_t$ be the interpretation of the context C at the instant t, C_1 and C_2 be two contexts and b a factor between 0 and 1, we formally define the operators in this language as:

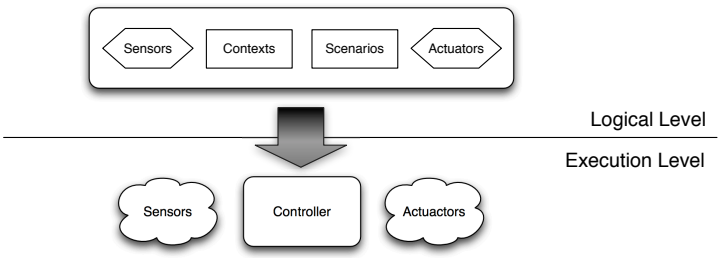


Fig. 1: Proposed framework architecture. This architecture can be analysed in two distinct levels, the logical level where the components of a controller are defined and the execution level.

$$
\begin{aligned}
[\![\uparrow C]\!]_t &\stackrel{\text{def}}{=} \begin{cases} [\![C]\!]_t - [\![C]\!]_{t-1} \text{ if } [\![C]\!]_t - [\![C]\!]_{t-1} > 0 \\ 0 \qquad\qquad\qquad\quad \text{otherwise} \end{cases} \\
[\![\downarrow C]\!]_t &\stackrel{\text{def}}{=} \begin{cases} [\![C]\!]_{t-1} - [\![C]\!]_t \text{ if } [\![C]\!]_{t-1} - [\![C]\!]_t > 0 \\ 0 \qquad\qquad\qquad\quad \text{otherwise} \end{cases} \\
[\![\neg C]\!]_t &\stackrel{\text{def}}{=} 1 - [\![C]\!]_t \\
[\![C_1 \wedge C_2]\!]_t &\stackrel{\text{def}}{=} min([\![C_1]\!]_t, [\![C_2]\!]_t) \\
[\![C_1 \vee C_2]\!]_t &\stackrel{\text{def}}{=} max([\![C_1]\!]_t, [\![C_2]\!]_t) \\
[\![C_1 \wedge_b C_2]\!]_t &\stackrel{\text{def}}{=} (1-b) \times [\![C_1]\!]_t + b \times [\![C_2]\!]_t \\
[\![C_1 \widetilde{\mathcal{U}} C_2]\!]_t &\stackrel{\text{def}}{=} \begin{cases} min(max([\![C_1]\!]_t, [\![C_1 \widetilde{\mathcal{U}} C_2]\!]_{t-1}), 1 - [\![C_2]\!]_t) \text{ if } t > 0 \\ min([\![C_1]\!]_t, 1 - [\![C_2]\!]_t) \qquad\qquad\qquad\quad \text{if } t = 0 \end{cases}
\end{aligned}
$$

4 Proposed Framework

In this section we present the framework behind this work which is based on the declarative context description language and fuzzy inference process. We start by describing the architecture and then detail the fuzzy inference process.

4.1 Architecture

In this framework, behaviour specification resides in a logical level and is based on four main components: *sensors*, *contexts*, *scenarios* and *actuators* (see Fig. 1).

Sensors are the way by which the system captures the environment of the building and its properties. They act like an interface between the system and the physical building. In this framework we suggest an event triggering approach as the system reacts and interprets the event from the sensors as soon as it happens

instead of checking the state of the environment with a predefined time interval of time.

Contexts are abstractions of the physical environment and are used in the conditions of the fuzzy rules. They are represented as a value between 0 and 1 which represent the degree of confidence of that context being true. The definition of the contexts is done using the context description language which specifies the context as a fuzzy set. Contexts can be divided into low level contexts which represents a normalized value of a given sensor, and high level contexts which can be values both from sensors or from other contexts that are already normalized. It is important to notice that a context must have its initial value pre-defined.

The scenarios specify the behaviour of the system through the definition of fuzzy rules. This component performs the fuzzy inference of the system which decides the actuators that should be manipulated and with which values. A scenario is a set of fuzzy rules, however for the system only the rules matter. The aggregation of the rules into scenarios is only an abstraction for the user important in terms of engineering, by modelling the behaviour into different scenarios. The rules specified in this system are if-then fuzzy rules with the following representation: if *antecedent* then *consequent*.

Finally, the actuators are the ultimate goal of the system as they are the ones that will effectively perform the desired behaviour in the building. They transform the action received by the system into physical changes in the devices installed in the building.

4.2 Fuzzy Inference Process

The fuzzy rules used in this framework are divided into two parts, the *antecedent* and the *consequent*.

The antecedent part can have one or multiple sub-antecedents, which are connected by a fuzzy connector: $\wedge$ and $\vee$. Each of these sub-antecedents are a fuzzy condition expression which is given by the following grammar: $cond ::= C \mid \chi\, C \mid \neg cond$, where χ is a fuzzy context operator. This operator is a function $f : [0, 1] \rightarrow [0, 1]$. The application of the fuzzy context operator to a certain context C generate a fuzzy set characterized by a membership function given by the function f. This fuzzy set represents the degree of truth of the context C belonging to the semantically meaning of the application of the fuzzy context operator to this context.

To evaluate the antecedent part of a rule, one has to evaluate each sub-antecedent and then calculate the overall value of the antecedent having into account the fuzzy connectors used between each sub-antecedent. If the fuzzy connector is the *and*, the *min* operation between the two sub-antecedents is returned while if the connector is the *or*, the *max* operation is used instead. The calculated value of the antecedent part of the rule is then propagated to the consequent part.

The consequent part is given by the following grammar: $consequent ::= actuator = \gamma$, where γ is a fuzzy consequent operator. This operator is a fuzzy set in the universe of discourse of the actuator *actuator*. A fuzzy context operator

is defined by the correspondent membership function of its fuzzy set as well as the inverse function of its membership function in order to transform a fuzzy value into a crisp one.

The aim of this system is to control the actuators of a building in order to perform the desired behaviour. However, a fuzzy consequent operator defines a fuzzy set to be attributed to the actuator, which in practice is not possible as the actuators can only assume a crisp value as control input. To solve this issue, a deffuzification strategy must be applied.

In literature, there are many different deffuzification schemas used in transforming a fuzzy set into a single representative crisp value and some different ways of implementing a fuzzy control a system, which differ essentially in the deffuzification process. In this framework we propose an intermediate system between the Mandani fuzzy inference system and the Tsukamoto fuzzy model. In the Tsukamoto model, the deffuzification is a simple weighted average due to the restriction of having a consequent fuzzy set represented by a monotonous membership function, which generates a single crisp value. To simplify the Mandani system and not impose any restriction to the developer of the fuzzy consequent operator, we propose a system where the propagation of the antecedent value to the consequent generates a crisp value as it happens in the Tsukamoto model, regardless of the fuzzy consequent operator chosen. To accomplish this, the propagation of the antecedent value when intersected with the consequent fuzzy set, instead of generating a fuzzy set, it will opt to choose a single crisp value. Depending on the consequent fuzzy set, the intersection of the antecedent value of the rule with its consequent fuzzy set may produce more than one value, but in a fuzzy control system we will have to choose only one. This framework proposes that the value chosen is the closer to the current value of the actuator.

However, there are many rules that actuate on the same device, so different values must be set on a device where only one value can be chosen. To solve this conflict, we opt to choose the method used both in the Sugeno and in the Tsukamoto models, which calculates the weighted average of the antecedent value with the resultant consequent crisp value of each rule. The weighted average is the application of the centroid of area method when there are only crisp values. This method consider both the strength of a rule as well as its resultant value, by weighting them.

To summarize this inference process, it can be divided into three steps:

1. Calculate the antecedent value of each rule, which will be denoted by w_i and is named by firing strength.
2. Propagate the firing strength of the antecedent part to the consequent part in each rule by retrieving the resultant single crisp value z_i.
3. Aggregate the resultant crisp values for each actuator by calculating its weight average given by the following formula:

$$actuator \;=\; \frac{\sum_i w_i z_i}{\sum_i w_i}.$$

5 Validation

To validate this work we firstly have defined a simple case study that illustrate a real application of the proposed framework and then we have developed a simulator to apply and test this work. In this section we describe the case study as well as we discuss the results of the tests performed.

5.1 Case Study

This case study reflects a simple office which is a small room with a desk and a support table used for meetings. In the ceiling there are two luminaries, one closer to the window and another to the inner wall. Near the window there is the desk where the occupant usually works, while near the inner wall there is placed the support table. To prevent the excessive inside glare and ensure privacy of the occupants, the windows have curtains that can be adjusted throughout the day. It is important to refer that in this case study we have only considered one person in the room at the same time.

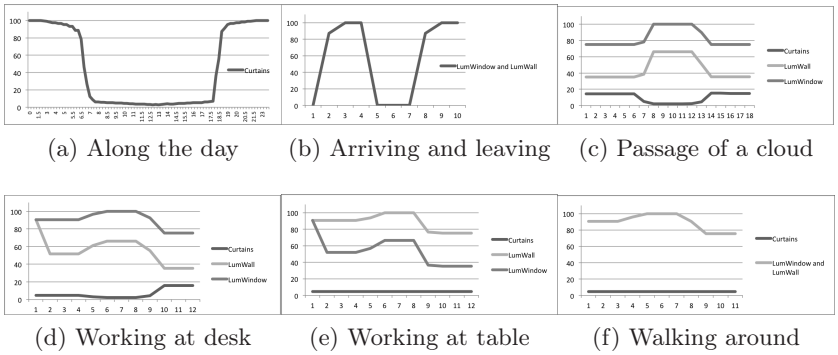
Following the proposed framework we started the specification of this case study by listing the sensors installed in the office, then we define the contexts needed in the fuzzy conditions of the rules specified in the scenarios.

In this case study we have the following sensors: a door sensor to detect the entrance or leaving of someone; the table sensor and desk sensor to detect which activity is the user performing; two luminosity sensors inside the room, one near the window and other near the inner wall which allow the inference of the amount of light over the support table and the desk; a luminosity sensor placed in the exterior; and a virtual time sensor to allow the reasoning about time.

The contexts are one of the core components of this framework as the fuzzy conditions are all based on them. To define the contexts we have used the proposed declarative language. We can separate the contexts defined in this case study into two categories, low level and high level contexts.

As low level contexts we have the ones that deal directly with the sensors by normalizing its values as well as some lighting models. The luminosity sensors output a value bigger which is not normalized, so for each one we have defined a context which is a simple linear normalization of the sensors. Besides this, two lighting models have been specified which indicates the predictable amount of day and night for a given time.

As high level contexts, we have defined the core contexts of this case study. Firstly we define the contexts *Arriving* and *Leaving* in order to detect if the user is arriving or leaving the room. To define these two contexts we use the operators raise and fall defined in the proposed language. The context *InTheRoom* is defined based on the two previous contexts using the operator until. Related with the activity of the user we have defined the contexts *WorkingDesk*, *WorkingTable* and the *WalkingAround*. To reason about the amount of light in each workplace, we defined some high level contexts which are based on the normalized luminosity contexts. They are the contexts *TableIlluminated*, *DeskIlluminated* and



(a) Along the day (b) Arriving and leaving (c) Passage of a cloud

(d) Working at desk (e) Working at table (f) Walking around

Fig. 2: Graphic representation of the simulations performed

RoomIlluminated. Besides these contexts, two more were also defined in order to reason about the *Day* and *Night*. These contexts use the lighting models weighted with the luminosity low-level contexts.

The next step specified in the framework proposed is the definition of the scenarios and its rules. However, before specifying this component, it is necessary do define the fuzzy context and fuzzy consequent operators which are used in the rules. As fuzzy context operators, we have only defined three operators to qualify the value of a given context: *Insufficient*, *Sufficient* and *TooMuch*. As fuzzy consequent operators we have defined the ones related with our actuators, more specially related with the luminaries and the curtains. For the curtains actuator the operators defined were *Completely Closed*, *PartiallyClosed*, *PartiallyOpened* and *CompletelyOpened*. For the luminaries actuators we have defined the operators *FewLight*, *NormalLight* and *MuchLight*. With these operators defined we could then specify our scenarios which were related with the activity of the user (*Arriving*, *Leaving*, *Working at Desk*, *Working at Table* and *WalkingAround*) and a scenario for *Privacy* issues which has the aim of avoiding that someone look to inside the office specially at night.

Finally, as actuators of this case study we have two luminaries, one near the window and other near the inner wall, and curtains at the window.

5.2 Discussion

The case study defined above is the basis of all the validation process. To validate this work the building of an example system of the proposed framework is needed. However, after having defined the case study, we have not build immediately the desired system. We have started to developed a simulation in an *Excel* sheet as it is a quick way to apply and test the framework. Despite being a good way to quickly test the system, it is not very robust, so we decided to develop a graphical simulator where one can interact with the system and simulates the real events occurring in a building. Finally, for more robust validation tests, with

the basis of this simulator, we have converted it to a non-graphical simulator in order to accept input flat files and outputting a *csv* file that could be analysed with more details and generate some graphics that illustrate the behaviour of the system. Figure 2 illustrates the graphics generated in the validation process.

We performed six simulations with series of events that retract some particular scenarios that could happen in a real case. The scenarios specified in the case study were also tested with independent input files. The evolution of the curtains along the day controlled by this system is represented graphically in the figure 2a. Here we can see that as the sun is risen the curtains remain open while during night they are closed. To test if the lights turn off when someone leaves the room as well as in the arriving we have performed a simulation illustrated in the figure 2b. This simulation shows that while the user is outside the office the luminaries are turned off and in the arriving they are set to a maximum value because there is insufficient light in the office. Another test performed in our validation process was to test the evolution of the multiple actuators in the room when a cloud passes in the sky. In this case the curtains must be completely opened to allow the entrance of more light inside ans the lights should increase their values to compensate the reduction of natural light. This behaviour in seen in the figure 2c.

Besides the validation we have also tested this system when simulating the three different activities of the user with slightly the same simulation input. These tests simulates that there is sufficient light in the given in the room, then there is insufficient light in the specific workplace and finally the light in that workplace is too much. In the case the user is working at the desk (see figure 2d) there are two points to note, the different values of $LumWindow$ and $LumWall$ and the partially closing of the curtains to avoid the excessive glare over the desk. In the case of the support table (see figure 2e) the excessive amount of light does not disturb the work of the user making the curtains to remain constant along the simulation. However, the values of the two luminaries is not the same as well. Finally, the last test performed evaluates the actuators in the activity working around (see figure 2f). In this case both the luminaries have the same value as the office is seen as a whole. By analysing the result of the simulations performed, one can conclude that they can validate this work, however more robust validations, specially in a real environment, should be performed.

6 Conclusions

In this work we presented a system that allows the specification of the behaviour of a BAS where the actuators varies continuously based on sensors, in an easy way to the user. This was achieved through a proposal of a declarative language based on fuzzy logic and temporal logic. We have described this language as well as the whole system that use such language. In order to validate the worked done, we have created a case study that applies the proposed framework to exemplify its use and perform validation tests on it. The validation tests where performed on a simulator of this system. With the discussion of the results achieved we

can conclude that the tests performed validate this work in order to specify the behaviour of a BAS.

This work resulted in the proposal of a declarative language for specifying the behaviour of building automation systems with the following main contributions:

- Framework proposal to specify the behaviour of building automation systems based on the fuzzy logic.
- A formal syntax and semantics of the language proposed drawing on concepts borrowed from fuzzy logic and temporal logic.
- Adaptation of the concept of *"context"* from intelligence environment to fuzzy logic.
- A generalization of the temporal *Until* operator to deal with fuzzy propositions.
- A new approach to defuzzyfication process of fuzzy control systems that is computationally less demanding than the Mandani fuzzy system but more flexible than Tsukamoto and Sugeno models.

7 Future Work

This work has touched some different subjects and has presented some new contributions, however there are some topics that can be developed further. One aspect critical to effectively validate this work is a more robust validation. More case studies based on the proposed framework should be performed as well as to implement it in a real environment. When we started this work, our initial aim was to develop a BAS capable of learning the routines of the user, however we have directed the work to the specification of the behaviour instead of the learning. With the basis of this solution it would be easier to incorporate a learning strategy. Another possible future work related with the proposed system and language is the creation of a compiler for the propose declarative language. With such a compiler the user could specify their own rules and implement itself the system. Finally this solution and the logic inherent to it could be applied in other domains such as business for example.

References

1. G. Acampora, M. Gaeta, V. Loia, and A. Vasilakos. Interoperable and adaptive fuzzy services for ambient intelligence applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 5(2):1–26, 2010.
2. J. Cao, N. Xing, A. Chan, Y. Feng, and B. Jin. Service adaptation using fuzzy theory in context-aware mobile computing middleware. 2005.
3. E. Clarke. Model checking. In *Foundations of software technology and theoretical computer science*, pages 54–56. Springer, 1997.
4. E. Clarke and E. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. *Logics of Programs*, pages 52–71, 1982.
5. E. Emerson and J. Halpern. "sometimes" and"not never" revisited: on branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.

6. E. Emerson and J. Srinivasan. Branching time temporal logic. *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, pages 123–172, 1989.
7. J. Jang, C. Sun, and E. Mizutani. Neuro-Fuzzy and Soft Computing-A Computational Approach to Learning and Machine Intelligence [Book Review]. *Automatic Control, IEEE Transactions on*, 42(10):1482–1484, 1997.
8. J. Kamp. Tense logic and the theory of linear order. 1968.
9. W. Kastner, G. Neugschwandtner, S. Soucek, and H. Newmann. Communication systems for building automation and control. *Proceedings of the IEEE*, 93(6):1178–1203, 2005.
10. E. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1):1–13, 1975.
11. J. Mantyjarvi and T. Seppanen. Adapting applications in handheld devices using fuzzy context information. *Interacting with Computers*, 15(4):521–538, 2003.
12. H. Park, J. Yoo, and S. Cho. A context-aware music recommendation system using fuzzy bayesian networks with utility theory. *Fuzzy Systems and Knowledge Discovery*, pages 970–979, 2006.
13. A. Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*, pages 46–57. IEEE, 1977.
14. A. Prior. Time and modality. 1957.
15. A. Ranganathan, J. Al-Muhtadi, and R. Campbell. Reasoning about uncertain contexts in pervasive computing environments. *IEEE Pervasive Computing*, 3(2):62–70, 2004.
16. M. Sugeno and G. Kang. Structure identification of fuzzy model. *Fuzzy sets and systems*, 28(1):15–33, 1988.
17. T. Takagi and M. Sugeno. Fuzzy identification of system and its applications to modelling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 15:116–132, 1985.
18. Y. Tsukamoto. An approach to fuzzy reasoning method. *Advances in fuzzy set theory and applications*, pages 137–149, 1979.
19. L. Zadeh. Fuzzy sets*. *Information and control*, 8(3):338–353, 1965.
20. L. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *Systems, Man and Cybernetics, IEEE Transactions on*, (1):28–44, 1973.
21. L. Zadeh. The concept of a linguistic variable and its application to approximate reasoning–I* 1. *Information sciences*, 8(3):199–249, 1975.