Towards High-Level Fuzzy Control Specifications for Building Automation Systems

Juan C. Vidal[1], Paulo Carreira[2], Vasco Amaral[3], Joao Aguiam[2], and João Sousa[4]

[1] Centro de Investigación en Tecnoloxías da Información (CiTIUS), Universidade de Santiago de Compostela, Spain
juan.vidal@usc.es
[2] INESC-ID, Instituto Superior Técnico, Unversidade de Lisboa, Portugal
paulo.carreira@tecnico.ulisboa.pt,joao.aguiam@ist.utl.pt, jmsousa@tecnico.ulisboa.pt
[3] NOVA LINCS, DI, FCT Universidade Nova de Lisboa, Portugal
vasco.amaral@fct.unl.pt
[4] IDMEC, Instituto Superior Técnico,
jmsousa@tecnico.ulisboa.pt

Abstract. The control logic underlying building automation systems has consisted, traditionally, of embedded discrete programs created using either low-level or proprietary scripting languages, or using general purpose 4th generation visual languages like Simulink. It is also well known that programs developed in this way are hard to evolve, test, and maintain. These difficulties are intensified when continuous control problems have to be tackled or when the actuation must vary continually subject to the sensor inputs. Such is the case in day-lighting or occupancy-based control applications. In this paper, we propose a declarative high-level Domain-Specific Language (DSL) that aims to reduce the effort required to specify the control logic of building automation systems. Our language combines fuzzy logic and temporal logic, enabling to define the behaviour in terms of domain abstractions. Finally, the approach has been validated in two ways: i) in a case study that simulates the control system of an automated office room; and, ii) by means of an empirical study to confirm usability (with a System Usability Scale questionnaire) and effectiveness, here regarded from the perspective of correctness, of the proposed language with respect to a well known language like Simulink.

Keywords: Ambient Intelligence · Context-Aware Systems · Building Automation · Fuzzy Control Systems · Domain-Specific Languages

1 Introduction

Building automation (BA) refers to the concept of coordinating electrical devices to deliver building services in the most efficient way while minimizing human

intervention. Despite BA systems have been around since the 1980s, the internet-of-things (IoT) has propitiated an upsurge of interest in this domain. In addition, since it has been demonstrated that these systems can play a critical role in lowering the energy consumption in buildings, even international organizations and governments have started to endorse their adoption [50,7,10].

In a BA system, devices such as luminaries, window blinds, or HVAC systems, among others, besides being attached to a traditional power network, are also connected to a digital networked bus and exchange messages with each other. Broadly speaking, electrical devices consist of *sensors* and *actuators*. On the one hand, sensors send messages to the bus informing about the physical reality or events, such as, the current temperature, a door that just opened, or a button being pushed. On the other hand, actuators are responsible for changing this reality. For instance, messages sent to actuators can turn lights on, open a gate, or close a specific valve. The third type of device, known as *controllers*, receive messages from the bus and orchestrate the actuator devices. Traditionally, the logic of controllers in most BA systems has been developed using low-level embedded programming languages like assembly, C-language, or PLC programs. Also, other approaches bring the accidental complexity of general purpose languages to the visual level, like the 4th generation language Simulink.

Although there is no single factor driving the cost of BA systems, it is well known that developing, testing, and maintaining the control logic of BA components is time-consuming and error-prone. Moreover, it requires considerable technical expertise. One essential problem is that embedded programming languages are essentially discrete and fail in addressing continuous control problems where actuators must be driven continually according to information coming from the sensors (such as *daylight harvesting* or *occupancy-based control*). Taking this into account, most efforts on the field were focused on BA control, mainly targeting performance and efficiency but not concerned about the cognitive gap that hinders how end-users can implement or change the behaviour of their control system with effectiveness and satisfaction. These difficulties related to the peculiarities of the BA system and the cognitive gap have been recognized over the years, e.g. [39,40,44,26].

Many approaches to common control have been proposed in the literature. However, these controllers have some disadvantages. They usually need a model of the building and have difficulties in controlling non-linear parameters [41]. Furthermore, the controller's design is not friendly and directly depends on the paradigm used to create the control model. For instance, model predictive control (MPC) [1,35], artificial neural networks [19,4], or linear and non-linear programming languages [49,34] are just some examples of BA controllers that are hard to interpret by the typical end-user. Indeed, these control schemes are quite different from how humans reason, which is undoubtedly a problem [27].

Fuzzy logic [60] is one of the paradigms that closes the gap between machine and human reasoning, among other reasons because it provides the means to reason with uncertainty, imprecise, or fuzzy knowledge. Moreover, fuzzy control [15] looks like a PI or PID controller (Proportional Integral Derivative) but,

in contrast with classical automation methods, it does not require an explicit numerical model of the system to be controlled. In fact, a fuzzy logic controller (FLC) is implemented by *if-then* rules which are similar to the natural language of humans and bring the concept of computing with words closer to the user [56]. This is certainly one of the reasons why FLCs have been employed successfully in many application domains, including BA, as a means of capturing the knowledge to synthesize hardware controllers that operate in real-time [33,43,37].

BA specialists should be able to specify system behaviour in a friendly way, easy to understand, test, and re-use. To this aim, we propose a domain-specific declarative language, based on fuzzy logic control, for defining the behaviour of BA systems. A declarative language with a formal semantics enables focusing on what the user wants to do instead of how to do it. As aforementioned, fuzzy logic provides the means to define control in a human-like way, simplifying thus the synthesis of control programs. However, a more BA-oriented language should facilitate its use by control designers.

Although there is no consensus about a single definition of context [9,2], a context is an abstract state that characterizes a situation relevant from the interaction viewpoint between the system and the user/environment. Aspects, such as the current situation, location, nearby people, changes in the objects, time, users' identity, and emotional state are usually taken into account. In this paper, we define a Domain-Specific Language (DSL) adapted to the BA control needs, that allows the definition of contexts that can capture user intentions and the environment state from the sensors. Furthermore, it also allows the description of rules to capture complex situations, such as the combination of contexts, which may be very useful for conflict resolution.

Also, the proposed language integrates some operators of temporal logic. As it will become clear later, this is a significant improvement compared to traditional FLCs that usually do not allow reasoning about time. However, this feature is necessary to control complex situations. Specifically, we use *Linear Temporal Logic* (LTL) so the model time flow is ordered in a linear sequence, i.e., each state can only lead to a single next state.

We also compared our DSL with Simulink, which is a graphical programming environment for modelling, simulating and analyzing multi-domain dynamical systems, widely used for automatic control. Our evaluation compares both languages asking subjects to undertake modelling challenges and concludes that the proposed DSL scores above Simulink in terms of usability and expressiveness.

Finally, it should be noticed that the proposed language is not tied to any domain, although it has been created bearing in mind BA. In fact, its ability to handle uncertain information and to reason with that information, but in different moments in time, is a clear need in many other domains. Moreover, our DSL organizes the definition of a controller from sensors, actuators, contexts, and scenarios, which are common elements in any control domain. A direct domain of application is IoT, where the ability of this DSL to explicitly define rules able to reason about past events could support the design of more complex control systems. Smart cities, industrial Internet, or connected health could benefit from

this ability and handle better complex behaviours and, thus, go from the traditional reactive control to a more responsive one, able to consider long and short term past outcomes.

The rest of the paper is organized as follows. In Section 2, we present an overview of relevant concepts regarding fuzzy logic and temporal logic. In Section 3 we analyze recent approaches in BA dealing with control design. In Section 4 and Section 5 we describe the proposed framework and language, which is validated in Section 6 with a simple case study. In Section 7, we describe the empirical study held with subjects to evaluate the proposed language. Finally, the conclusions and future work are summarized in Section 8.

2 Background

This section introduces the main concepts, syntax and semantics of Fuzzy Logic and Temporal Logic on which the language specification language that we propose is based.

2.1 Fuzzy Logic

Fuzzy logic [60] generalizes propositional logic to enable reasoning over imprecise and uncertain information. In fuzzy logic, propositions have truth values that range in degree between 0 and 1, in contrast with the crisp false and truth of two-valued logic. One motivating example is the definition of the predicate $hot(x)$ that holds whenever a place x, in a given universe X of places, is considered hot. A threshold above which a place is deemed to be hot does not capture the implicit definition of a hot place (i.e., at which temperature should it be considered hot?).

A fuzzy set A over a universe X of objects is defined as the set $A = \{(x, \mu_A(x))|x \in X\}$, where $\mu_A(x) : X \to [0, 1]$ is a function, known as the *membership function*, that characterizes the membership of the elements $x \in X$, indicating the degree to which a certain object x belongs to a fuzzy set A. Using fuzzy sets, the notion of a hot place can be defined as a certain fuzzy set A over the universe X of temperatures. The membership function $\mu(x)$, depicted in Fig. 1, displays a continuous transition from 0 to a full degree of membership, where, for instance, a temperature of 26 °C it is not considered a completely hot place, while 35 °C is considered a hot place.

In fuzzy logic terminology [57,58], a universe X is called a *linguistic variable*. Each linguistic variable has associated one or more fuzzy sets that describe certain intervals of the universe. These fuzzy sets are called *linguistic values* [25]. For instance, the fuzzy set *'hot place'* (a linguistic value) is defined over the linguistic variable *temperature*. Further linguistic values can be defined over the linguistic variable *'temperature'*, such as *'cold place'* or *'mild place'*, with their corresponding membership functions. The most common shapes of membership function are *triangular*, *trapezoidal*, *gaussian*, *bell* and *sigmoid* functions.This section introduces the main concepts, syntax and semantics of Fuzzy Logic and

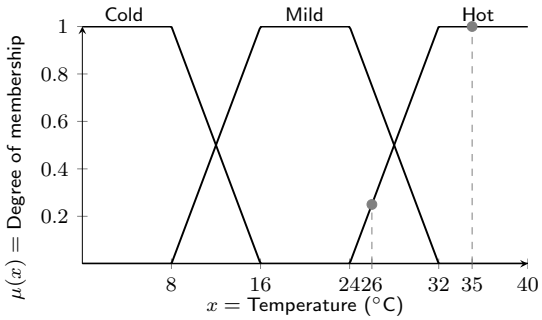


Fig. 1. Example of the trapezoidal membership functions $Cold(x)$, $Mild(x)$, and $Hot(x)$ for the variable *temperature*

Temporal Logic on which the language specification language that we propose is based.

Fuzzy Logic Operations Let X be a universe and A and B be fuzzy sets with corresponding membership functions $\mu_A(x)$ and $\mu_B(x)$. The complement, union, and the intersection of fuzzy sets over X can be defined as:

$$\begin{aligned}
\overline{A} &\stackrel{\text{def}}{=} \{(x, 1-\mu_A(x)) \mid x \in X\} \\
A \cup B &\stackrel{\text{def}}{=} \{(x, max[\mu_A(x), \mu_B(x)]) \mid x \in X\} \\
A \cap B &\stackrel{\text{def}}{=} \{(x, min[\mu_A(x), \mu_B(x)]) \mid x \in X\}
\end{aligned}$$

When the operands of a binary operation are in different spaces, e.g., B is defined over a distinct universe Y, the resulting set is defined over the product space $X \times Y$. Given a fuzzy set A defined by a membership function $\mu_A(x)$ over X, the *cylindrical extension* of A by Y, represented as $A_{X\times Y}$ transforms A into a fuzzy set over the product space $X \times Y$ with a membership function $\mu_A(x,y) = \mu_A(x),\ \forall (x,y) \in X \times Y$, defined as:

$$A_{X\times Y} \stackrel{\text{def}}{=} \{(x, y, \mu_A(x)) \mid (x,y) \in X \times Y\}$$

The dual of the extension operator is the *fuzzy projection*. The projection of a fuzzy set A defined over $X \times Y$ is a fuzzy subset of X denoted $\Pi_X(A)$ defined as:

$$\Pi_X(A) \stackrel{\text{def}}{=} \{(x, \sup_{x\in X}(\mu_A(x,y))) \mid (x,y) \in X \times Y\}.$$

Mutatis mutandis for $\Pi_Y(A)$.

Fuzzy Inference Fuzzy inference rests on fuzzy rules and on the implication operator. Given linguistic values A and B defined over linguistic variables X and Y, a *fuzzy implication*, $A \rightarrow B$, takes the from "IF x *is* A THEN y *is* B".

A fuzzy implication denotes a fuzzy set defined over the product space $X \times Y$ with a two-dimensional membership function $\mu_{A \to B}(x, y) = f(\mu_A(x), \mu_B(y))$, that combines the membership functions of the sets A and B, known as *fuzzy implication function* [25,21]. From the implication $A \to B$, one can infer that proposition B (the *consequent*) holds whenever A (the *antecedent*) holds.

Fuzzy modus ponens [59], which generalizes classical *modus ponens*, is the inference mechanism used in fuzzy control. Let A and A' be fuzzy sets over X, B a fuzzy set over Y, and $A \to B$ defined over $X \times Y$ with a membership function $\mu_{A \to B}(x, y)$, the application of the *fuzzy modus ponens*, is a scheme:

Rule:	$A \to B$
Fact:	A'
Conclusion:	B'

where the conclusion B' is inferred by computing the degree of overlap between A' and A, and then projecting this value according to the strength of the causal link[5] from A to B. Formally, it is defined as $B' = \Pi_Y(A'_{X \times Y} \cap (A \to B))$ or, equivalently, $\mu_{B'}(y) = \Pi_Y\big(A'_{X \times Y} \cap (A \to B)\big)(y)$ simplifies to $\mu_{B'}(y) = \sup_{x \in X}\big(min[\mu_{A'}(x),\ \mu_{A \to B}(x, y)]\big)$.

Fuzzy Control Fuzzy control systems are characterized by a set of fuzzy rules, where the antecedent is a fuzzy condition and the consequent is a control action. Let $X_1, ...X_n$ and Y be universes. A fuzzy system is a structure $\Phi = \langle \{X_1, ..., X_n\}, Y, \mathcal{R} \rangle$ where $\mathcal{R}$ is a set of rules in the form:

$$R_i\text{: IF } x^1 \textit{ is } A_i^1 \text{ AND ... AND } x^n \textit{ is } A_i^n \text{ THEN } y \textit{ is } C_i$$

where A_i^j, for $1 \leq j \leq n$, denotes a fuzzy set over the universe X_j and C_i a fuzzy set over Y.

The first step in fuzzy control is to take the crisp inputs and determine the degree to which these inputs belong to appropriate fuzzy sets $A'_1, ..., A'_n$. Each rule R_i is then evaluated with the fuzzified inputs, and its value w_i is propagated through the fuzzy implication function to determine the consequent C'_i (a variant of C_i) computed as $\mu_{C'_i}(y) = T(\mu_{A_i^1}(x_1), ..., \mu_{A_i^n}(x_n)) \to \mu_{C_i}(y)$. The values of the consequents are then aggregated into a single fuzzy set C with membership function $\mu_{C'}(w) = agg\{\mu_{C'_1}(w_1), ..., \mu_{C'_n}(w_n)\}$. Finally, the resulting set is then defuzzified, i.e., converted into a crisp value to be assigned to a control variable. An abundance of aggregation and defuzzification functions have been proposed in literature [25].

2.2 Linear Temporal Logic

Classical propositional logic does not allow reasoning about time. To solve this issue, temporal logic introduces modalities referring to time, inspired by the

[5] In a sense, this follows the classical modus ponens where the conclusion is obtained from the expression $A \wedge (A \to B)$.

traditional modalities of truth. In this paper, we use Linear Temporal Logic (LTL) with the past to represent the time flow since it fits the behaviour of control systems. The common interpretation of LTL formulas is a linear path of states, where each state corresponds 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, $0 \leq t \leq n$.

Let $\top$ stand for true, p be an atomic proposition, $\neg$ and $\wedge$ be the boolean operators for negation and conjunction respectively, $\oplus$ and $\mathcal{U}$ be the temporal operators *next* and *until*, a LTL formula φ is defined by the following grammar:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \oplus\varphi \mid \varphi_1 \mathcal{U} \varphi_2.$$

Other classic boolean connectors, such as $\vee$, $\Rightarrow$ and $\Leftrightarrow$, and TL operators can be derived from the previous definition [46]. In this paper, we use $\bot$ (*false*), $\diamond\varphi$ (*eventually*), $\Box\varphi$ (*always*) and $\varphi_1 \mathcal{R} \varphi_2$ (φ_1 *releases* φ_2) to abbreviate $\neg\top$, $\top\mathcal{U}\varphi$, $\neg(\top\mathcal{U}\neg\varphi)$ and $\neg(\neg\varphi_1\mathcal{U}\neg\varphi_2)$, respectively.

Let φ be a LTL formula, $\pi = s_0, s_1, \ldots, s_n$ a path, and s_i a state observed in the path π at instant i. We say that φ is valid for the state s_i of the path π (with $i < n$, where n is the length of π) writing $s_i \models \varphi$ iff:

$s_i \models p$	iff $p \in s_i$
$s_i \models \neg\varphi$	iff $s_i \not\models \varphi$
$s_i \models \varphi_1 \wedge \varphi_2$	iff $s_i \models \varphi_1 \wedge s_i \models \varphi_2$
$s_i \models \oplus\varphi$	iff $s_{i+1} \models \varphi,\ i+1 \leq n$
$s_i \models \varphi_1 \mathcal{U} \varphi_2$	iff $\exists_k (k \geq i)$ such that $s_k \models \varphi_2 \wedge s_i \models \varphi_1$

where $s_i \models \oplus\varphi$ means that φ holds in the next state to be observed in the path and $s_i \models \varphi_1 \mathcal{U} \varphi_2$ means that φ_1 holds from now and while $s_i \models \varphi_2$ holds.

Fuzzy Linear Temporal Logic As previously mentioned, fuzzy logic facilitates the definition of fuzzy statements such as "partially opened" or "completely opened". However, it does not assist in reasoning over time. Thus, we cannot conclude that a door has been opened during a period of time or until a specific time event.

Only few attempts investigated the fuzzy version of temporal modalities [17,16,36,47,30,20]. In most of the cases, fuzzy temporal logic is used to express uncertainty about the time in which some specific events may occur and the temporal relationships among events and states. In [17] the authors define the occurrence of an event as the possibility of its occurrence in any time interval and use the degree an event overlaps another one to define temporal relations. In [16] the author represents dates as a possibility distribution and use fuzzy comparators to express relations between time instants. In [36] the authors use a directed graph to define an order relation among fuzzified events and states. In [47] and [30] the authors define a fuzzy operator for each classic temporal one (e.g. always, until, etc.). This operator keeps the same semantics as its crisp counterpart. Finally, in [20] the authors extend this approach but associate a truth

degree to temporal expressions. In this paper, we follow a similar approach to [47,30,20] but with a different fuzzy interpretation of temporal operators. The new temporal semantics was defined to better fit control systems needs, reducing the events and states needed to compute temporal expressions.

Let φ be a Fuzzy Linear Temporal Logic (FLTL) formula, $\pi = s_0, s_1, \ldots, s_n$ a path, and s_i a state observed in the path π at instant i. The fuzzy interpretation $[[.]]_{s_i}$ in the state s_i of LTL operators is a mapping to $[0, 1]$ defined as follows:

$$
\begin{aligned}
&[\![\chi(\varphi)]\!]_{s_i} && \stackrel{\text{def}}{=} \chi([\![C]\!]_t)\\
&[\![\neg\varphi]\!]_{s_i} && \stackrel{\text{def}}{=} 1 - [\![\varphi]\!]_{s_i}\\
&[\![\varphi_1 \wedge \varphi_2]\!]_{s_i} && \stackrel{\text{def}}{=} min([\![\varphi_1]\!]_{s_i}, [\![\varphi_2]\!]_{s_i})\\
&[\![\varphi_1 \vee \varphi_2]\!]_{s_i} && \stackrel{\text{def}}{=} max([\![\varphi_1]\!]_{s_i}, [\![\varphi_2]\!]_{s_i})\\
&[\![\oplus\varphi]\!]_{s_i} && \stackrel{\text{def}}{=} [\![\varphi]\!]_{s_{i+1}}\\
&[\![\varphi_1 \mathcal{U} \varphi_2]\!]_{s_i} && \stackrel{\text{def}}{=} max([\![\varphi_2]\!]_{s_i}, min([\![\varphi_1]\!]_{s_i}, [\![\varphi_1 \mathcal{U} \varphi_2]\!]_{s_{i+1}}))
\end{aligned}
$$

where φ, φ_1, and φ_2 are valid formulas for the state s_i of the path π (with $i < n$, where n is the length of π), and χ is a fuzzy qualifier.

3 Related Work

FLCs have been widely used in BA, as detailed in recent reviews [48,43]. Although there is abundant literature on this subject, it is challenging to directly translate an FLC to a different domain or environment since it directly depends on expert knowledge. In fact, the design of an FLC requires more decisions than usual, e.g., regarding the rule base, inference engine, defuzzification, and data pre- and post-processing. Therefore, the simplification of this process and its adaptation to the design of context-aware controllers will improve its usability. A simple step is to use these contexts as building blocks of the control language, instead of diving in a set of parameters.

The representation of the contexts, i.e., the cases to control, is one of the contributions of this paper. A context is basically a model of humans' perceptions of the environment and is usually defined by information that is mostly subjective and inconsistent. To take advantage of this concept, in [31] authors present a study focused on the use of contexts both for the control as for the user interface (UI) in hand-held devices. Specifically, this approach uses a set of predefined contexts to design the FLC and studies how to represent these contexts in the UI. The experimental results showed that this strategy facilitates the development of the application control and, furthermore, users also considered the UI adaptation positive based on the control system. However, FLCs usually do not explicitly represent the context of context-aware systems. This is the case in [14] where thermal and visual comfort is managed by an FLC, or in [13] where authors describe the design of a living space comfort regulator using fuzzy logic. In

these papers, authors create a rule for each possible input linguistic variable combination and to all possible actuator conditions, eliminating all rules that can never apply. In general, most of the FLCs follow a similar approach [6,24] and represent the contexts implicitly as a series of fuzzy conditions on the input parameters of the system. However, the more rules and parameters, the more difficult to identify contexts.

This drawback is even more obvious when adaptive or learning strategies are applied to FLCs. In these systems, the objective is to automatically learn the fuzzy rules and/or membership functions from the data collected in a period of time. For instance, in [11], a life-long learning approach for intelligent agents that are embedded in intelligent environments is presented. In [28], a genetic algorithm is applied to shift the membership functions of the FLC properly in order to satisfy the occupants' preferences while minimizing energy consumption. In [38], authors developed a context-aware music recommendation tool that uses a Bayesian network to infer the contexts of the system. In [55], authors use particle swarm optimization (PSO) to optimize the system, where FLCs are applied to calculate the required power of the corresponding sub-systems. In [51], authors also use PSO to derive the optimal ventilation rate, where fuzzy logic is used to represent the relationship between the ventilation rate and the corresponding power consumption. The former papers, besides not providing a language to facilitate the definition of a context, have an additional drawback since there is no guarantee that the derived rule antecedents represent a context.

The complexity of the environment also has an important effect on the controller design. Some authors use multi-agent systems or hierarchical structures to organize the controller. The objective of these approaches is to simplify the specification of complex controllers with a bottom-up strategy, where controllers in the lower layers usually handle actuators while the upper-layer adapts the overall system behaviour. In [23], authors provide a self-adapting building control system with two different levels of control. In [24], authors propose a hierarchical fuzzy genetic multi-embedded-agent architecture comprising a low-level behaviour based reactive layer whose outputs are coordinated in a fuzzy way according to deliberative plans. In [52], a hierarchical architecture controls multiple local FLC agents for achieving the maximum user comfort in two different operation modes. However, the complex control structures presented in these papers have the inconvenience that conflicting objectives between different layers are difficult to solve if contexts are not clearly identified.

It should be noticed that, from the designer perspective, these hierarchical structures can be seen as a way to compose sub-contexts. Therefore, a language with the ability to compose contexts will also simplify the specification of these controllers, independently of the FLC architecture.

4 Framework Architecture

Schmidt and Van Laerhoven proposed a strategy to reduce the amount of data provided by sensors separating logical from physical processing in context-aware

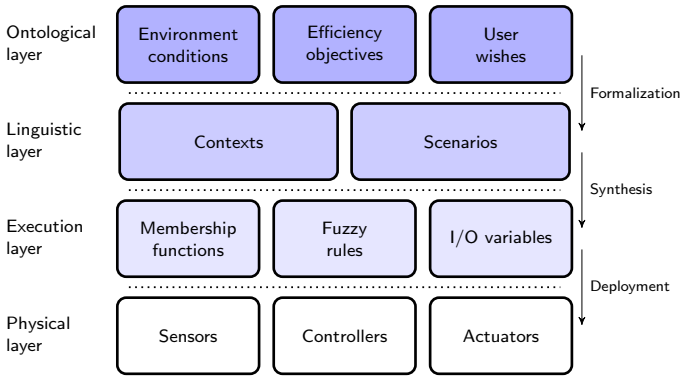


Fig. 2. Conceptual framework stack organized into layers of successive refinement. Aspects closer to the universe of discourse appear at the top, while aspects closer to the solution appear at the bottom layers.

systems [42]. This idea was later adapted by Dey et al. to propose a conceptual framework aiming at decoupling the application logic from context acquisition and derivation [8]. In this paper, we propose a framework that follows these guidelines and separates the application logic from context acquisition and inference (execution layer), enabling applications to interact with the other modules of the system through an interface.

The proposed framework stack is depicted in Fig. 2. The stack is composed of four layers, each one at a different level of abstraction, and where each layer uses the capabilities of the layers below. The top layer, namely, the *ontological layer*, is used to describe the context-aware system from the perspective of the environmental conditions, the efficiency objectives, and the user wishes. The *linguistic layer* formalizes these elements by means of the declarative language described in Section 5 and is used to define each one of the contexts and scenarios supported by the context-aware system. It is the main contribution of this paper since it provides an additional layer of abstraction to simplify the definition of context-aware fuzzy controllers. The next layer supports the execution semantics of the declarative language. In this case, each scenario is mapped to a fuzzy rule, while contexts are associated with the corresponding combination of variables and membership functions. Finally, the last layer provides support for physical devices.

It should be remarked that the proposed language can be viewed as a specialization of fuzzy logic to BA. Therefore, the architecture of the execution layer, depicted in Fig. 3, is based on the typical architecture of FLCs. This model is conceptually straightforward and consists of an input stage, a processing stage, and an output stage. The input stage, the *fuzzifier*, maps sensor, or other inputs (such as switches, thermometer, and so on), to the appropriate membership functions and truth values. In the processing stage, the *inference engine* invokes

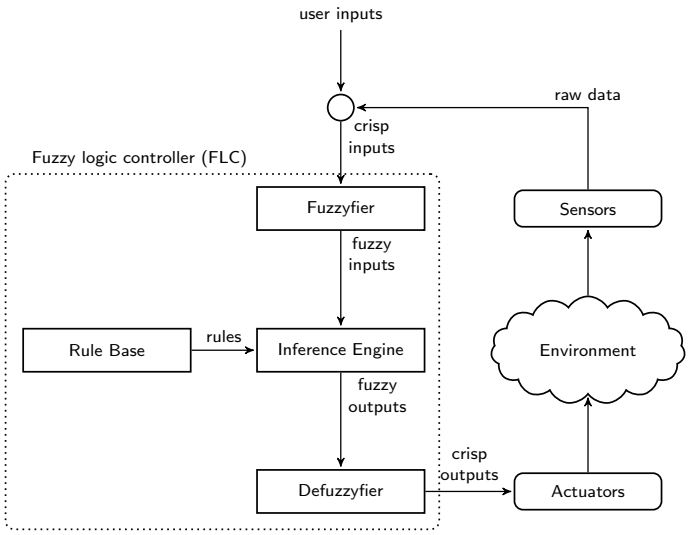


Fig. 3. Block diagram depicting the interaction of the fuzzy logic controller (left) with the environment (right).

each appropriate rule and generates the corresponding results. Then it combines the results of the rules. Finally, the output stage, the *defuzzifier*, converts the combined result back into a specific control output value.

5 Context Description Language

Ideally, BA systems should gather as much information from sensors as possible to be transparent to the user, foreseeing user needs and avoiding prompting the user for any input. In practice, however, obtaining relevant knowledge from sensors has known limitations. Hence, the context must be inferred from a multiplicity of sensors, in a process know as *data fusion* [32], which integrates multiple data and knowledge into a consistent, accurate, and useful representation.

In this section, we propose a language for the description of contexts based on fuzzy logic and temporal logic. Specifically, a context is defined by combining information from multiple inputs (sensors or other contexts) and represented by a value between 0 and 1. Since sensors and actuators are very heterogeneous, our model assumes the existence of sensor and actuator drivers that abstract the specificities of the different types of sensors and actuators. The contexts defined by this language will be used in the antecedents of the fuzzy rules and thus determine the behaviour of the control system.

5.1 Syntax

Let $c \in [0,1]$ be a constant, $f : S \to [0,1]$ a function where $s \in S$ is a sensor, $r_t(s)$ the reading of s at instant t, and let $\chi : [0,1] \to [0,1]$ be a continuous

function. A *context* C is defined recursively by the following grammar, where C_1 and C_2 are also contexts and $b \in [0, 1]$ a bias factor:

$$
\begin{array}{lll}
\mathbf{C} ::= & c & \textit{constant} \\
| & f(r_t(s)) & \textit{function} \\
| & \chi(C) & \textit{qualifier} \\
| & \neg C & \textit{complement} \\
| & C_1 \wedge C_2 & \textit{and} \\
| & C_1 \vee C_2 & \textit{or} \\
| & C_1 \wedge_b C_2 & \textit{combination} \\
| & \uparrow C & \textit{raise transition} \\
| & \downarrow C & \textit{fall transition} \\
| & \widetilde{\ominus} C & \textit{previous context} \\
| & C_1 \widetilde{\mathcal{U}} C_2 & \textit{until}
\end{array}
$$

The application of $f(r_t(s))$ represents the context resulting from the fuzzification of reading r_t of sensor s fuzzified through f, and $\chi(C)$ denotes the context obtained from the application of a fuzzy qualifier to a context. In the present work, we also use the notation "C IS χ" as syntactic sugar of $\chi(C)$. The negation operator represents the complement of a context. The logic operators '$\wedge$' and '$\vee$' translate to fuzzy equivalents. The logic operator $C_1 \wedge_b C_2$ combines two contexts with a given bias factor b between 0 and 1. The raise operator '$\uparrow C$', is a fuzzy-temporal operator that detects if the value of the parameter context C has increased. Conversely, the fall operator '$\downarrow C$' checks that the context has fallen. In the proposed grammar, the *next* temporal operator is replaced by the $\widetilde{\ominus}$ which defines a context that holds in the previous time (controllers cannot access to the future value of a context). Finally, the fuzzy until $C_1 \widetilde{\mathcal{U}} C_2$ defines a context that holds (in a fuzzy way) since the value of the first context C_1 argument holds until the value of the second one C_2 holds.

5.2 Semantics

Let C, C_1, and C_2 be contexts. The interpretation of a context formula in an instant t is a mapping $[\![.]\!]_t : C \rightarrow [0, 1]$ defined as follows:

$$
\begin{array}{lcl}
[\![c]\!]_t & \stackrel{\text{def}}{=} & \text{c} \\
[\![f(r_t(s))]\!]_t & \stackrel{\text{def}}{=} & f(r_t(s)) \\
[\![\chi(C)]\!]_t & \stackrel{\text{def}}{=} & \chi([\![C]\!]_t) \\
[\![\neg C]\!]_t & \stackrel{\text{def}}{=} & 1 - [\![C]\!]_t \\
[\![C_1 \wedge C_2]\!]_t & \stackrel{\text{def}}{=} & min([\![C_1]\!]_t, [\![C_2]\!]_t) \\
[\![C_1 \vee C_2]\!]_t & \stackrel{\text{def}}{=} & max([\![C_1]\!]_t, [\![C_2]\!]_t) \\
[\![\uparrow C]\!]_t & \stackrel{\text{def}}{=} & \begin{cases} [\![C]\!]_t - [\![C]\!]_{t-1} & \textit{if } [\![C]\!]_t - [\![C]\!]_{t-1} > 0 \\ 0 & \textit{otherwise} \end{cases} \\
[\![\downarrow C]\!]_t & \stackrel{\text{def}}{=} & \begin{cases} [\![C]\!]_{t-1} - [\![C]\!]_t & \textit{if } [\![C]\!]_{t-1} - [\![C]\!]_t > 0 \\ 0 & \textit{otherwise} \end{cases} \\
[\![\widetilde{\ominus} C]\!]_t & \stackrel{\text{def}}{=} & [\![C]\!]_{t-1} \\
[\![C_1 \widetilde{\mathcal{U}} C_2]\!]_t & \stackrel{\text{def}}{=} & \begin{cases} min(max([\![C_1]\!]_t, [\![C_1 \widetilde{\mathcal{U}} C_2]\!]_{t-1}), 1 - [\![C_2]\!]_t) & \textit{if } t > 0 \\ min([\![C_1]\!]_t, 1 - [\![C_2]\!]_t) & \textit{if } t = 0 \end{cases}
\end{array}
$$

It must be noticed that this language integrates the notion of time in the same way as in linear temporal logic. Some of the operators reason over the sequence past values of sensors and contexts to calculate its value. Therefore, given a path π of sensor readings (or past contexts), the value of a context C at instant t can be obtained recursively from C.

5.3 Fuzzy Inference Process

Each *scenario* is composed by a set of aggregated fuzzy rules that define the system's behaviour. As aforementioned, fuzzy rules have the following structure:

$$\text{IF } \textit{antecedent} \text{ THEN } \textit{consequent}.$$

where the *antecedent* part is a context. The *consequent* part is given by the following grammar:

$$\textit{consequent} ::= \chi(\textit{actuator})$$

where χ is a qualifier in the universe of discourse of the actuator. Again, the syntactic sugar "actuator IS χ" stand for χ(actuator).

The result (strength) of a rule is determined by the truth value of consequent part. Specifically, inputs are mapped to the membership function of the premise, returning a corresponding truth value. If this antecedent part is complex, the overall value of the sub-antecedents and operators is calculated. In our approach, $\wedge$, $\vee$, and $\neg$ are defined as Zadeh operators [60], that is, *and* represents the intersection or minimum between the two sets ($\mu_{A \cap B} = min[\mu_A(x), \mu_B(x)]$), *or* the union or maximum between the two sets ($\mu_{A \cup B} = max[\mu_A(x), \mu_B(x)]$), while *not* represents the opposite set ($\overline{\mu_A} = [1 - \mu_A(x)]$). Finally, each rule

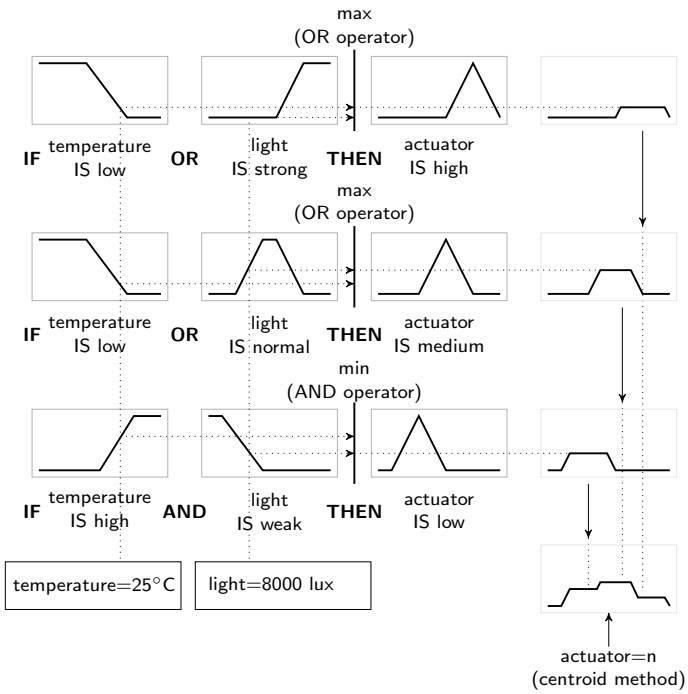


Fig. 4. Illustration of centroid defuzzification using min-max inference method for three rules relating the variables *temperature* and *luminosity* with a window actuator

strength is determined from the truth value of the antecedent part propagated to the membership function of the consequent. This mechanism is depicted in the upper part of Fig. 4. For instance, the evaluation of the first rule projects the crisp values of *temperature* and *luminosity* to their corresponding membership functions. Then the sub-antecedents are combined with the OR operator, and the max value of these antecedents is projected to the consequent membership function.

The last step is to transform the consequent parts since actuators only support crisp values. For example, Fig. 4 uses the centroid method, which returns the centre of the area under the curve. However, we must remark that the proposed language is not tied to any specific defuzzification method.

5.4 Implementation

Following a Model-Driven approach, we have implemented the language in Xtext. Xtext is a modelling workbench for textual languages. This means that by specifying the grammar of the DSL (metamodel), we are able to take advantage of Eclipse to generate an IDE dedicated to the DSL with syntax highlighting and

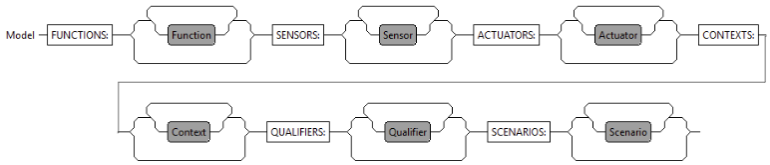


Fig. 5. Graph view of the grammar - part 1.

code auto-completion features. When the modeller/programmer is specifying its code in the DSL, the tool will check if the code is well-formed wrt the language's grammar.

Xtext is a plugin for Eclipse [45], making use of the Ecore Modelling Framework (EMF) [29], where the languages are described in terms of the metamodelling language, Ecore. This brings the advantage of interoperability between tools in the Eclipse ecosystem.

Once in a modelling framework, such as Eclipse EMF, model instances can be manipulated with specific intent, such as simulation, code generation for the controllers or even analysis of specific pertinent properties.

The specification of a language metamodel in Xtext can be done with an EBNF grammar like language, or first with an Ecore model. In Fig. 5 and Fig. 6 we present a simplified visual representation of the grammar, or modelled language, using Xtext. The complete grammar can be found in our companion site[6].

In Fig. 8, we present a snapshot of our editor with an example of a model instance of the case of the smart office presented in the next section. The use of generation technology of Xtext allows the editor to offer the common features of a modern programming environment such as syntax-highlight, syntax validation and suggestions, auto-completion. This set of features are usually seen as usability enhancers for textual languages.

As aforementioned, our approach adds time modalities to fuzzy logic. However, these modalities are implemented using traditional fuzzy operators, and thus can be interpreted directly by a fuzzy logic engine. Specifically, we followed a Model-Driven approach that consists of *(i)* the definition of Model-to-model direct and simple transformation rules, using ATL[7] transformation language, and *(ii)* Model-to-code generation, using Xpand template based approach[8].

6 Validation

This work is validated through a Building Automation case study of a smart office. The setup is a simplified example that conjoins the main aspects of Building Automation [12] and Home Automation [18] preserving feature-richness while

[6] `https://gitlab.citius.usc.es/juan.vidal/fuzzy-temporal-dsl`
[7] `https://www.eclipse.org/atl/`
[8] `https://www.eclipse.org/modeling/m2t/?project=xpand`

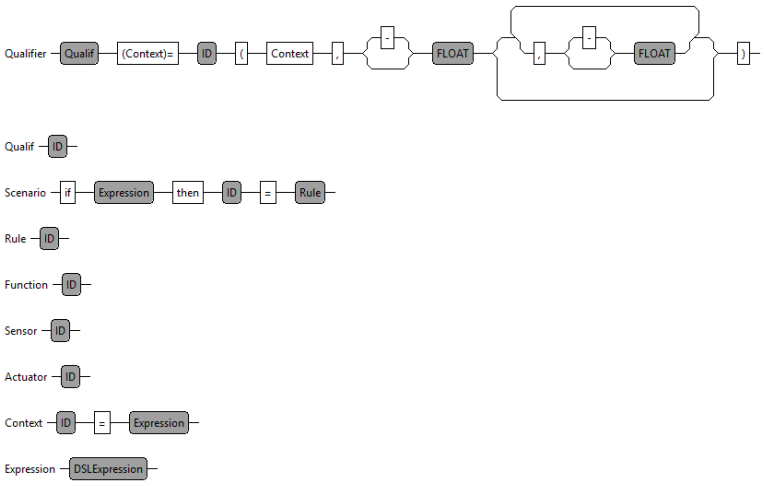


Fig. 6. Graph view of the grammar - part 2.

avoiding tedious details of the actual setup developed in the scope of the Smart Campus project [22][9]

We start by defining the activity contexts using the syntax defined in Section 5.1 and corresponding scenarios using fuzzy inference rules. This example scenario highlights the language's expressiveness and the adequacy of its operators. The specification will test the capability of a system to react correctly to variations of the environment inputs to drive the actuators. The results are obtained using an emulator of the language semantics.

6.1 Case Study

Consider the smart office BA setup depicted in Fig. 9, with a desk, a support table used for meetings, and the following sensors:

- S_{door}: A door sensor to automatically detect the people as they enter or leave the room.
- S_{table}: A pressure sensor to detect activity at the table.
- S_{desk}: A pressure sensor to detect activity at the desk.
- S_{win}: Luminosity sensors inside the room, near the window.
- S_{wall}: Luminosity sensors inside the room, near the inner wall.
- S_{ext}: A luminosity sensor placed in the exterior.
- S_{time}: A virtual time sensor to allow reasoning about time.

And three actuators:

[9] `https://cordis.europa.eu/project/rcn/191915/factsheet/en`

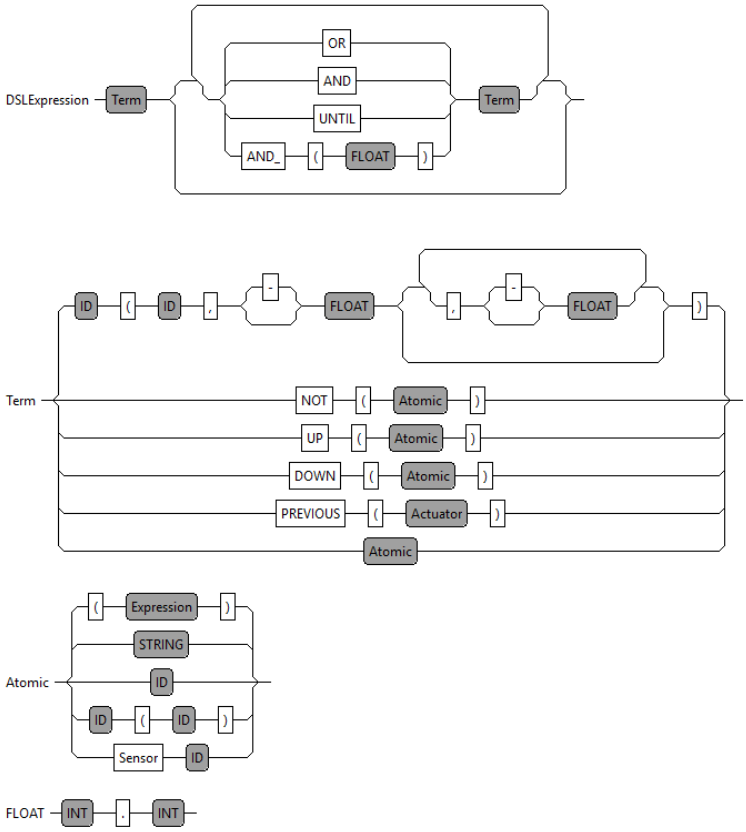


Fig. 7. Graph view of the grammar - part 3.

- A_{blind}: The blind curtains, used to control the amount of natural light.
- A_{lum_1}: A luminary close to the window.
- A_{lum_2}: A luminary near the inner wall.

The objective is to design an intelligent controller that adjusts the light level in the room to balance the comfort following the occupants needs and, whenever possible, save energy by dimming the lamps or switching them off when the occupant is not in the room. To simplify the example, we will only describe one of the control scenarios and only consider one person in the room.

6.2 Luminosity contexts

Luminosity sensors have their interval of response values. For example, an exterior luminosity sensor may measure a value much higher than a sensor placed inside the office. However, for the user, the perception inside the office may be

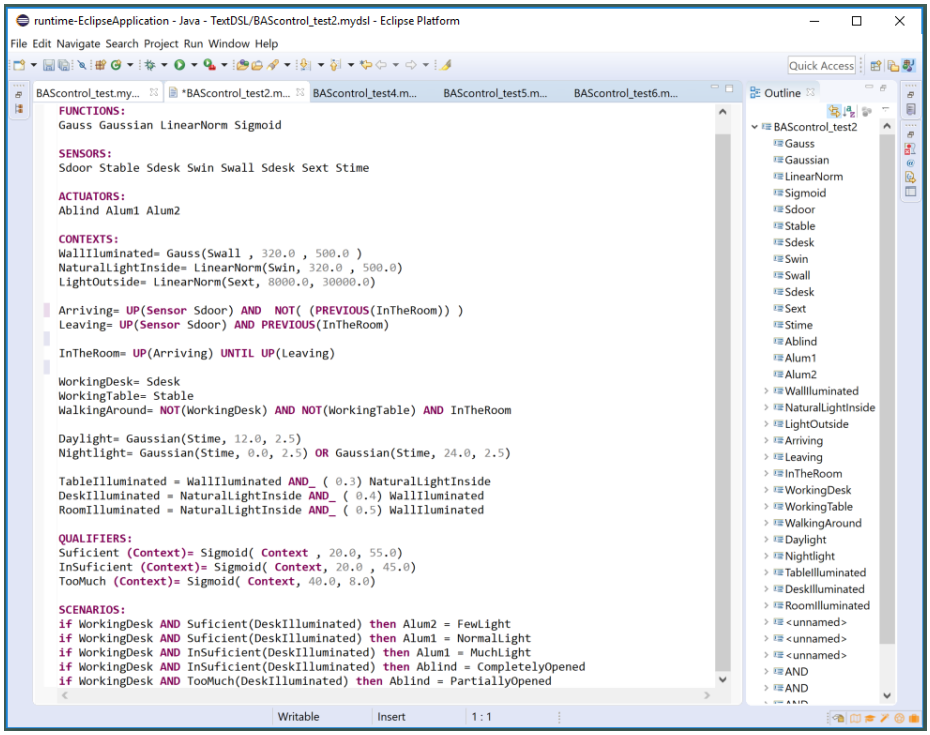


Fig. 8. Generated Xtext editor with example related to smart office.

much higher than outside. This happens because the perception of the natural light by humans and sensors is different since the latter is also sensitive to other wavelengths. Therefore a low-level context that normalizes these values is required. Specifically, we created three contexts: *WallIlluminated* which represents the amount of light near the wall, *NaturalLightInside* which indicates the natural light inside, and *LightOutside* which captures the amount of light outside:

$$\begin{aligned} WallIlluminated &\equiv Trapezoid(S_{wall}, 320, 500, 2000, 2500) \\ NaturalLightInside &\equiv Trapezoid(S_{win}, 320, 500, 2000, 2500) \\ LightOutside &\equiv Trapezoid(S_{ext}, 800, 3000, 5000, 7200) \end{aligned} \tag{1}$$

In this case, we used a linear normalization function to convert the value of the luminosity sensors to a normalized value between 0 and 1. For each luminosity sensor, a minimum and maximum value have been specified where the minimum value represents the degree of 0 of the context while the maximum corresponds to the degree of 1. Please note that the constants used in the normalization

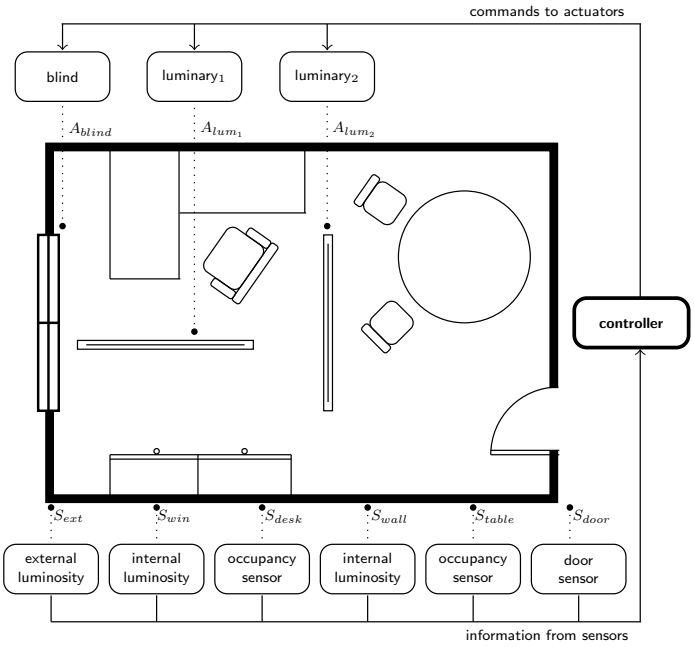


Fig. 9. Illustration of the smart office setup. The office is equipped with six sensors and three actuators orchestrated by a controller.

functions are particular of our case-study and are based on sensors characteristics and the physical place where they are positioned.

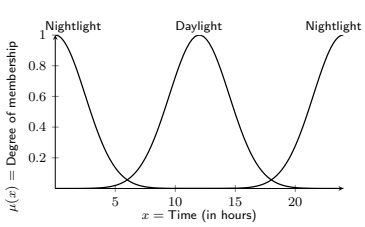
Low-level contexts can also be used to define some models, i.e. predictions of certain parameters to adjust the behaviour of the system according to them. In this case study, we have defined two models that predict if there is daylight or night at a certain point in time. *Daylight* and the *Nightlight* are represented in Fig. 10(a) which uses a *Gaussian function* to fuzzify the timeline:

$$\begin{aligned} Daylight &\equiv Gaussian(S_{time}, 12, 2.5) \\ Nightlight &\equiv Gaussian(S_{time}, 0, 2.5) \vee Gaussian(S_{time}, 24, 2.5) \end{aligned}$$

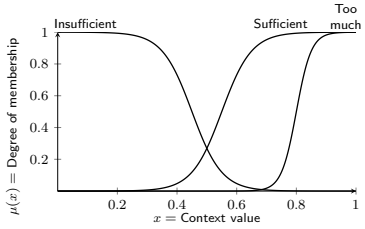
As we can see in the following definitions, *Day* and *Night* high-level contexts are defined from *Daylight* and *Nightlight* contexts:

$$\begin{aligned} Day &\equiv Daylight \ \wedge_{0.7} \ LightOutside \\ Night &\equiv Nightlight \ \wedge_{0.6} \ \neg LightOutside \end{aligned}$$

We have also defined another high-level luminosity contexts to predict the amount of luminosity in a given place inside the room. Like in the definition of *Day* and *Night*, the values of the low-level luminosity contexts must be weighted, in this case, to compensate for their respective placement:

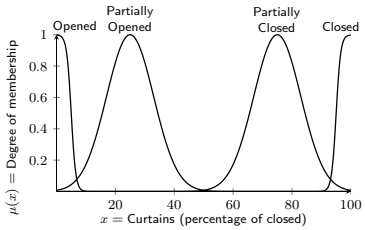


(a) Light levels during day and night

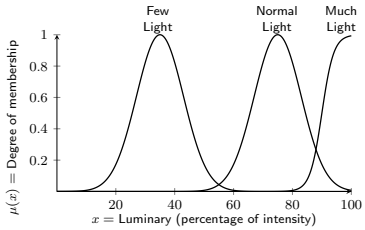


(b) Insufficient, Sufficient, and TooMuch context operators

Fig. 10. Examples of membership functions used to (a) fuzzyfy the light measured by external luminary sensors and (b) to better characterize the light intensity.



(a) Linguistic values defined for the curtains actuator



(b) Linguistic values defined for the luminaries actuator

Fig. 11. Examples of membership functions considered during the development of the prototype to illustrate the fuzzy values that (a) blind and (b) internal luminary actuators may take. These values are used in rules (1) to (5) of the described scenario.

$$
\begin{aligned}
TableIlluminated &\equiv WallIlluminated \ \wedge_{0.3} \ NaturalLightInside \\
DeskIlluminated &\equiv NaturalLightInside \ \wedge_{0.4} \ WallIlluminated \\
RoomIlluminated &\equiv NaturalLightInside \ \wedge_{0.5} \ WallIlluminated
\end{aligned}
$$

Finally, to qualify the degree of light we have defined three different fuzzy context operators (see Fig. 10(b)):

$$
\begin{aligned}
\mathit{Sufficient}(C) &= Sigmoid(C, 20, 0.55) \\
\mathit{InSufficient}(C) &= Sigmoid(C, -20, 0.45) \\
\mathit{TooMuch}(C) &= Sigmoid(C, 40, 0.8)
\end{aligned}
$$

In this case study, we opted to choose the same definition of these operators for all the contexts, although this is not usual in fuzzy control.

6.3 Activity contexts

Three contexts determine when someone enters, leaves, or is in the room. Contrary to the luminosity contexts, here time is especially important. For instance, the contexts *Arriving* and *Leaving* hold when the door has been opened. However, more complex situations, like to determine if someone is in the office, are challenging to establish without the use of temporal logic. Precisely, this behaviour has been implemented applying the operator until to the contexts *Arriving* and *Leaving*:

$$\begin{aligned} Arriving &\equiv \uparrow S_{Door} \ \wedge \ \neg(\widetilde{\ominus}\, InTheRoom) \\ Leaving &\equiv \uparrow S_{Door} \ \wedge \ \widetilde{\ominus}\, InTheRoom \\ InTheRoom &\equiv \uparrow Arriving \ \widetilde{\mathcal{U}} \ \uparrow Leaving \end{aligned}$$

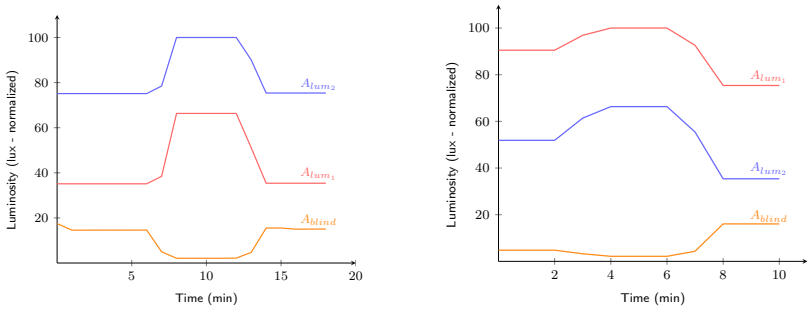
We have also defined three different contexts to detect the activity inside the office room. Contexts *WorkingDesk* and *WorkingTable* are a direct mapping of the motion sensor placed near the support table and the presence sensor in the desk chair, respectively, since both sensors returned a normalized value between 0 and 1. The third context captures the case in which the user is neither located at the desk nor at the table:

$$\begin{aligned} WorkingDesk &\equiv S_{Desk} \\ WorkingTable &\equiv S_{Table} \\ WalkingAround &\equiv \neg WorkingDesk \ \wedge \ \neg WorkingTable \ \wedge \\ &\quad \wedge \ InTheRoom \end{aligned}$$

6.4 Scenarios

A set of fuzzy rules specifies the behaviour of the system and how it will react according to the contexts present in the system. In this work, we defined six different scenarios to model the behaviour of the system when someone *(i)* enters or *(ii)* leaves the office, when the user is *(iii)* working at the desk or *(iv)* at table, *(v)* when he is walking around, or when he needs some privacy. For reasons of space, we will detail the rules that implement the fourth scenario:

$$\begin{aligned} &\textbf{if } WorkingDesk \ \wedge \ Sufficient(DeskIlluminated) \\ &\quad \textbf{then } A_{lum_2} = FewLight \qquad (2) \\ &\textbf{if } WorkingDesk \ \wedge \ Sufficient(DeskIlluminated) \\ &\quad \textbf{then } A_{lum_1} = NormalLight \qquad (3) \\ &\textbf{if } WorkingDesk \ \wedge \ Insufficient(DeskIlluminated) \\ &\quad \textbf{then } A_{lum_1} = MuchLight \qquad (4) \\ &\textbf{if } WorkingDesk \ \wedge \ Insufficient(DeskIlluminated) \\ &\quad \textbf{then } A_{blind} = CompletelyOpened \qquad (5) \\ &\textbf{if } WorkingDesk \ \wedge \ TooMuch(DeskIlluminated) \\ &\quad \textbf{then } A_{blind} = PartiallyOpened \qquad (6) \end{aligned}$$



(a) Cloud in the sky (b) Working at desk

Fig. 12. The behaviour of A_{blind}, A_{lum_1}, and A_{lum_2} actuators when (a) the cloud pass and thus reduce the light in the office, and (b) when the person in the room starts working at the desk and then leaves the office.

Rules (2) and (3) specify the behaviour of the luminaries when there is sufficient light at the desk which will set the actuator A_{lum_2} to a low level of light and the actuator A_{lum_1} to a normal level of light. The rules (4) and (5) control the behaviour of the actuators A_{lum_1} and A_{blind} when there is insufficient light at the desk. In this case, A_{lum_1} should be set to a high and A_{blind} should be completely opened to allow the maximum amount of natural light inside. Finally, rule (6) adjusts the curtains to avoid the excessive glare at the desk. The fuzzy operators applied to the curtains and luminaries actuators are depicted in Fig. 11(a) and Fig. 11(b), respectively.

6.5 Discussion

The previously described scenarios have been implemented and validated. Specifically, for each scenario, we performed a ten-fold test validation to verify the robustness of the rules. For instance, plots of Fig. 12 show the actuation of the FLC in two different situations. In the first scenario, Fig. 12(a) represents how the luminosity actuators react when some clouds appear in the sky. In this case, the clouds reduce the amount of light inside the room triggering an increase of the two luminaries inside the room, A_{lum_1} and A_{lum_2}, at instant $t = 5$. Notice also that the curtains A_{blind} are also completely opened during this event to take advantage of the maximum amount of light from the outside. Another example is depicted in Fig. 12(b), which represents the behaviour of the actuators when a person is working at the desk. Specifically, the FLC detects insufficient light at instant $t = 2$ (clouds are passing) and thus increases the luminosity of its three actuators. This situation continues until $t = 8$ when the light intensity increases by resulting in too much light. At this moment, the FLC starts reducing light intensity provided by the actuators.

7 Usability Evaluation

We performed a quasi-experiment for comparing how practitioners can use the proposed DSL in this paper. As comparison baseline, we use Simulink, from MathWorks[10], a graphical programming environment for modelling, simulating and analyzing multi-domain dynamical systems widely used for automatic control. We compared both languages with an experiment where the subjects had to solve similar modelling challenges.

The experiment focused on the extent to which the different language constructs with fuzzy logic and temporal logic operators impact on the developer experience. These tasks were monitored so that we could assess their success and effort involved (measured in terms of time taken to complete). For the sake of space, we make available the material used (questionnaires, slides, questions, and other) and raw data in a companion website `https://gitlab.citius.usc.es/juan.vidal/fuzzy-temporal-dsl`.

7.1 Goals

Generally, we are interested in assessing the usability and effectiveness of our proposed DSL and Simulink in the context of specifying automated control challenges. We can describe our research goals using the Goal-Question-Metric [3] template:

Our main goal (**G1**) is to *analyze* the effect of using the proposed DSL, *for the purpose of* evaluation, *with respect to* the **correctness** with which a developer specifies the control logic, *from the viewpoint of* researchers, *in the context of* an experiment conducted at IST (Universidade de Lisboa) and Universidad Santiago de Compostela.

The following questions that refine the goal to a more quantifiable way, regarding the usability of the proposed DSL, can be described as:

- (**Q1**) is the language easy to interpret?
- (**Q2**) is the language error prone?
- (**Q3**) are the language specifications easy to complete?
- (**Q4**) are the language specifications easy to evolve?
- (**Q4**) have the users a positive attitude towards the language?

The independent variables are the Language type and the Question Type: **interpretation**, where the user has to explain the semantics that can be perceived from a given sentence (measured as if it is correct or incorrect); **error identification**, like misplaced elements in the sentence (number of errors); **complete**, where an incomplete sentence is provided, and the subject must fill in the missing information for a particular purpose (completes everything or not); and **evolve**, where a correct sentence is provided, but the subject is challenged to rework it for a new purpose (measured with the grade "correct" or "not"). The dependent variables are the answers correction rate and the System Usability Scale (SUS)

[10] http://mathworks.com

score [5]. SUS is a simple test with ten questions regarding attitude with a Likert Scale (five possible answers ranging from Strongly agree to Strongly disagree) that gives a global view of subjective assessments of usability.

For each of our high-level goals, we define the null (H_0) and alternative (H_1) hypotheses. Similar hypotheses can be written for contrasting the *DSL* with *Simulink* in terms of their effect on **correctness** and perceived **usability** of the languages.

> $H_{0Correctness}$: Using *DSL* rather than *Simulink* does not influence the successful completion of the task **correctness**.
>
> $H_{1Correctness}$: Using *DSL* rather than *Simulink* the successful completion of the task **correctness**.
>
> $H_{0Usability}$: Using *DSL* rather than *Simulink* does not influence the perceived **usability** of the use of the language.
>
> $H_{1Usability}$: Using *DSL* rather than *Simulink* influences the perceived **usability** of the use of the language.

How to assess correctness? The proposed eight challenges have a "gold standard" solution defined in both languages, with which we compare the answers provided by our participants. The correctness is measured in terms of their *precision* as follows: the percentage of model elements and relationships in the model built by the participant that correctly address the challenge (even if the participant chose alternative ways to answer compared to the *"gold standard"*, as long as they are considered correct.

High values of *precision* support the claim for higher correctness, with 0% representing totally incorrect and 100% totally correct models.

How to assess the perceived usability? We assess the *perceived usability* through a SUS questionnaire which provides a SUS score from 0 to 100, with an average value of 68 [5]. Higher values support the claim for better usability.

7.2 Subjects

The experiment was replicated in two sites: IST (Universidade de Lisboa), and Universidad de Santiago de Compostela. The ten selected participants in this study were aged between 19 and 47 years old researchers (three postdocs, one PhD student and two MSc) with a background ranging from Physics, Electronics, and Software Engineers, and a group of BSc. students in Telecommunications and Computer engineering. Two subjects declared to have had previous experience with Simulink, where one declared experience with BA Systems or alike, and four (including those two) declared to have used modelling and simulation tools for control systems specification. Initially the group of subjects was composed by fifteen elements, however, during the execution of the experiment, we realized that 5 of them were constrained in the time, which increased the stress level and

lead to incomplete answers. Therefore the data related to these elements were not considered; however, it is detailed in the raw data available at our companion site `https://gitlab.citius.usc.es/juan.vidal/fuzzy-temporal-dsl`.

We used convenience sampling to recruit our participants on both sites. Each participant was randomly assigned to one of two groups (called A and B), keeping a balanced sample on each of the two groups. The purpose of this separation was to remove a possible bias of using the languages in just one particular order. This way, group A will perform the tasks of one language an then the other, while group B will do the reverse.

7.3 Process

We have designed the experiment in the following way. Before starting, each participant signed a letter of consent and filled in a questionnaire, so that we record information about our participants' age, academic level, previous experience with Control engineering and, in particular, with each of the two analyzed languages. Following that, a short tutorial on both languages was presented. The subjects were grouped randomly in two groups A and B. The experiment proceeded in pen and paper. After these preparatory tasks, the experiment itself started. During the whole session, the time taken to answer each question was logged. Furthermore, the participants had no time limit to finish their tasks, but our pilot sessions pointed to a duration of no more than 20 minutes to perform the given set of eight tasks.

Finally, the subjects answered to a SUS test, so that we could contrast their opinion on the usability of our proposed DSL and of Simulink.

7.4 Tasks

As mentioned before the chosen research instrument to assess the tasks performed by the subjects was a test in pen and paper with free response (to remove any usability interference of the IDE or tooling environments). We have inserted two questions per following groups: i) Reading/Interpretation Question; ii) Identify Errors; iii) Challenges - use language to solve question; and, iv) Evolution/change existing expression. The aim was to look at the correctness, that we regard as effectiveness, of using both languages in these mentioned tasks. All eight questions involved the concepts of fuzzy classifiers and fuzzy-temporal operators.

The learning material was available to the subjects at all time, in case any question regarding the syntax of both languages could arise.

An example of the Simulink models used in this usability study is depicted in Fig. 13[11]. Specifically, this model represents the context *inTheRoom* previously described in Section 6.3, thus only a part of a controller. This context is defined with the "until" operator and holds from the moment someone comes in the

[11] More models, including complete controllers are available in the experiments memory at `https://gitlab.citius.usc.es/juan.vidal/fuzzy-temporal-dsl`

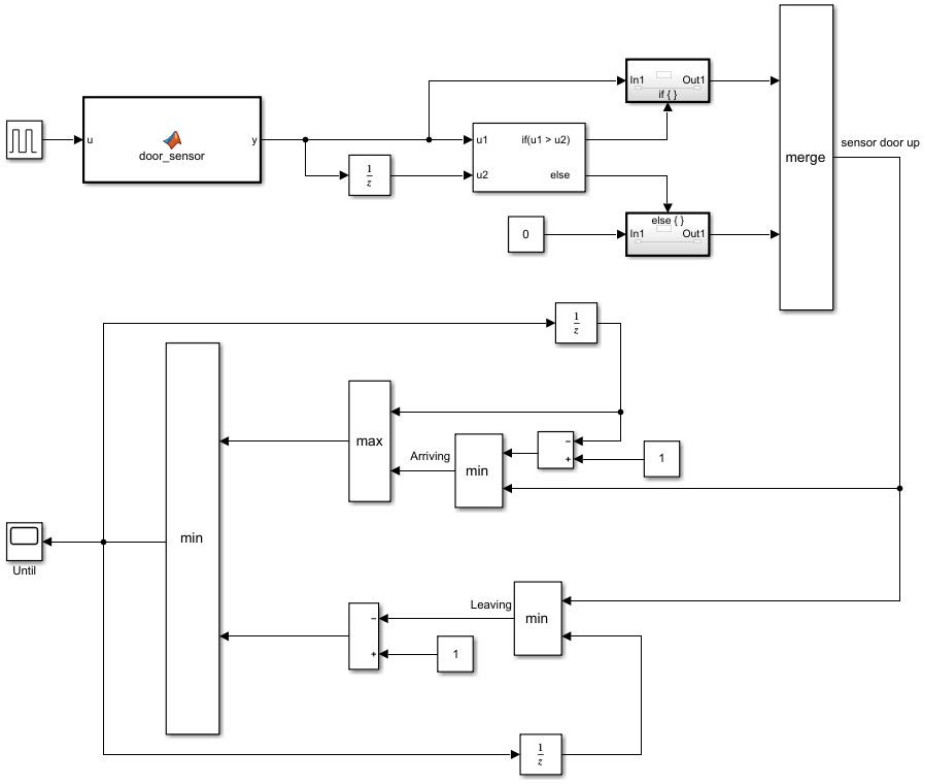


Fig. 13. Representation in Simulink of the context *inTheRoom* defined in Section 6.3

room until he comes out. As can be seen, the model is quite complex and has two different parts. The upper part models the *raise* operator identifying, by means of an *if-then-else* block, when the signal that comes from the door sensor increases. Thus, this block receives the signal, and the signal delayed one-time unit (z^{-1} block). This result is merged and constitutes one of the inputs of the lower part of the diagram. This part of the model defines the *Arriving* context and the *Leaving* context, and then use their signals as the input of the several blocks that are combined to simulate the *until* operator.

7.5 Discussion

The data collected during the experiment sessions was analyzed using manual data collection. Concerning the descriptive statistics, we collected the following ones, adjusting the actual set of descriptive statistics to the scale type (ordinal, and ratio) of each variable: the number of cases, mean, median, mode, standard deviation, skewness, kurtosis, the *p-value* of the Shapiro-Wilk normality test. After this, we used the Welch t-test (which is a more robust alternative to the

t-test [53]) to compare the distributions of correctness and usability obtained with the proposed DSL *vs.* Simulink.

Correctness The data concerning *correctness* was collected through inspection of the solutions produced by the participants during our study. This implied a qualitative assessment of the solutions in a similar process to grading, in an academic context. We have then computed descriptive statistics, as in table 1, for the collected metrics as summarized in table 3 and test for significant differences between the level of correctness achieved with each language.

Perceived usability As planned, we assessed usability through a SUS test. The SUS instrument was available in a pen and paper questionnaire. The result of the comparative analysis of the distributions of the usability scores is summarized in Table1.

Table 1. Descriptive statistics

	Language	N	Mean	Std. Deviation	Skewness	Kurtosis	Shapiro-Wilk
Corr.	DSL	10	82.50	10.90	0.38	0.32	0.60
Corr.	Simulink	10	45.00	15.81	1.32	2.47	0.22
SUS	DSL	10	75.25	14.00	-0.90	0.38	0.46
SUS	Simulink	10	31.00	8.00	0.05	-0.05	0.99

As observed in Table 1, the SUS score is higher in the proposed DSL in comparison to Simulink. Besides, knowing that [5] if the score is over 68, then the probability of having satisfaction concerning the usability of the language is high.

Our participants, except for two cases, could not complete all the requested tasks in Simulink. On the other hand, the complete opposite situation occurs for the proposed DSL, where we could observe that there is only one case of quitting one particular task.

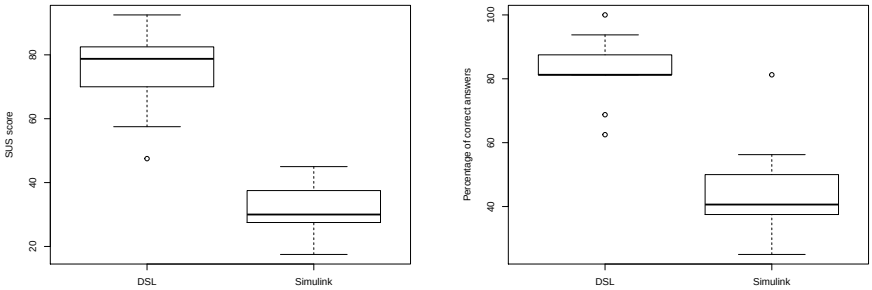
Table 2. Welch's t-test scores

	DSL mean	Simulink mean	Difference	95% Dif. CI Lower	95% Dif. CI Upper	t	df	p-value
Corr.	0.825	0.45	37.50	0.25	0.50	6.17	16	0.00
SUS	0.820	0.45	44.25	33.30	55.20	8.67	14	0.00

From Table 2, the two-tailed P value is less than 0.0001, which, by conventional criteria, is considered to be extremely statistically significant. Therefore,

Table 3. Results of usability tests

	Correct Answer (%)	
Question	DSL	Simulink
1. Describe in words the presented context rules.	90	60
2. Mark code expressing case.	100	65
3. Identify errors in the model, first case.	45	25
4. Identify errors in the code, second case.	70	20
5. Complete the model, first case.	100	65
6. Complete the model, first case.	100	75
7. Evolve existing expression, first case.	65	25
8. Evolve existing expression, second case.	90	25



(a) SUS distribution (b) Rate distribution of correct answers

Fig. 14. The DSL performs better in terms of usability and correct answer rate than Simulink.

it can be confirmed the correctness of the DSL over Simulink as well as of its Usability.

A possible explanation for the obtained SUS and correct answer rate distributions depicted in Fig. 14(a) and Fig. 14(b) is the likelihood that Simulink needs much more time for training than the DSL that has domain notations more close to the way of thinking of the user being, therefore, self-descriptive. Further experiments should be done to confirms this fact.

7.6 Threats to validity

As in any empirical study, it is common to identify potential validity threats [54]. In this case, we consider that the population selection is a threat, as, due to resource constraints, all the participants of the usability experiments were members of the academy. Further research is required to access the DSL with practitioners in control engineering. Also, due to the reduced availability of the

participants, the experiments were reduced to eight tasks of varying complexity. While those tasks were selected for being representative of the four categories, it would be interesting to increment the external validity with replications of the evaluation presented here.

8 Conclusions and Future Work

Specifying the behaviour of a BAS traditionally requires a great deal of technical knowledge, mainly when the output depends continuously on the inputs. Besides, behaviour specifications are usually embedded in the hardware of proprietary systems, forcing developers to use the manufacturer's tools and languages. Moreover, these languages and tools do not often take into account the continuous actuation according to the sensor inputs. To tackle these issues, we proposed a high-level declarative language that enables the automatic generation of BA controllers. Our language draws on aspects of fuzzy logic and temporal logic to specify each component of the controller.

The proposed language is used to define each of the components of a BAS controller (*Sensors*, *Contexts*, *Scenarios* and *Actuators*). The *Sensors* and *Actuators* are an abstraction of the corresponding physical devices, which are used in the specification of the behaviour. The *Contexts* component represents an abstraction of a high-level notion defined by the end-user, while the *Scenarios* component aggregates a set of related rules that will define the value to be set on the actuators in run-time.

We envisioned a high-level specification of the behaviour of the system through a straightforward way of specifying fuzzy contexts and fuzzy rules that associate scenarios with contexts. An essential aspect of our approach is enabling reasoning about past events. To that end, we defined the semantics of the temporal logic *Until* operator to fuzzy propositions. The fuzzy logic allows the user to specify contexts and rules in a high-level form as well as to deal with the continuous variation of the actuation according to the input sensors, while the usage of the temporal logic permits the reasoning about temporal aspects.

Finally, as proof of concept, we illustrate our framework and language with a real case scenario that highlights distinct aspects of our proposal. As described in this practical validation, the proposed language simplifies the development of BA controllers. In fact, this language provides a higher level of abstraction to developers that can define their controllers in terms of contexts and scenarios instead of using the control constructs of a programming language. These results validate the adequacy of fuzzy logic as an inference mechanism similar to human reasoning and thus that models are easier to understand and update. To complement our validation, we held an experiment with subjects that confirms that users of our language could use it with effectiveness (regarding correctness in tasks such as interpretation, identification of errors, problem-solving and evolution) and usability. As a comparison baseline, we used Simulink.

As future work, despite the positive results concerning usability, as in any DSL life-cycle, languages evolve. In fact, it is typical to incorporate extensions

after the user's feedback in an iterative fashion. In this respect, our proposed DSL would benefit from its usage with new real-world case studies, perhaps outside the scope of BA. As already mentioned before, it will be interesting to hold more experimental studies with more subjects. In addition, we also identified a number of possible expressiveness improvements such as *(i)* the introduction of a structural quantifier to enable reasoning of structural aspects of the facility and the relative positioning of sensors, and (ii) encapsulation features to support larger scale specifications.

9 Acknowledgements

INESC-ID authors were supported by national funds through FCT (Fundação para a Ciência e a Tecnologia) under contract UID/CEC/50021/2019. The authors would like to thank the COST Action IC1404 Multi-Paradigm Modeling for Cyber-Physical Systems (MPM4CPS) for the context and partial support to this work, as well as NOVA LINCS Research Laboratory (Grant: FCT/MCTES PEst UID/ CEC/04516/2013) and DSML4MAS Project (Grant: FCT/MCTES TUBITAK/0008/2014) "Modelação de Sistemas Sócio Ciberfísicos" FCT/DAAD - 2018/2019 (Poc. DAAD 441.00). IDMEC author was supported by FCT project UID/EMS/50022/2019. The CiTIUS author was also supported by the Spanish Ministry of Economy and Competitiveness under the project TIN2015-73566-JIN and by the Consellería de Cultura, Educación e Ordenación Universitaria (accreditation 2016-2019, ED431G/08 and reference competitive group 2019-2021, ED431C 2018/29) and the European Regional Development Fund (ERDF).

References

1. Álvarez, J., Redondo, J., Camponogara, E., Normey-Rico, J., Berenguel, M., Ortigosa, P.: Optimizing building comfort temperature regulation via model predictive control. Energy and Buildings **57**(0), 361 – 372 (2013)
2. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. International Journal of Ad Hoc and Ubiquitous Computing **2**(4), 263–277 (2007)
3. Basili, V., Caldiera, G., Rombach, H.: Goal Question Metric Paradigm. Encyclopedia of Software Eng. **1**, 528–532 (2001)
4. Boithias, F., El Mankibi, M., Michel, P.: Genetic algorithms based optimization of artificial neural network architecture for buildings indoor discomfort and energy consumption prediction. Building Simulation **5**(2), 95–106 (2012)
5. Brooke, J.: Sus-a quick and dirty usability scale. In: Jordan, P.W., Thomas, B., Weerdmeester, B.A., McClelland, I.L. (eds.) Usability evaluation in industry, chap. 21, pp. 189–194. Taylor & Francis, London (1996)
6. Calvino, F., Gennusa, M.L., Rizzo, G., Scaccianoce, G.: The control of indoor thermal comfort conditions: introducing a fuzzy adaptive controller. Energy and Buildings **36**(2), 97 – 102 (2004)
7. CEN: En 15232-2006, calculation methods for energy efficiency improvements by the application of integrated building automation systems. Tech. rep., European Committee for Standardization, Brussels, Belgium (2006)

8. Dey, A., Abowd, G., Salber, D.: A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-Computer Interaction **16**(2), 97–166 (2001)
9. Dey, A.K., Abowd, G.D.: Towards a better understanding of context and context-awareness. In: In HUC'99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing. pp. 304–307. Springer-Verlag (1999)
10. Dixon, R.K., McGowan, E., Onysko, G., M. Scheer, R.: US energy conservation and efficiency policies: Challenges and opportunities. Energy Policy **38**(11), 6398–6408 (2010)
11. Doctor, F., Hagras, H., Callaghan, V.: A fuzzy embedded agent-based approach for realizing ambient intelligence in intelligent inhabited environments. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on **35**(1), 55–65 (Jan 2005)
12. Domingues, P., Carreira, P., Vieira, R., Kastner, W.: Building automation: Concepts and technology review. Computer Standards and Interfaces Journal **45**, 1–12 (March 2016)
13. Dounis, A., Manolakis, D.: Design of a fuzzy system for living space thermal-comfort regulation. Applied Energy **69**(2), 119–144 (2001)
14. Dounis, A., Santamouris, M., Lefas, C., Argiriou, A.: Design of a fuzzy set environment comfort system. Energy and Buildings **22**(1), 81 – 87 (1995)
15. Driankov, D., Hellendoorn, H., Reinfrank, M.: An Introduction to Fuzzy Control. Springer-Verlag New York, Inc., New York, NY, USA (1993)
16. DuBois, D., Prade, H.: Processing fuzzy temporal knowledge. IEEE Transactions on Systems, Man, and Cybernetics **19**(4), 729–744 (July 1989)
17. Dutta, S.: An event based fuzzy temporal logic. In: [1988] Proceedings. The Eighteenth International Symposium on Multiple-Valued Logic. pp. 64–71 (1988)
18. Ejnar Rovsing, P., Larsen, P., Skjdeberg Toftegaard, T., Lux, D.: A reality check on home automation technologies. Journal of Green Engineering **1** (04 2011)
19. Ferreira, P., Ruano, A., Silva, S., Conceio, E.: Neural networks based predictive control for thermal comfort and energy savings in public buildings. Energy and Buildings **55**(0), 238 – 251 (2012), cool Roofs, Cool Pavements, Cool Cities, and Cool World
20. Frigeri, A., Pasquale, L., Spoletini, P.: Fuzzy time in linear temporal logic. ACM Trans. Comput. Logic **15**(4), 30:1–30:22 (Aug 2014)
21. Fukamios, S., Mizumoto, M., Tanaka, K.: Some considerations on fuzzy conditional inference. Fuzzy Sets and Systems **4**(3), 243–273 (1980)
22. Gomes, R., Pombeiro, H., Silva, C., Carreira, P., Carvalho, M., Almeida, G., Domingues, P., Ferrão, P.: Towards a Smart Campus: Building-User Learning Interaction for Energy Efficiency, the Lisbon Case Study, pp. 381–398. World Sustainability Series book series, Springer International Publishing (November 2017)
23. Guillemin, A., Molteni, S.: An energy-efficient controller for shading devices self-adapting to the user wishes. Building and Environment **37**(11), 1091–1097 (2002)
24. Hagras, H., Callaghan, V., Colley, M., Clarke, G., Pounds-Cornish, A., Duman, H.: Creating an ambient-intelligence environment using embedded agents. Intelligent Systems, IEEE **19**(6), 12–20 (Nov 2004)
25. Jang, J., Sun, C., Mizutani, E.: Neuro-fuzzy and soft computing-a computational approach to learning and machine intelligence. Automatic Control, IEEE Transactions on **42**(10), 1482–1484 (1997)
26. Jia, R., Jin, B., Jin, M., Zhou, Y., Konstantakopoulos, I.C., Zou, H., Kim, J., Li, D., Gu, W., Arghandeh, R., Nuzzo, P., Schiavon, S.,

Sangiovanni-Vincentelli, A.L., Spanos, C.J.: Design automation for smart building systems. Proceedings of the IEEE **106**(9), 1680–1699 (Sep 2018). https://doi.org/10.1109/JPROC.2018.2856932
27. Johnson-Laird, P.N.: Mental models and human reasoning. Proceedings of the National Academy of Sciences of the United States of America **107**(43), 18243–18250 (Oct 2010)
28. Kolokotsa, D., Stavrakakis, G., Kalaitzakis, K., Agoris, D.: Genetic algorithms optimized fuzzy controller for the indoor environmental management in buildings implemented using PLC and local operating networks. Engineering Applications of Artificial Intelligence **15**(5), 417 – 428 (2002)
29. Kolovos, D.S., Rose, L.M., Abid, S.B., Paige, R.F., Polack, F.A., Botterweck, G.: Taming EMF and GMF Using Model Transformation. In: Petriu, D.C., Rouquette, N., Haugen, Ø. (eds.) Model Driven Engineering Languages and Systems SE - 15, Lecture Notes in Computer Science, vol. 6394, pp. 211–225. Springer Berlin Heidelberg (2010). https://doi.org/10.1007/978-3-642-16145-2_15, http://dx.doi.org/10.1007/978-3-642-16145-2_15
30. Lamine, K.B., Kabanza, F.: Using fuzzy temporal logic for monitoring behavior-based mobile robots. In: The IASTED International Conference on Robotics and Applications (RA 2006). pp. 116–121. ACTA Press (aug 2006)
31. Mantyjarvi, J., Seppanen, T.: Adapting applications in handheld devices using fuzzy context information. Interacting with Computers **15**(4), 521–538 (2003)
32. Meyer, S., Rakotonirainy, A.: A survey of research on context-aware homes. In: Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003-Volume 21. pp. 159–168. Australian Computer Society, Inc. (2003)
33. Michels, K., Klawonn, F., Kruse, R., Nrnberger, A.: Fuzzy Control - Fundamentals, Stability and Design of Fuzzy Controllers, Studies in Fuzziness and Soft Computing, vol. 200. Springer (2006)
34. Milan, C., Bojesen, C., Nielsen, M.P.: A cost optimization model for 100renewable residential energy supply systems. Energy **48**(1), 118 – 127 (2012), 6th Dubrovnik Conference on Sustainable Development of Energy Water and Environmental Systems, SDEWES 2011
35. Molina, D., Lu, C., Sherman, V., Harley, R.: Model predictive and genetic algorithm based optimization of residential temperature control in the presence of time-varying electricity prices. In: Industry Applications Society Annual Meeting (IAS), 2011 IEEE. pp. 1–7 (Oct 2011)
36. ick Moon, S., Lee, K., Lee, D.: Fuzzy branching temporal logic. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **34**(2), 1045–1055 (April 2004)
37. Nguyen, T.A., Aiello, M.: Energy intelligent buildings based on user activity: A survey. Energy and Buildings **56**(0), 244 – 257 (2013)
38. Park, H., Yoo, J., Cho, S.: A context-aware music recommendation system using fuzzy bayesian networks with utility theory. Fuzzy Systems and Knowledge Discovery pp. 970–979 (2006)
39. Prähofer, H., Hurnaus, D., Schatz, R., Wirth, C., Mössenböck, H.: Monaco: A dsl approach for programming automation systems. In: Procs. of the Conference on Software Engineering (2006)
40. Preuer, S.: A domain-specific language for industrial automation. In: Software Engineering 2007 - Beiträge zu den Workshops, Fachtagung des GI-Fachbereichs Softwaretechnik, 27.-30.3.2007 in Hamburg. pp. 349–352 (2007)
41. Safdar, A., Kim, D.: Energy conservation and comfort management in building environment. International Journal of Innovative Computing **9**(6), 2229–2244 (2013)

42. Schmidt, A., Van Laerhoven, K.: How to build smart appliances? IEEE [see also IEEE Wireless Communications] Personal Communications **8**(4), 66–71 (2001)
43. Shaikh, P.H., Nor, N.B.M., Nallagownden, P., Elamvazuthi, I., Ibrahim, T.: A review on optimized control systems for building energy and comfort management of smart sustainable buildings. Renewable and Sustainable Energy Reviews **34**(0), 409 – 429 (2014)
44. Soucek, S., Zucker, G.: Current developments and challenges in building automation. e & i Elektrotechnik und Informationstechnik **129**(4), 278–285 (Jun 2012). https://doi.org/10.1007/s00502-012-0013-4, https://doi.org/10.1007/s00502-012-0013-4
45. Steinberg, D., Budinsky, F., Merks, E., Paternostro, M.: EMF: eclipse modeling framework. Pearson Education (2008)
46. Strejček, J.: Linear temporal logic: Expressiveness and model checking (2005)
47. Thiele, H., Kalenka, S.: On fuzzy temporal logic. In: [Proceedings 1993] Second IEEE International Conference on Fuzzy Systems. pp. 1027–1032 vol.2 (March 1993)
48. Torunski, E., Othman, R., Orozco, M., Saddik, A.E.: A review of smart environments for energy savings. Procedia Computer Science **10**(0), 205 – 214 (2012)
49. Üçtuğ, F.G., Yükseltan, E.: A linear programming approach to household energy conservation: Efficient allocation of budget. Energy and Buildings **49**(0), 200 – 208 (2012)
50. Waide, P., Gerundino, D.: International standards to develop and promote energy efficiency and renewable energy sources. International Energy Agency. International Organization for Standardization (IEAISO). (jun 2007)
51. Wang, Z., Wang, L.: Intelligent control of ventilation system for energy-efficient buildings with co2 predictive model. Smart Grid, IEEE Transactions on **4**(2), 686–693 (June 2013)
52. Wang, Z., Yang, R., Wang, L.: Multi-agent intelligent controller design for smart and sustainable buildings. In: Systems Conference, 2010 4th Annual IEEE. pp. 277–282 (April 2010)
53. Welch, B.L.: The generalization of 'student's' problem when several different population variances are involved. Biometrika **34**(1-2), 28–35 (1947). https://doi.org/10.1093/biomet/34.1-2.28, http://dx.doi.org/10.1093/biomet/34.1-2.28
54. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in software engineering. Springer-Verlag Berlin Heidelberg, Germany (2012). https://doi.org/10.1007/978-3-642-29044-2
55. Yang, R., Wang, L.: Multi-objective optimization for decision-making of energy and comfort management in building automation and control. Sustainable Cities and Society **2**(1), 1 – 7 (2012)
56. Zadeh, L.A.: Fuzzy logic = computing with words. Trans. Fuz Sys. **4**(2), 103–111 (May 1996)
57. Zadeh, L.A.: Outline of a new approach to the analysis of complex systems and decision processes. Systems, Man and Cybernetics, IEEE Transactions on **SMC-3**(1), 28–44 (Jan 1973)
58. Zadeh, L.A.: The concept of a linguistic variable and its application to approximate reasoning–i* 1. Information sciences **8**(3), 199–249 (1975)
59. Zadeh, L.A.: A theory of approximate reasoning. Machine Intelligence **9**, 149–194 (1979)
60. Zadeh, L.A.: Fuzzy logic. Computer **21**(4), 83–93 (1988)