



**Diego de Freitas Bezerra**

## **Automatic Synthesis of Controllers for the Internet of Things**



Universidade Federal Rural de Pernambuco  
coordenacao@ppgia.ufrpe.br  
<http://ppgia.ufrpe.br>

Recife  
2020

**Diego de Freitas Bezerra**

**Automatic Synthesis of Controllers for the Internet of Things**

A M.Sc. Dissertation presented to the Postgraduate Program in Applied Informatics of Federal Rural University of Pernambuco in partial fulfillment of the requirements for the degree of Master of Science in Applied Informatics.

**Concentration Area:** Intelligent Computing and Modeling

**Advisor:** Prof. Dr. Glauco Estácio Gonçalves

**Co-advisor:** Prof. Dr. Victor Wanderley Costa de Medeiros

Recife

2020

**FICHA**

**Diego de Freitas Bezerra**

**“Automatic Synthesis of Controllers for the Internet of Things”**

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática Aplicada da Universidade Federal Rural de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Informática Aplicada.

Aprovado em: 20/02/2020.

**BANCA EXAMINADORA**

---

Prof. Dr. Djamel Fawzi Hadj Sadok  
Centro de Informática / UFPE

---

Prof. Dr. Sidney de Carvalho Nogueira  
Departamento de Computação / UFRPE

---

Prof. Dr. Glauco Estácio Gonçalves  
Departamento de Estatística e Informática / UFRPE  
(**Orientador**)

*I dedicate this work to my parents, Jane and José (In Memoriam), who gave me all the necessary support to get me here.*

## ACKNOWLEDGEMENTS

First of all, I would like to thank my parents, Jane and José (*In Memoriam*), for the love, teachings, and all the encouragement they always gave me so that I could get here.

I would like to thank my girlfriend, Clicia, for their friendship, patience on stressful days, and all the support you gave me during this research.

I would like to thank my advisor Dr. Glauco Gonçalves and co-advisor Dr. Victor Medeiros, for the friendship, support, and extra-academic conversations that provided me with a lot of learning and experience. Thank you for all the opportunities you have provided me.

I would like to thank to my colleagues at Juá Labs who made many of my days happier during this hard academic journey. I learned a lot from each of you. Special thanks to the masters Clodoalves and Wellington. Thanks for the partnership, guys.

Finally, I would like to thank the Fundação de Amparo a Ciência e Tecnologia do Estado de Pernambuco (FACEPE) for funding this work through grant IBPG-0188-1.03/18.

*There is something good in this world,  
and it's worth fighting for.*  
— J.R.R. TOLKIEN

## RESUMO

A popularização da Internet das Coisas abriu oportunidades para aplicações em vários setores econômicos, permitindo monitorar e controlar diversos tipos de ambientes. No entanto, com o crescente número de dados enviados pelos dispositivos, as aplicações passaram a demandar maior disponibilidade e largura de banda da Internet, mas essas demandas ainda são um gargalo nas regiões mais distantes dos grandes centros urbanos e com menos desenvolvimento econômico. Neste trabalho, examinamos como as limitações de largura de banda podem afetar aplicações de Internet das Coisas hospedadas em nuvem e propomos uma arquitetura de Controle como Serviço baseada em *Fog Computing* para processar dinamicamente eventos no contexto de ambientes inteligentes. A arquitetura é composta por um Mecanismo de Regras e um Processador de Eventos Complexos, baseado nas técnicas da Teoria de Controle Supervisório para Sistemas de Eventos Discretos. O Mecanismo de Regras permite a definição dinâmica de regras que condicionam o controle a partir de diferentes eventos de entrada. O Processador de Eventos Complexos com capacidade de síntese e reconfiguração automática permite executar ações de controle para regras estabelecidas. Apresentamos cenários em que essa solução pode ser aplicada, como Agricultura Inteligente, Edifícios Inteligentes e Casas Inteligentes. Para validação, a arquitetura foi implementada e integrada a uma simulação de estufas agrícolas, permitindo o controle da temperatura no ambiente. Dados climáticos reais de estações meteorológicas brasileiras foram utilizados para avaliar diferentes cenários de controle nas quatro estações do ano.

**Palavras-chaves:** Síntese de Controladores Discretos, Infraestrutura Computacional, Controle de Ambientes



## ABSTRACT

The popularization of the Internet of Things has opened opportunities for applications in various economic sectors, allowing them to monitor and control many types of environments. However, with the increasing number of data sent by devices, applications have come to demand increased availability and Internet bandwidth, but these demands are still a bottleneck in regions farther from large urban centers and with less economic development. In this work, we examine how bandwidth limitations can affect cloud-based applications and propose a Fog Computing-based Control-as-a-Service architecture to dynamically process events in the context of the Internet of Things and Smart Environments. The architecture is composed of a Rules Engine and a Complex Event Processor based on Supervisory Control techniques for Discrete Event Systems. The Rules Engine allows defining dynamic rules conditioning control from different types of input data. The Complex Event Processor with automatic synthesis and reconfiguration capability allows performing control actions to established rules. We present scenarios where this solution can be applied, such as Smart Agriculture, Smart Building, and Smart Home. For validation, the architecture was implemented and integrated into a simulation for agricultural greenhouses, allowing them to control the temperature inside the environment. Climate data from actual Brazilian meteorological stations were used to evaluate different control scenarios in the four seasons of the year.

**Key-words:** Discrete Controller Synthesis, Computing Infrastructures, Environment Control

## LIST OF FIGURES

Figure 1 – Example of finite-state deterministic automata. . . . .	21
Figure 2 – Performance of a supervisor (S) in a plant (G). . . . .	23
Figure 3 – Overview of the discrete controller synthesis. . . . .	24
Figure 4 – Simple BZR node. . . . .	25
Figure 5 – Fog architecture. . . . .	28
Figure 6 – Scenario simulated. . . . .	36
Figure 7 – Proposed architecture. . . . .	38
Figure 8 – RE graphical interface. . . . .	39
Figure 9 – Events designed for smart home controller in the Rules Engine. . . . .	44
Figure 10 – Events designed for greenhouse controller in Rules Engine. . . . .	49
Figure 11 – Simulation results for air temperature: The red lines indicates the temperature inside the greenhouse without control; the green lines represent the temperature inside the greenhouse when the CEP is acting; and the blue lines present the air temperature outside the greenhouse. . . . .	51
Figure 12 – Simulation results for absolute humidity: the red lines indicates the absolute humidity inside the greenhouse without control; the green lines represent the absolute humidity inside the greenhouse when the CEP is acting; and the blue lines present the absolute humidity outside the greenhouse. . . . .	52
Figure 13 – Simulation results for relative humidity: the red lines indicates the relative humidity inside the greenhouse without control; the green lines represent the relative humidity inside the greenhouse when the CEP is acting; and the blue lines present the relative humidity outside the greenhouse. . . . .	53

## LIST OF TABLES

Table 1 – Related work based on Complex Event Processing . . . . .	32
Table 2 – Related work based on Discrete Controller Synthesis . . . . .	34
Table 3 – Packet loss rate in a 4G application scenario . . . . .	36
Table 4 – Events designed for smart home controller. . . . .	44
Table 5 – Events designed for greenhouse controller. . . . .	48

## **LIST OF ABBREVIATIONS AND ACRONYMS**

**IoT** Internet of Things

**CEP** Complex Event Processing

**RE** Rules Engine

**IBGE** Instituto Brasileiro de Geografia e Estatística

**DES** Discrete Event System

**CVDS** Continuous Variable Dynamic System

**DCS** Discrete Controller Synthesis

**UDP** User Datagram Protocol

**CPU** Central Processing Unit

**RAM** Random Access Memory

## CONTENTS

<b>1</b>	<b>INTRODUCTION . . . . .</b>	<b>14</b>
1.1	MOTIVATIONS . . . . .	15
1.2	OBJECTIVES . . . . .	16
1.3	ORGANIZATION OF THE WORK . . . . .	16
<b>2</b>	<b>BACKGROUND . . . . .</b>	<b>18</b>
2.1	DISCRETE EVENT SYSTEMS . . . . .	18
<b>2.1.1</b>	<b>Modelling Discrete Event Systems . . . . .</b>	<b>19</b>
2.1.1.1	Language . . . . .	19
2.1.1.2	Automata . . . . .	21
2.2	SUPERVISORY CONTROL THEORY . . . . .	22
2.3	DISCRETE CONTROLLER SYNTHESIS . . . . .	23
2.4	RULES ENGINE AND COMPLEX EVENT PROCESSING . . . . .	26
2.5	FOG COMPUTING . . . . .	27
2.6	CONCLUDING REMARKS . . . . .	28
<b>3</b>	<b>RELATED WORK . . . . .</b>	<b>30</b>
3.1	COMPLEX EVENT PROCESSING . . . . .	30
3.2	DISCRETE CONTROLLER SYNTHESIS . . . . .	32
3.3	CONCLUDING REMARKS . . . . .	34
<b>4</b>	<b>PROPOSED ARCHITECTURE . . . . .</b>	<b>35</b>
4.1	WHY FOG-BASED APPLICATIONS? . . . . .	35
4.2	ARCHITECTURE . . . . .	37
<b>4.2.1</b>	<b>Rules Engine . . . . .</b>	<b>37</b>
<b>4.2.2</b>	<b>Complex Event Processor Generator . . . . .</b>	<b>39</b>
<b>4.2.3</b>	<b>Complex Event Processor . . . . .</b>	<b>40</b>
4.3	CONCLUDING REMARKS . . . . .	41
<b>5</b>	<b>CASE STUDIES . . . . .</b>	<b>43</b>
5.1	CASE STUDY I: SMART HOME . . . . .	43
5.2	CASE STUDY II: SMART FARMING . . . . .	45
<b>5.2.1</b>	<b>Mathematical model . . . . .</b>	<b>46</b>
<b>5.2.2</b>	<b>Data and method . . . . .</b>	<b>47</b>
5.2.2.1	Data . . . . .	47
5.2.2.2	Rules . . . . .	48
<b>5.2.3</b>	<b>Greenhouse setup . . . . .</b>	<b>49</b>

5.2.4	<b>Results</b> . . . . .	<b>50</b>
5.3	CONCLUDING REMARKS . . . . .	54
<b>6</b>	<b>CONCLUSION</b> . . . . .	<b>55</b>
6.1	LIMITATIONS . . . . .	55
6.2	CONTRIBUTIONS . . . . .	56
6.3	PUBLICATIONS . . . . .	56
6.4	FUTURE WORKS . . . . .	57
	<b>REFERENCES</b> . . . . .	<b>58</b>

## 1 INTRODUCTION

Internet of Things (IoT) has become more and more present in people's lives. It has been causing a significant impact on the industry, agriculture, healthcare, logistics, transports, and several other activities around the world by solving various problems more efficiently (ASGHARI; RAHMANI; JAVADI, 2019).

These solutions have been playing an essential role in the infrastructure of these sectors. Through sensors, actuators, wireless networks, and software capable of processing and analyzing environmental data, it is possible to deliver solutions that provide essential information that directly impacts the quality of services and optimizes the resources usage (SOURI et al., 2019; TALAVERA et al., 2017).

In the agricultural sector, for example, these solutions have contributed to optimize resource usage by reducing costs and increasing productivity through optimum conditions for farming (MUANGPRATHUB et al., 2019). In this context, sensors (e.g., soil moisture, relative air humidity, air temperature, among others) provide data for controlling environmental parameters, like daily irrigation duration, based on agro-technical measurements.

In Smart Home and Smart Building context, it is possible to optimize energy consumption by controlling equipment such as heating, ventilation, air conditioning, and lighting systems (MOHAMED; AL-JAROODI; JAWHAR, 2018). From data obtained by the sensors (e.g., presence, lighting, temperature, and others), these solutions monitor the energy consumption and adjust the operation of each equipment according to the energy-saving policies and comfort levels required by the residents. Besides, these consumption data are also of interest to electric energy service providers who can analyze peak energy consumption times (ZHOU; FU; YANG, 2016).

Although there are several off-the-shelf solutions for sensing and actuating on different types of environments (SINCHE et al., 2019), they usually not correspond to users' needs in specialized contexts. These solutions offer generic features to reach a more extensive and diverse number of consumers. However, this feature can often require the development of other solutions to integrate and meet specifications, or even the replacement of the technology used (CARDUCCI et al., 2019).

Typically, such solutions are complex to develop, integrate, and deploy, requiring IoT experts and other domain specialists (MAZON-OLIVO et al., 2018), which imply a high financial, personnel, and time cost. The process of developing, for example, must consider aspects such as the heterogeneity of the devices, the distributed, highly dynamic, and mostly asynchronous nature of software (TAIVALSAARI; MIKKONEN, 2017). In addition to these aspects, specific aspects of each domain must be considered for the use of the most appropriate technology.

Commonly, state-of-the-art solutions for controlling have low adaptability to meet new

requirements (e.g., new parameters and control rules) and to support the growing number of sensor and control devices. In practical terms, adaptation would involve only the adjustment of the control rules, and it would not require the development of a new controller from scratch. Thus, there is a demand for solutions that quickly and inexpensively adapt the control to meet new requirements and to serve different scenario conditions.

In order to meet these smart environments demands, Mazon-Olivo et al. (2018) propose a generic Cloud-based architecture to process events based on a Rules Engine (RE) and a Complex Event Processor (CEP). In this proposal, the CEP analyzes the input data based on the rules defined in the RE and generates control actions. The CEP verifies each event that arrives in order to assure that it meets a defined rule, and, consequently, takes the corresponding action.

However, this approach relies heavily on the underlying communication resources as the solution scales – i.e., as the data flow increases – requiring a large amount of data to be sent to the cloud service that implements the CEP. For example, a 100-hectare banana plantation with 14 soil moisture sensors distributed on each hectare can generate more than 8GB of data in just one day. This way, bandwidth and Internet availability can impose a substantial limitation on the viability of a cloud-based CEP in a real scenario (CASTILLO-CARA et al., 2018; MIHAI et al., 2018).

## 1.1 MOTIVATIONS

Even in urban environments, user experiences with 4G networks in a developing country like Brazil reveal that signal availability is not optimal (82%). Besides, upload data rates are still low, ranging from 2.9 Mbps to 6.4 Mbps, depending on carrier (FITCHARD, 2019). When we extended this analysis to Ecuador, where the experiments presented by Mazon-Olivo et al. (2018) were performed, user experience reveals even lower availability (62.6%) and upload rates between 2.4 Mbps and 4.6 Mbps (BOYLAND, 2019).

It is essential to highlight that rural areas often face access problems with low upload data rates and limited or nonexistent connectivity. In Brazil, for example, 71.8% of rural properties do not have internet access (IBGE, 2018). This problem affects about 3.64 million of the 5.07 million rural properties. In this sense, the distance to urban centers and low user density pose challenges for cloud-based applications.

Another important fact is that Mazon-Olivo et al. (2018) does not define a mechanism capable of verifying the existence of conflicting rules in the Rules Engine. This type of mechanism is essential to ensure that the rules do not violate or contradict each other, preventing scenarios in which, for example, one rule defines turning on a lamp while another rule turns it off under some conditions.

Given the connectivity limitations that Cloud-based applications face, Fog Computing-based approaches emerge as candidate solutions for providing network, compute, and storage services in areas with weak or nonexistent network infrastructure (PULIAFITO et



al., 2019). Fog Computing enables to extend current Cloud Computing services to the edge of the network – where data is generated – minimizing response time issues for latency-sensitive applications, bandwidth consumption, and signal availability (AL YAMI; SCHAEFER, 2019).

This work proposes a Control-as-a-Service architecture that delivers control to Smart Environments. Based on Discrete Control Synthesis (DCS) techniques (CASSANDRAS; LAFORTUNE, 2009), our solution is capable of automatically synthesize a CEP in order to satisfy the rules defined by the user. Our work extends the solution presented in (MAZON-OLIVO et al., 2018) to support a Fog Computing based architecture but preserving adaptability and reconfigurability requirements. Our solution requires a single query to RE in order to automatically produce a new CEP-as-a-Service that, after synthesis, runs independently of the RE in the Fog. This feature enables applications in environments where there is not enough Internet bandwidth to send a large amount of data collected by devices.

To demonstrate the main advantages of our architecture, we present possible scenarios in the context of Smart Farms, Smart Building, and Smart Home, to which the proposed architecture can be applied. In the Smart Farms context, a functional prototype was implemented and the control was verified through simulations. In those simulations, a synthesized CEP controls a set of actuators in order to adjust the environmental conditions inside an agricultural greenhouse. The system was tested using actual hourly air temperature, absolute humidity, and solar radiation data from different seasons of the year in Brazil.

## 1.2 OBJECTIVES

The main objective of this work is to propose and evaluate a control strategy for IoT applications in the context of Smart Environments. The proposed strategy is expected to allow automatically derive controllers according to multi-tenant requirements. To achieve this goal, specifically, it is planned:

- Propose a Fog-based framework that allows automatic reconfiguration of Smart Environments;
- Develop a prototype in order to evaluate the application of the solution using real data;

## 1.3 ORGANIZATION OF THE WORK

This work is organized as follows: Chapter 2 describes some basic concepts about Discrete Controller Synthesis, Rules and Complex Event Processor, and Fog Computing; Chapter 3 presents the related work; Chapter 4 describes the proposed architecture, detailing

each component; Chapter 5 presents scenarios in which the proposed architecture can be applied; At last, Chapter 6 discusses some conclusions and future works.

## 2 BACKGROUND

This Chapter describes some of the main concepts necessary to understand the purpose of this dissertation. Section 2.1 discusses the concept of Discrete Event Systems and presents formal models for describing such systems; Section 2.2 presents the main concepts of the Supervisory Control Theory applied in this work. The Section 2.3 presents the concept of Discrete Controller Synthesis and the tools used in the literature for the automatic synthesis of controllers. Section 2.4 introduces the concept of the Rules Engine and Complex Event Processors. Finally, Section 2.5 introduces the definition of Fog Computing and concludes this chapter.

### 2.1 DISCRETE EVENT SYSTEMS

The sophistication of the problems faced by humanity in the various areas of knowledge has demanded increasingly sophisticated answers and the search for tools to analyze, understand, and solve the problems presented by complex systems. In this context, a system can be defined as a set of interdependent components that perform a task, and this task cannot be performed by only one of the components. This concept can be applied not only to physical systems but also to abstract phenomena such as financial transactions.

Studies applied to a system usually seek to understand the relationships between a set of input and output variables, describing the behavior of that system. To understand behavior, it is usually necessary to describe it through a mathematical model. In this sense, the input variables,  $u(t) = [u_1(t) \cdots u_n(t)]^T$ , present information about some phenomenon and can be directly manipulated. In turn, the output variables,  $y(t) = [y_1(t) \cdots y_m(t)]^T$ , describe the behavior of the system. The mathematical model that describes the relationship between these variables can be defined as  $y = g(u)$ , where  $g(\cdot)$  represents the mathematical model.

In the literature, there are numerous classifications for the various types of systems. Systems whose output values at a given time  $t$  do not depend on input values in the period before  $t$  are called static systems – they are also called memoryless systems. On the other hand, systems in which the output values at time  $t$  depend on the current input values and past values are called dynamic systems. In this context, the past behavior information needed to know future system output values is based on the state concept.

In dynamic systems, the smallest set of variables that determine the behavior of a system at any given time is defined as a state. The set of all possible states that the system can assume is defined as state space. System behavior may change as input variables become known. The behavior change is defined by transitions between the states and happen when events are triggered.

An event is defined as an instantaneous occurrence that causes the transition between system states. Events may be associated with a specific action, such as turning the light switch on or associated with a spontaneous occurrence of nature, such as a power outage for some unexpected reason. In different types of systems, events can occur at well-defined time intervals. However, in other types of systems, these events may occur at irregular and unknown time intervals.

A dynamic system of discrete-state space whose state transitions occur through the occurrence of one or more events that occur abruptly, at generally irregular and unknown time intervals, is characterized as a Discrete Event System (DES). Importantly, these characteristics differ the Discrete Event Systems from Continuous Variable Dynamic Systems (CVDS), which have a continuous state space and are usually described by differential equations. On the other hand, because they are dynamic discrete state space systems, DES require distinct modeling.

### 2.1.1 Modelling Discrete Event Systems

Different types of formalism can define discrete event systems. The choice for a particular formalism is usually associated with the level of abstraction required by the model and the objectives of the analysis. In this sense, we are interested in understanding system at their logical level. That is, we are concerned with ensuring that a sequence of events meets a particular set of specifications.

At a logical level, considering the evolution of the states of a Discrete Event System, our main concern is with the sequence of visited states and the associated events causing state transitions. We can say that a DES model is composed of two elements: states and events. A formal way to describe the behavior of a DES, at a logical level, is through a language. In this case, we can characterize as system language all sequences of events that can be generated by DES. In this approach, information about the time of occurrence of each event is not considered.

Among the formalisms that can be used to represent the languages of a DES, the most widespread are the Automata and Petri Nets (CASSANDRAS; LAFORTUNE, 2009). Although automata can present large state spaces in the modeling process, they can be intuitive and parsable, advantages that will be useful for the applications of this work. Here, we will present the fundamental concepts of language theory and automata for modeling the behavior of systems at a logical level.

#### 2.1.1.1 Language

A language defined over a set of events  $E$  (alphabet) is a set of finite sequences (strings) of arbitrarily long length, formed by events belonging to  $E$ . For example, consider the event set  $E = \{a, b\}$ , we can define the following languages:

- $L_1 = \{\epsilon, a, abb\}$  - consisting of three strings;
- $L_2 = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$  - consists of all possible strings of length three;
- $L_3 = \{a, aa, ab, aaa, aab, aba, abb, \dots\}$  - all possible strings of finite length which start with event  $a$ .

Given the string  $s = abc$  such that  $a, b, c \in E^*$ , we define that  $a$  is the prefix of  $s$ ;  $b$  is a substring of  $s$ ; and  $c$  is the suffix of  $s$ . The length of  $s$  is given by  $|s|$  and defines the number of events that make up the string. Finally,  $\epsilon$  and  $s$  are prefixes, substrings, and suffixes of  $s$ .

The set of all finite-length strings formed by events belonging to  $E$ , including  $\epsilon$ , is called Kleene closure and is denoted by  $E^*$ . Note that any language defined over  $E$  is a subset of  $E^*$ . From the previously defined set of events, the Kleene close operation is given by  $E^* = \{\epsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, bab, bba, bbb, \dots\}$ .

It is important to note that, by definition, the usual operations in set theory with respect to  $E^*$ , – i.e., union, intersection, difference, and complement –, are applicable. In addition to these, other operations are also used, such as concatenation, prefix-closure, and Kleene-closure. We will add these three operations here.

- **Concatenation:** Let  $L_1, L_2 \subseteq E^*$ , then:

$$L_1 L_2 := \{s \in E^* : (s = s_1 s_2) \text{ and } (s_1 \in L_1) \text{ and } (s_2 \in L_2)\}.$$

This operation associates strings pairs with a string formed by juxtaposing the first with the second string.

- **Prefix-closure:** Let  $L \subseteq E^*$ , then

$$\bar{L} := \{s \in E^* : \exists t \in E^* (st \in L)\}$$

$\bar{L}$  consisting of all the prefixes of all the strings in  $L$ .

- **Kleene-closure** Let  $L \subseteq E^*$ , then

$$L^* := \{\epsilon \cup L \cup LL \cup KKK \cup \dots\}.$$

This is the same operation defined previously for the set  $E$ .

### 2.1.1.2 Automata

Automata are tools that allow us to represent an important class of formal languages. This formalism allows us to describe the behavior of a system by defining all states and transitions that are associated with events that promote state changes.

Among the different types of automata, the Finite-State (Deterministic) Automata best represent a Discrete Event System (CASSANDRAS; LAFORTUNE, 2009). The deterministic term is used to emphasize that two transitions cannot exist with the same event label. Formally, a Finite-State Deterministic Automata is defined as a 5-uple  $G = (X, E, f, x_0, X_m)$ , where:

- $X$  is the set of states;
- $E$  is the finite set of events that describe the possible state transitions of  $G$
- $f : X \times E \rightarrow X$  is the transition function;
- $x_0$  is the initial state of  $G$ ;
- $X_m$  is the set of marked states of  $G$ ,  $X_m \subseteq X$ .

The behavior of an automaton can be illustrated through a transition diagram, where nodes represent states, and labeled arcs represent transitions between states. Figure 1 illustrates an example of finite-state deterministic automata, the formal description of which is given by:

- $X = \{x, y\}$ ;
- $E = \{a, b\}$ ;
- The transition function:  $f(x, a) = x$ ,  $f(x, b) = y$ ,  $f(y, a) = x$ ,  $f(y, b) = y$ ;
- $x_0 = \{x\}$ ;
- $X_m = \{x\}$ ,

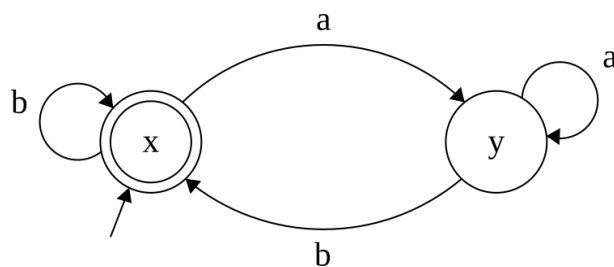


Figure 1: Example of finite-state deterministic automata.

The automata  $G$  has the initial state  $x$  and remains in it until the  $a \in E$  event occurs and triggers the  $f(x, a) \in X$  transition. This process continues based on the transition functions defined in  $f$ .

$G$  is associated with two languages: the generated language  $L(G)$  and the marked language  $L_m(G)$ .  $L(G)$  is the set of strings that can be generated from  $G$ , starting from the initial state. It is formally defined by:

$$L(G) := \{s \in E^* : f(x_0, s) \text{ is defined}\}.$$

The  $L_m(G)$  language considers all strings that, starting from the initial state, reach a marked state. Its definition is given by:

$$L_m(G) := \{s \in L(G) : f(x_0, s) \in X_m\}.$$

From these definitions, a DES can be modeled by a  $G$  automaton, where  $L(G)$  is the behavior of the generated system and  $L_m(G)$  is the marked behavior or complete task set of the system (CASSANDRAS; LAFORTUNE, 2009). Importantly, from these definitions, all references to the term automata is assumed to be a finite-state deterministic automata.

## 2.2 SUPERVISORY CONTROL THEORY

As seen earlier, a system can be interpreted as a set of components that act to solve a task. Looking at each component (subsystem) individually, we have that each of them has a specific behavior and act in a coordinated manner so that the overall system (plant) achieves its goals. However, the occurrence of some sequences of events may cause subsystems to behave unwantedly.

To solve this problem, the Supervisory Control Theory proposes the existence of a supervisor to coordinate the subsystems in a coordinated manner, keeping the system following the desired behavior (RAMADGE; WONHAM, 1989). In this approach, the supervisor observes the sequence of events generated by the system and instantly enables or disables a subset of events to maintain the desired behavior. This subset of events defines a control input and specifies all events enabled to occur. In practical terms, the application of this theory controls the behavior of subsystems, ensuring that specifications are not violated.

The application of supervisory control theory can consider two strategies: centralized supervision or modular supervision (BRANDIN; CHARBONNIER, 1994). Centralized supervision consists of a single supervisor that performs the system supervision task, observing every occurrence of events in the plant. In the case of modular supervision, the supervision task is divided into two or more supervisors, who have a partial view of the plant, acting concurrently.

Considering the centralized supervisory approach, we define that the composition of all behaviors of each subsystem can be identified with the plant  $G$ , with generated be-

havior  $L(G)$  and marked behavior  $L_m(G)$ . In turn, a set of rules defines a specification to be complied with by the system, and in this sense,  $L(G)$  contains strings that are not acceptable because they violate at least one of these rules. In this case, the role of the supervisor, denoted here by  $S$ , is to observe all events occurring in  $G$  and to define which events, among all possible in the current state, are allowed to occur next. In this case,  $S$  acts on the subsystem, disabling some events, but not necessarily all, according to control specifications.

As stated earlier, the supervisor acts on a specific set of events. In this case, the set of events of  $G$  can be partitioned into two disjoint sets: the set of controllable events and the set of uncontrollable events (RAMADGE; WONHAM, 1989). This partitioning can be defined as  $E = E_c \cup E_{un}$ , where  $E$  is the set of all events of  $G$ ,  $E_c \subseteq E$  is the set of controllable events and  $E_{un} \subseteq E$  is the set of uncontrollable events.

The supervisor's role in the plant defines a  $\gamma$  subset of events that can occur. In this case, we define that a control entry in  $G$  generates a subset  $\gamma \subseteq E$  that satisfies  $E_{un} \subseteq \gamma$ . In other words,  $\gamma$  is the set composed of all uncontrollable events and the controllable events that can occur. In this case, we define the control structure  $\Gamma$  such as  $\gamma \in \Gamma \subseteq 2^E$ .

The  $S$  performance in a DES match is formally defined as  $S : L(G) \rightarrow \Gamma$  which associates a sequence of events  $s \in L(G)$  generated by  $G$ , with an entry of  $\gamma = S(s)$  control that restricts its behavior. Figure 2 illustrates the operation of a centralized supervisor suing  $G$ .

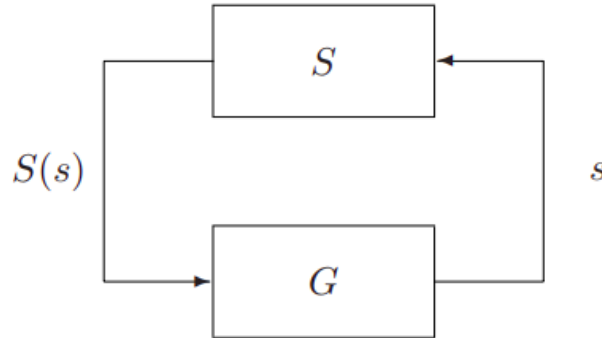


Figure 2: Performance of a supervisor (S) in a plant (G).

Note that  $G$  generates sequences of controllable and uncontrollable events ( $s$ ) that are being observed by  $S$ , which has the responsibility to act on controllable events only – i.e.,  $S(s)$ .

### 2.3 DISCRETE CONTROLLER SYNTHESIS

Discrete Controller Synthesis (DCS) is a formal method used to generate a controller from a set of rules (contract) defined by a model. Based on Supervisory Control Theory, to guarantee the desired behavior, the method partitions the model input variables into



controllable and uncontrollable variables (DELAVAL; RUTTEN; MARCHAND, 2013). Thus, for a given set of rules, the DCS algorithm calculates the constraint on controllable variables through the symbolic exploration of the overall state space, providing a controller that complies with the contract for any input values (CANO; DELAVAL; RUTTEN, 2014).

DCS is integrated into the synchronous language Heptagon/BZR compilation process. Heptagon/BZR is a reactive language that allows the generation of controller models based on finite state automata. The language combines imperative and declarative programming paradigms. The behavior of the system is described through the imperative paradigm based on automata. The specifications of the control objectives are defined through the declarative paradigm. It integrates a contract mechanism that allows to use DCS through Sigali (DELAVAL; RUTTEN; MARCHAND, 2013), a DCS tool. The result of this process is executable code (C or Java) of a controller capable of responding to the contract defined in the modeling process.

There are four main elements for constructing the Heptagon/BZR model: node, equation, automata, and contract. The node defines blocks of equations or automata with input and output events; equation defines the output value of the node; automata describes finite automata using states, input, and output events; and the contract defines the rules to be followed by the controller (DELAVAL; RUTTEN; MARCHAND, 2013).

Figure 3 presents the synthesis process of a DCS from the DES description in Heptagon/BZR. The model is constructed in a \*.ept file which provides the logical description of the DES control. The Heptagon component is responsible for extracting control objectives and describing DES behavior for Sigali.

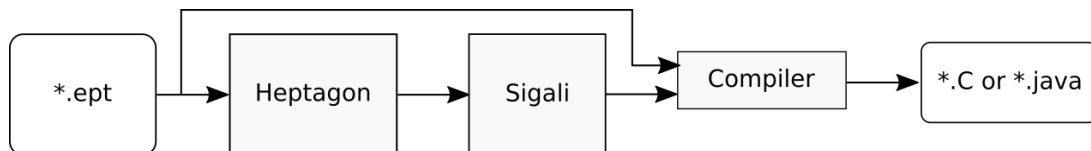


Figure 3: Overview of the discrete controller synthesis.

Sigali, in turn, applies the synthesis process from the DES description and control objectives, generating an intermediate controller. The Compiler component is responsible for composing the controller synthesized by Sigali with the DES description, producing the correct controlled automaton. Finally, the resulting composition is compiled by generating one C or Java code (according to the choice of the modeller) that implements the controller.

To exemplify the process of synthesizing a controller using Heptagon/BZR and Sigali, we describe and model a controller to coordinate the operation of a DES. In this case, we propose a controller that coordinates the use of a particular resource to which two processes compete.

Figure 4 illustrates the automaton of a process named *delayable*. Each process has

three possible states: *Idle*, *Wait*, or *Active*, with *Idle* being the initial state. The *act* variable can assume true or false values and indicates whether the system is in the Active state. Three input variables define transitions between states. In this case, when *r* and *c* are true, the system state changes to *Active*. If *e* is true while the state is *Active*, the system returns to *Idle*. When *c* is false and *r* is true, the system state turns to the *Wait* state and remains in that state until *c* is true.

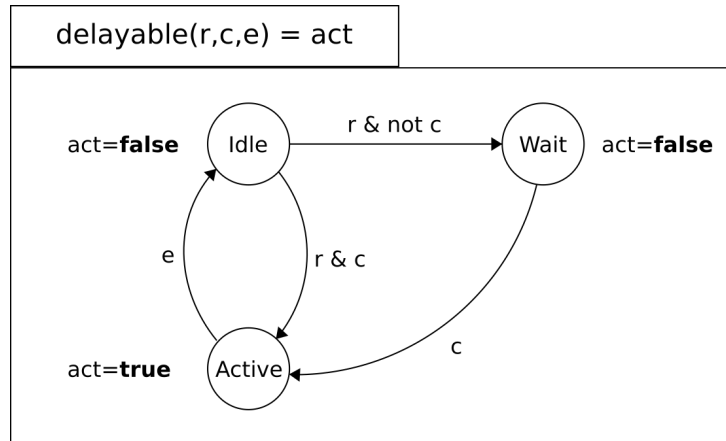


Figure 4: Simple BZR node.

In this model, *c* is a controllable event and it represents the act of the supervisor on the plant, in order to coordinate the transition of the process to the Active state. Variables *r* and *e* are uncontrollable events.

Listing 2.1 presents the definition of the *delayable* automaton using the Heptagon/BZR language. The first line declares the node (*delayable*), input variables (*r*, *c* and *e*, of boolean type), and an output variable (*act*, boolean also). Lines 2 and 16 delimit the node implementation, and automaton description is made between lines 3 and 15, where the *Idle*, *Wait*, *Active* states, and possible transitions (*unless* declarations) are defined.

```

1 node delayable(r,c,e: bool) returns (act:bool)
2 let
3   automaton
4     state Idle do
5       act = false;
6       unless r & c then Active
7       | r & not c then Wait
8     state Wait do
9       act = false;
10      unless c then Active
11      | e then Idle
12    state Active do
13      act = true;
14      unless e then Idle
15    end
16 tel

```

Listing 2.1: Delayable automata description using Heptagon/BZR language.

Listing 2.1 describes only the behavior of each process. The node *twotasks* (cf. Listing 2.2) models the processes and the rules to be followed are defined (*contract* declaration). The two delayable automata are declared at lines 6 and 7.

Such automata are at first, independent since each one has its own *r*, *c*, and *e* events. However, the supervisor coordinates its execution as the contract defined for this example prevents both processes from remaining in the Active state at the same time (line 3), preventing the simultaneous usage of the resource. To ensure that this rule is met, the supervisor acts on each automata through variables *c1* and *c2*.

```

node twotasks(r1,e1,r2,e2: bool) returns (a1,a2: bool)
2   contract
    enforce not (a1 & a2)
4   with (c2, c1:bool)
    let
6       a1 = inlined delayable(r1,c1,e1);
       a2 = inlined delayable(r2,c2,e2)
8   tel

```

Listing 2.2: Delayable contract node description using Heptagon/BZR language.

Several work have used the Heptagon/BZR and Sigali for the synthesis of controllers. Some of these works applied DCS to the context of smart environments, as smart houses or smart offices, modeling each device present in the environment and establishing contracts to define the policies of operation on the environment (ZHAO et al., 2013; SYLLA; LOUVEL; RUTTEN, 2017; SYLLA et al., 2018; GUILLET; BOUCHARD; BOUZOUANE, 2013).

## 2.4 RULES ENGINE AND COMPLEX EVENT PROCESSING

A RE is a component that allows non-professional users to define the rules of a system, managing business logic through simple or complex rules, combining logic and relational variables and operators (MAZON-OLIVO et al., 2018; SUN et al., 2015). In the context of IoT, this means offering to the non-professional users a graphical interface that allows establishing rules between the devices in an environment (i.e., sensors and actuators). Thus, users add, modify, and delete rules according to their requirements (SUN et al., 2015).

CEP is a technology capable of communicating with a Rules Engine and other applications to processing, analyzing, and correlating large streams of data from multiple sources – e.g., wireless sensor networks, internal databases, web services, among others –, allowing the detection of patterns for decision making from the detection of composite events (WANG; RUNDENSTEINER; ELLISON, 2011; WANG; GAO; CHEN, 2018).

Among other development factors of CEPs, they generally differ in terms of the language used to define rules for handling events. Most of them have their language (BECK;

DAO-TRAN; EITER, 2018). Some approaches use more friendly rules mechanisms for the use of non-expert users (MAZON-OLIVO et al., 2018).

Communicating directly with the RE, the CEP can define complex events over simple events – i.e., raw data. In this sense, each existing rule in RE defines a pattern and triggers a complex event when that pattern corresponds to some simple event that reaches the CEP (MORENO et al., 2019). In this sense, CEP focuses on continuously analyzing and processing data that reaches it (JUNIOR; OLIVIERI; ENDLER, 2019). This feature makes CEP useful for processing and analyzing large data streams in real-time and producing results with low latency.

Two essential types of CEP can be found: reactive or proactive. A reactive CEP can process a pattern of complex events and automatically generate decisions. This type of solution allows, for example, sending control actions to the devices existing in an environment (MAZON-OLIVO et al., 2018). In a smart home scenario, this type of solution can detect the thermal discomfort of residents, either through temperature and humidity data or by the number of people occupying the place in a given time interval, and reacts by turning on an air-conditioning.

A proactive CEP can identify, eliminate, or mitigate the effects of future events by applying forecasting and decision techniques using machine learning, data mining, or other approaches (WANG; GAO; CHEN, 2018; AKBAR et al., 2017). This type of application allows, for example, to detect that the consumption of electricity in a smart home will exceed the defined limits to ensure the comfort of residents. In this case, the CEP starts to coordinate the functioning of the devices to prevent the limit of energy consumption from being exceeded, redefining the time interval that each one can remain turned on and synchronizing the operation of each device.

## 2.5 FOG COMPUTING

The popularization of IoT applications in the context of intelligent environments – e.g., smart homes, smart city, smart agriculture, and among others – has increased the demand for services to store and process a large amount of data generated by such applications. Initially supported by Cloud Computing, as the number of devices grew, these applications started to require more resources and turning difficult to provide uninterrupted, real-time, and low-latency services. These demands go beyond what current cloud services can offer (CHIANG; ZHANG, 2016).

The Fog computing paradigm has emerged as a candidate to meet these new IoT requirements. This paradigm proposes to extend resources, computing services, networking, storage, and control to the edge of the network – geographically closer to IoT devices (BONOMI et al., 2012; PULIAFITO et al., 2019). This concept defines an intermediate layer between the Cloud and devices for data processing and storage.

The main services offered by Cloud (i.e., computing, networking, storage, and control services) are also provided by Fog, being closer to end-users, featuring dense geographical distribution and mobility support as key advantages (STOJMENOVIC; WEN, 2014). Figure 5 illustrates a Fog Computing-based architecture that allows integrating a heterogeneous set of devices into the things layer – the region where sensors and actuators are present – to Fog layer devices.

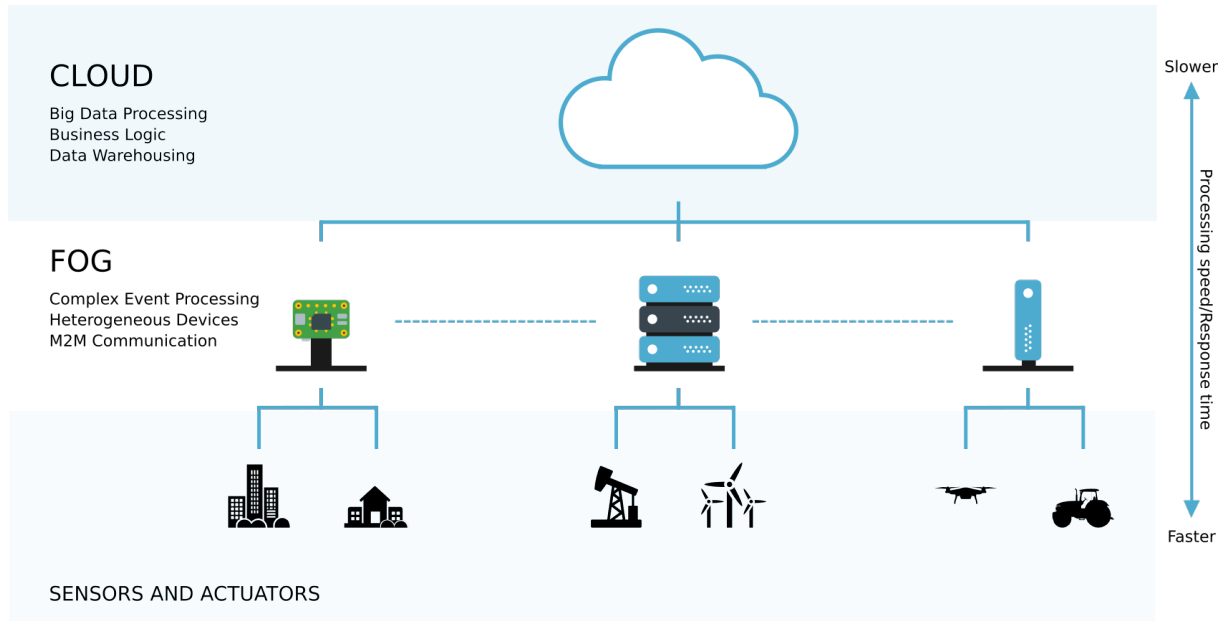


Figure 5: Fog architecture.

In turn, such Fog devices can still be interconnected and each one can connect to the Cloud, in order to integrate with computing services that can not be hosted at the Fog.

Geographically closer to end-users, Fog-based application infrastructure enables to minimize the problems faced by a large number of network and bandwidth-sensitive services by running their processes as close as possible to the data source (OKAY; OZDEMIR, 2016). With increased computing power in a region between the Cloud and devices, these services gain faster speed for storing, processing, and analyzing data, significantly reducing the amount of data sent to the Cloud. Note that processing some or all of the data coming to the Fog can significantly reduce Internet bandwidth consumption, benefiting applications that collect large amounts of data from multiple sensors and transmit over the network. (RAHMANI et al., 2018).

## 2.6 CONCLUDING REMARKS

This chapter introduced the fundamental concepts for understanding this work. In addition to the basic concepts related to Discrete Controller Synthesis, the definitions of Rules Engine and Complex Event Processing were presented. These concepts will be fundamental to the development of this work, ensuring the scalability of the proposed solution. The

idea of Fog Computing is also addressed in this work, seeking to meet the demands of IoT-based applications regarding Internet availability, bandwidth, and low latency.

### 3 RELATED WORK

This chapter presents some works related to the application of Complex Event Processing in different scenarios of the Internet of Things. In Section 3.1, we describe some of these solutions and limitations of each approach, providing some information about our solution. Section 3.2 presents some works that describe the development of controllers based on the Discrete Controller Synthesis techniques.

#### 3.1 COMPLEX EVENT PROCESSING

With the growing number of data generated in the context of IoT, the demand for applications capable of processing this data is increasing. Solutions with the potential to extract information and make decisions based on this data flow have gained an essential role in the areas of monitoring and control. In this context, Complex Event Processing (CEP) are solutions that have become increasingly popular to meet this demand, detecting patterns of interest from events, and acting in decision making. In this section, we will present some works that use CEPs in different areas.

Mazon-Olivo et al. (2018) presents a solution based on Cloud Computing to dynamically process events generated in the context of IoT and Precision Agriculture. The proposed solution integrates the CEP with a rules engine, providing control rules defined by the application user. Accordingly, for each event that arrives at the application, CEP consults the set of rules and triggers control actions. A case study integrated the solution into a cloud host with limited computing resources and integrated with an intelligent irrigation system for an experimental banana field.

Although the solution presented by Mazon-Olivo et al. (2018) is scalable and capable of serving different scenarios, relying entirely on an application centralized on a server in the Cloud can represent a limitation for applications in scenarios where the Internet connection it is weak or non-existent. In this sense, the application of this solution in these scenarios may face problems with the bandwidth necessary to travel data over the network. Besides, latency-sensitive applications may also be experiencing problems using this solution.

Some studies present strategies to reduce latency in processing large amounts of data. Cugola e Margara (2012) investigated the use of hardware parallelism to speed up CEP processing. In this work, the authors considered multi-core CPUs and CUDA (CUDA, 2019), a widespread architecture for general-purpose GPU programming. The study proposes an analysis of this solution under study case in which the CEP is applied in environmental monitoring to identify occurrences of fires from the data of geographical position, temperature, and presence of smoke data. Based on a set of rules, CEP analyzes this

data to notify users of fire occurrences. This study shows that the use of GPUs allows accelerating the processing of events in scenarios that involve complex rules, overcoming the intrinsic delay of this type of application.

The study presented by Fardbastani, Allahdadi e Sharifi (2018) proposes a Cloud-based solution for detecting fraud and security in the business context. The authors applied complex event processing to the monitoring of business processes in large organizations, resulting in a large number of monitoring rules and high event rates. To deal with latency, Fardbastani, Allahdadi e Sharifi (2018) proposed decentralized CEPs using task parallelism by decomposition and distribution of rules and data parallelism by partitioning and dispatching events.

Chen et al. (2014) proposes an architecture for a distributed CEP. The authors consider that if the data is pre-processed on the side of the gateways and sends only the essential information to the back-end servers, it is possible to reduce latency and processing load. The proposed solution also provides a rules engine with a graphical interface in which a set of rules can be dynamically defined by users to handle each event that arrives at the service. A Smart Building scenario is presented as a use case, in which a CEP controls a set of air conditioners from the sensor data to keep energy consumption within limits required by the rules.

Other works present models based on Fog Computing to deal with the problem of bandwidth consumption. Madumal, Atukorale e Usoof (2016) presents a computational model based on Fog Computing that proposes the implantation of a CEP at the edge of the network. In this proposal, the authors present a Fog gateway to the Cloud that schedules data to Fog or Cloud using a system resource rules and forecasting mechanism. This approach allows applications to achieve low latency and low bandwidth consumption responses.

Ramperez, Soriano e Lizcano (2018) proposes an architecture based on Fog Computing to process extensive data flows in the context of Smart Cities. The authors designed a Context Broker based on publish-subscribe middleware to be elastic and low latency and integrated it with CEP. The architecture was validated through a real smart city use case, showing how the proposed architecture can meet the requirements of Smart Cities, taking advantage of the Fog Computing approach.

All solutions presented seek different strategies to deal with latency and bandwidth problems. While some works bet on solutions based on the parallelization of CEPs in the Cloud, others propose to place the CEPs at the edge of the network adopting the Fog Computing paradigm, or even a hybrid approach, combining the operation between the existing CEPs in Fog and the Cloud. This latter approach takes advantage of both paradigms, taking advantage of the computing power of the Cloud and the advantages presented by Fog in terms of low latency and low bandwidth consumption.

Although related work describe all components of their solutions, it is essential to



highlight that neither solution mentions mechanisms to verify the consistency of the rules in the ER. In this sense, Chen et al. (2014) mentions only the existence of a syntactic validator of the rules. Mechanisms to verify the consistency of the rules are essential to guarantee service reliability. As the number of rules in an ER grows, it becomes increasingly difficult for users to check all possible conflicts.

Table 1 compares the related and present work. In our approach, we propose an architecture that allows the synthesis of a reactive CEP based on the techniques of Synthesis of Discrete Controllers. A RE allows users to define a set of rules that, during the synthesis process, are checked to generate a correct controller, and that meets the demands of different control scenarios. The generated CEP can be implanted in a node in the Fog, acting independently of the RE. This approach is suitable for scenarios where the Internet connection is limited or non-existent.

Table 1: Related work based on Complex Event Processing

Work	Rules checker	Deployment	CEP type
(MAZON-OLIVO et al., 2018)	No	Cloud	Reactive
(CHEN et al., 2014)	No	Cloud	Reactive
(FARDBASTANI; ALLAHDADI; SHARIFI, 2018)	No	Cloud	Reactive
(MADUMAL; ATUKORALE; USOOF, 2016)	No	Hybrid	Proactive
(RAMPEREZ; SORIANO; LIZCANO, 2018)	No	Fog	Reactive
(CUGOLA; MARGARA, 2012)	No	Cloud	Reactive
This work	Yes	Fog	Reactive

The comparison between the works presented takes into account the use of some mechanism for checking the rules defined by the users. Another criterion is the paradigm used, analyzing when the solutions are based on Cloud, Fog, or in Hybrid form between the two paradigms. We also consider the type of CEP (reactive or proactive) used by the solution. In this sense, we consider those capable of processing a pattern of complex events and generating automated decisions (reactive) or those capable of predicting events (proactive), mitigating their effects.

### 3.2 DISCRETE CONTROLLER SYNTHESIS

In the context of controller development, some works in the literature propose the development of a correct controller, – i.e., implemented with the assurance that all control rules have been verified and that the resulting controller meets all specifications. Based on the Supervisory Control Theory, these works apply the developed controller to control smart environments such as Smart Homes and Smart Buildings. This approach provides an alternative for checking the consistency of control rules.

Zhao et al. (2013) explore techniques of Supervisory Control Theory to develop a controller capable of controlling different types of devices in the context of Smart Home-/Buildings. The authors present the modeling of each component using automata formalism and define a set of rules that meets three modes of operation: comfort, energy-saving, and minimal safety of residents. In this sense, devices are coordinated by a centralized supervisor to meet the set of rules defined in the modeling. In order to validate, the authors present a case study in which a controller has been modeled using Heptagon/BZR language, synthesized and integrated with a Smart Home simulator.

Guillet, Bouchard e Bouzouane (2013) also proposes the use of DCS techniques in a Smart Home scenario, and the set of control rules caters for disabled users. In this scenario, the work presents even more specific modeling of the existing devices in each room of the environment, and the set of rules described are even more sensitive and adapted to the user, seeking to maintain the appropriate security. The authors also use the automata formalism, and the rules set is verified by the synthesis process, generating a correct controller.

In addition to smart environments, the application of these techniques has already been applied to control robots. Aboubekr et al. (2011) presents the synthesis of a controller applied to the control of a robotic arm, an approach that proposes to apply DCS techniques to design discrete control loops on top of continuous control tasks. The implementation of this solution integrates ORCCAD (ARIAS et al., 2010), a design environment for real-time control executives, and the Heptagon/BZR language. A simulation was proposed to evaluate the performance and behavior of the robotic arm controlled by the generated controller.

An et al. (2013) presents the development of a controller using DCS techniques for a class of dynamically reconfigurable embedded computing systems (FPGA). The authors considered a video processing system implemented on a platform containing an FPGA to visualize and validate the runtime reconfigurations of the FPGA controlled by a controller synthesized from a model in the Heptagon/BZR language. Unlike the works presented previously – that used simulators to validate their solutions, the approach presented by An et al. (2013) shows that the generated controller can be deployed on devices of different architectures.

More recent works also explore supervisory control theory. Sylla et al. (2018) presents modular and hierarchical structures of control, an approach that seeks to reduce the computational cost of synthesizing discrete controllers for the multi-state systems. Using the Heptagon/BZR language for modeling and the ReaX tool for controller synthesis, a case study was proposed to applies to the control of a Smart Office. Model defining rules for the control of lighting and privacy of users is presented, and the controller synthesized is deployed on Raspberry Pi to acts controlling a lamp and shutter on a prototype.

Table 2 presents the related work that use DCS techniques with applications in dif-

ferent domains.

Table 2: Related work based on Discrete Controller Synthesis

Work	Domain application	Validation
(SYLLA et al., 2018)	Smart Office	Experiments
(ZHAO et al., 2013)	Smart Home	Simulations
(GUILLET; BOUCHARD; BOUZOUANE, 2013)	Smart Building	Simulations
(AN et al., 2013)	Video Systems	Experiments
(ABOUBEKR et al., 2011)	Robotic Systems	Simulations

These works indicate the direction for the development of a correct controller capable of meeting the control demands in different contexts. The application of a DCS tool allowed the verification of all control rules defined by the models presented, generating a correct controller. Note that given the dynamics of the environments presented in these works (Smart Home and Smart Buildings), some characteristics of these environments can be extended to the context of smart farming, industry, among others. These characteristics allow the approach used to be used to develop a solution that is sufficiently generic to meet different demands.

### 3.3 CONCLUDING REMARKS

In this work, different types of paradigms and technologies will be applied in the development of the solution. CEP development process will be based on the theory of supervisory control, applying techniques of the Discrete Controllers Synthesis through the automatic generation of models in the Heptagon/BZR language. The focus of this work is on developing a solution capable of providing a CEP that meets the rules defined by multiple users and capable of processing data flows sent by different types of devices. The CEP generated by the solution must be able to be deployed on devices that make up a Fog Computing infrastructure, taking advantage of this paradigm.

## 4 PROPOSED ARCHITECTURE

Given the difficulties presented by applications based on Cloud Computing, this chapter proposes a control solution as a service based on Fog Computing. Section 4.1 looks at how low upload rates affect the quality of Cloud-based services. The section describes a scenario based on 4G technology and assesses the loss of packets sent at different transmission rates. This analysis motivates the development of the proposed solution based on Fog Computing, seeking to minimize these effects. Section 4.2 presents the proposed solution and describes each component.

### 4.1 WHY FOG-BASED APPLICATIONS?

The evolution of Cloud Computing has opened the door to an explosion of new services, taking advantage of the convenience and high performance that the cloud guarantees. Nevertheless, as already shown, the rapid evolution of IoT applications and the emergence of billions of smart devices have required increasing Internet bandwidth for data transmission (OKAY; OZDEMIR, 2016). Given the experience of users with low upload rates presented by 4G Internet carriers (FITCHARD, 2019), this problem tends to get more critical.

The results presented in Mazon-Olivo et al. (2018) have shown that a Cloud-based CEP is capable of processing large amounts of data at rates of up to 25,000 events per second. However, the authors do not analyze the impact caused by the data volume on the network, an essential resource in order to determine the feasibility of distributed computing solution's.

In order to fill this gap, we simulate the scenario presented by Mazon-Olivo et al. (2018) and how a constrained 4G network can impact the CEP. We send data packets at different rates – i.e., number of packets per second –, allowing to check how the application behaves when the upload rate is limited. The main objective is to analyze how the available bandwidth impacts the quality of services that need to transmit data at increasing quantities.

To model and simulate this scenario, we used the Network Simulator NS-3 (RILEY; HENDERSON, 2010), a discrete event simulator that lets us simulate computer networks including a broad range of components as mobile devices, base stations, routers, among others. In this scenario, data was sent by the IoT device to a server using the User Datagram Protocol through a 4G Long Term Evolution radio base station. We set the upload data rate for communication between the device and the radio base station at 5.0 Mbps, the average value presented by carriers in Brazil (FITCHARD, 2019). Figure 6 illustrates the scenario used in the simulation.

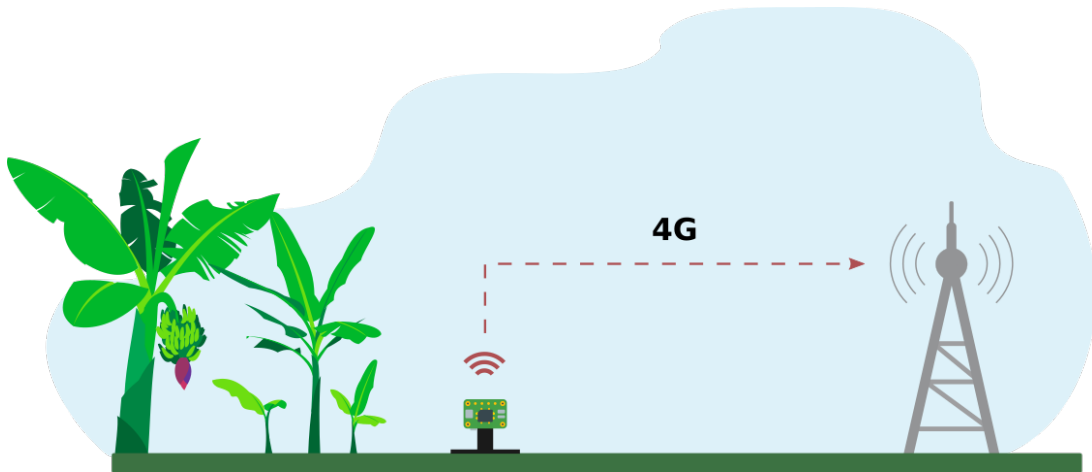


Figure 6: Scenario simulated.

The solution proposed by (MAZON-OLIVO et al., 2018) is applied in the irrigation control of banana plantations. This process takes place within two hours, and each irrigated hectare has twelve soil moisture sensors, one temperature sensor, one water pressure meter, and eight actuators (electro-valves). Mazon-Olivo et al. varied the rate of events per second in their experiments from 0.2 events/s to 25,000 events/s, in order to evaluate its CPE for different area plantations. The solution showed an acceptable CPU performance of 75.4% up to an approximate of 4,200 events/s, considering the few computing resources used to host its solution.

Considering the same scenario, we vary the number of events per second between 100 to 5,000 events/s, considering a payload of 782 bytes into each packet – the same size of the data packets sent to the control service presented in (MAZON-OLIVO et al., 2018). We measured the packet losses at each case. Results are presented on Table 3.

Table 3: Packet loss rate in a 4G application scenario

Events/s	Sent packets	Lost packets	Packet loss rate (%)
100	719990	0	0,00%
200	1439980	0	0,00%
500	3599950	240	0,01%
700	5039932	201052	3,99%
800	5759920	919132	15,96%
1000	7199900	2355296	32,71%
2000	14399800	9536116	66,22%
5000	17999750	15539288	86,33%

Under limited bandwidth, the packet loss grows as the number of events grows, becoming increasingly significant. Considering the case of 5000 events/s, which is closer to the best acceptable scenario (in terms of CPU consumption) of 4200 events/s, we observe

an impracticable packet loss of about 86%. Actually, the case of 700 events/s presented an acceptable packet loss in our simulations which corresponds to data rate from a small plantation of 40 hectares.

The impact of these losses on many applications can be costly and even endanger user safety. In a smart farm irrigation control scenario, for example, packet losses of data that signals when hydraulic bombs should be turned off can result in a waste of water or loss of productivity by under-irrigation or over-irrigation (KAMIENSKI et al., 2018). Scenarios in the context of smart cities (e.g., traffic control, security) and healthcare are also profoundly impacted by such losses, putting human lives at risk (CHIANG; ZHANG, 2016; HUANG; LU; CHOO, 2017; MUTLAG et al., 2019).

Fog Computing can reduce those losses and solve most of these problems. Since the Fog is at the network's edge, it is possible to use local network infrastructure (i.e., non-Internet connected), which dramatically reduces packet loss and latency. It also supports several IoT-ready communication protocols (e.g., SigFox, LoRaWAN, MQTT, ZigBee, and others), allowing to process data from multiple densely distributed data collection points with low power consumption (PERALTA et al., 2017). The ability to process data at the edge of the network also brings greater autonomy, security, and analysis capability for processing, making it less prone to failures (SHIRAZI et al., 2017).

## 4.2 ARCHITECTURE

Given the advantages presented by Fog Computing-based applications, we proposed a Control-as-a-Service architecture. The architecture proposed is presented in Figure 7. It consists of three general components: the Rules Engine, the Complex Event Processor Generator (CEP Generator), and the Complex Event Processor (CEP) itself.

The Rules Engine enables end-users to manage events, actions, and rules through a web interface. The CEP Generator reads rules from the Rule Engine database and generates CEPs. Finally, CEPs interface with sensors and actuators through indirect communication generating events for the controller and taking actions from it.

We implemented both the RE and the CEP Generator as Cloud applications. The CEP generated is downloaded and deployed at the Fog nodes, running on containers under high processing power computers or single-board computers (SHIRAZI et al., 2017). This strategy aims to minimize the problems faced by applications regarding low upload data rates by providing the control service closest to the devices to be controlled, avoiding Internet network issues. This section presents each component of this architecture in detail.

### 4.2.1 Rules Engine

The RE is the component that allows the user managing events based on input data (e.g., air temperature, the relative humidity of the air, and soil moisture) and actions based on

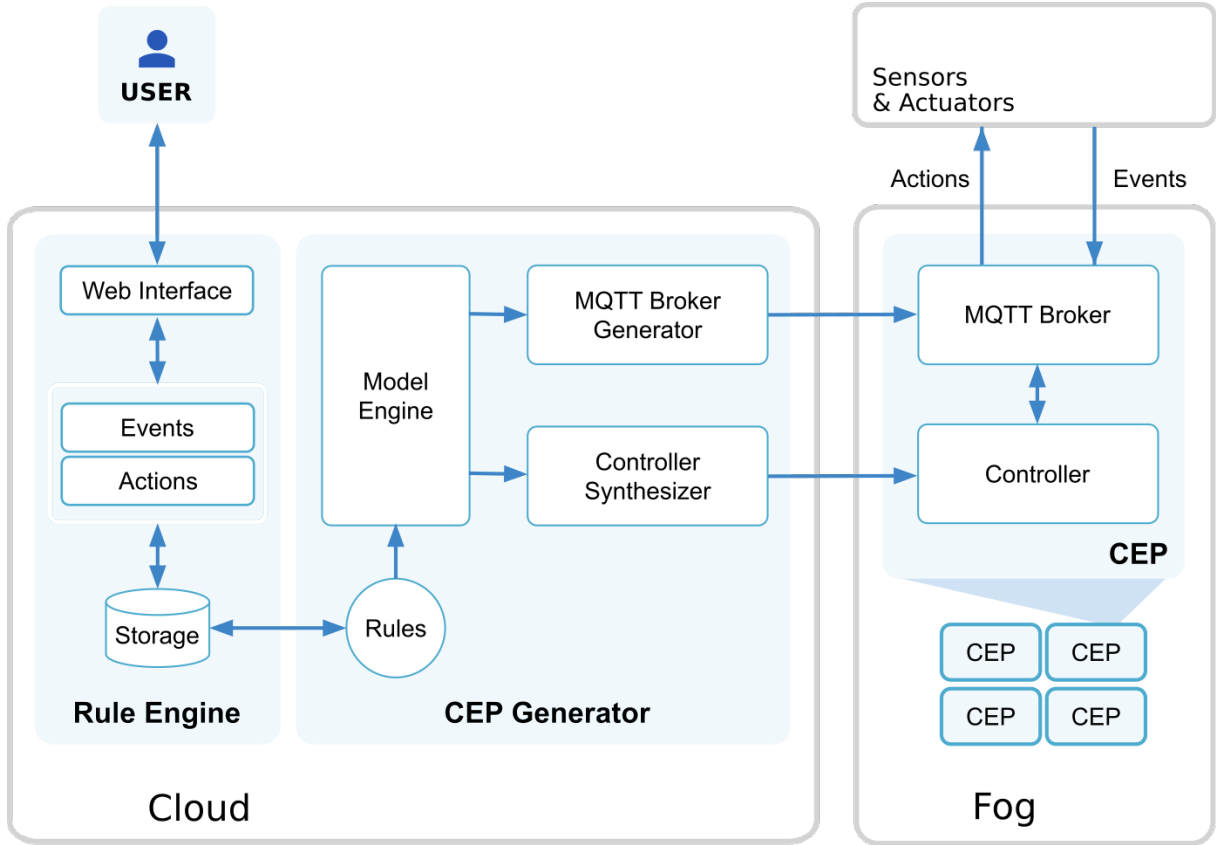


Figure 7: Proposed architecture.

actuator states (e.g., air conditioner, heaters, and solenoid valves). Through a web-based graphical user interface, end users can create events and associate them with a particular control action, defining rules that determine the system's business logic.

All events, actions, and rules are stored at a PostgreSQL database. All components were written in Python language using the Flask framework. The RE graphical interface is shown at Figure 8, where examples of events are displayed in the center of the screen. In the left corner are the menu options to navigate between the Event Manager, Rule Manager, and Actions Manager.

**Events** define the association between input variables and logical and relational operators, providing the conditions to trigger an action. All events are saved in the database and serve as input to the synthesis of CEP controllers. The **Actions** define the states of the actuators, allowing the association between them through the logical operators. In this context, our system assumes that the actuators have two well-defined states: on and off. Finally, each **Rule** establish an association between events and actions already defined. For all defined events, there will always be an associated action. Thus, an event will always trigger one specific action.

It is important to note that all rules defined in the ER have their consistency verified by Sigali during the CEP synthesis process. In this case, if there is a conflict between one or more defined rules, the synthesis process reports the existence of the conflict to the

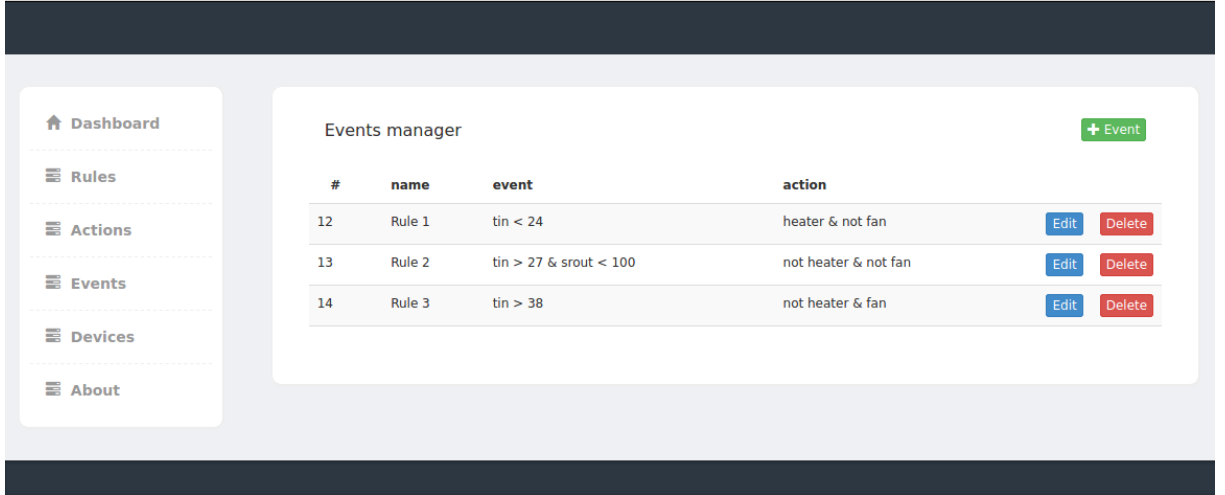


Figure 8: RE graphical interface.

user. This process of checking the rules ensures that the synthesized CEP is correct.

Expression 4.1 gives an example of a control rule. Note that, as presented in the definition, an event is composed of variables representing the values of sensors or actuators, associated by logical or relational operators. If the event is satisfied, there is an implication in executing an action of one or more actuators.

$$tin > 30 \text{ or } srout > 800 \implies \text{fan \& not heater}, \quad (4.1)$$

where *tin* and *srout* are variables representing respectively the air temperature and solar radiation, for example. In this example, that emulates a greenhouse control, when the air temperature is greater than thirty degrees and the solar radiation is greater than 800  $W \cdot m^{-2}$ , then the fan should be turned *on*, and the heater turned *off*. The term "not" preceding the *heater* variable changes the heater state to 0 (*off*). The notation presented in Expression 4.1 is based on Heptagon/BZR syntax.

#### 4.2.2 Complex Event Processor Generator

After defining the sets of events, actions, and rules, the CEP Generator automatically synthesizes the CEP based on the theory of supervisory control. The automatic synthesis process of the CEP involves: querying each of the associated events, actions, and rules; automatically constructing a Heptagon/BZR script; synthesizing the controller, which is associated to the consistency analysis of the control rules; automatically generating a MQTT broker interface for the controller.

The **Rules** extraction part is responsible for extracting all sets of events and actions that make up each rule. In this step, only the extraction is done, not analyzing the consistency of each rule.

**Model Engine** is the part capable of automatically building the Heptagon/BZR model from the events and actions associated with user-defined rules. The generated



script models each device at the smart environment as a state machine with transitions based on the business rules. Listing 4.1 shows an example of a generic device modeled using Heptagon/BZR. The model has two possible states (ON or OFF), and  $c$  represents a controlled variable.

```

node device(c: bool) returns (device_on:bool)
2 let
    automaton
4     state OFF do
        device_on = false;
6     unless c then ON
    state ON do
8     device_on = true;
        unless not c then OFF
10 end
tel

```

Listing 4.1: Device model example in Heptagon/BZR

The **Controller Synthesizer** part has the responsibility of synthesizing the CEP controller from the script generated by the Model Engine through Heptagon/BZR. In this step, Sigali evaluates the properties of stability, reachability, observability, and optimality of the smart environment model. When there is inconsistency in the user-defined rules, the application reports the need to redefine these rules, seeking to ensure that the generated CEP controller is consistent and satisfies the rules.

The **MQTT Broker Generator** is the component that automatically generates an MQTT broker for communication between the devices and the CEP controller generated by the Controller Synthesizer. From Heptagon/BZR model generated by Engine Model, a MQTT Broker is automatically generated fully adapted to the controller, reducing the use of computing resources while operating. Generated in C, the MQTT broker generated coordinates all data transfer among the CEP, sensors, and actuators.

### 4.2.3 Complex Event Processor

The CEP is the component responsible for interacting, through the MQTT broker, with sensors and actuators, receiving events from them and sending them actions. These exchanged data generally represent the environmental conditions (i.e., air temperature) or states of the devices (i.e., solenoid valve locked/unlocked ) so that the CEP controller evaluates them under the user-defined rules.

In this proposal, we assume that data are sent to CEP already pre-processed, submitted to processes of noise treatment, aggregation, or the calculation of other derived variables (e.g., evapotranspiration, an important environmental variable for irrigation that can be estimated from air temperature and other weather-related data). All data comes in the form of events, containing key-value sets that indicate the variables in the user-

defined rules. Listing 4.2 presents an example of the data structure sent to the CEP with air temperature and solar radiation data.

```

1 {
   data: [
3     {sensor: "tin", value: 39},
       {sensor: "srou", value: 1635},
5   ],
  }

```

Listing 4.2: Data format from sensors.

The **MQTT Broker** is responsible for managing publications and subscriptions, mediating the communication between the devices present in the environment to be controlled and the controller, allowing indirect communication to occur between them.

Finally, the **Controller** is the result of the synthesis process, and it is the part of the CEP responsible by react to events according to the rules. As the CEP is downloaded and deployed at the edge of the network, all rules are embedded at the Controller and there no queries to the RE on the Cloud in order to check rules and take actions.

Listing 4.3 exemplifies how CEP responds to the device that requested the control service by sending temperature and solar radiation data (as per Listing 4.2). Based on the control rule (Expression 4.1), CEP responds with actuators state – i.e., the heater must be turned off and the ventilation system must be turned on. Note that the data represents the values of air temperature (*tin*) and solar radiation (*srou*), based on JSON objects, key-value sets. Importantly, requests whose message content does not match the rules defined in the synthesis process are ignored by the CEP.

```

1 {
   data: [
2     {device: "heater", status: 0},
4     {device: "fan", status: 1},
       ],
6  }

```

Listing 4.3: Data format from service.

### 4.3 CONCLUDING REMARKS

This chapter presented a CEP architecture as a service capable of providing a CEP based on the techniques of Synthesis of Discrete Controllers from a set of rules defined in the RE. Based on the Fog Computing paradigm, the architecture allows the CEP generated by the solution to be deployed on a device at Fog, mitigating the problems faced by Cloud-based solutions.

One of the main advantages of this solution is the delivery of a correct controller, with the guarantee that it meets the specifications defined by the users through the RE. For defined rules, the controller synthesis process checks each state and the possibility of these states being reached according to those rules. This verification also considers the existence of contradictory rules, avoiding inconsistencies in the control service.

Although generic enough to be applied to different contexts, the characteristics of this solution meet the specialization requirement required from the ER. Besides, the solution allows a large number and varied types of devices to communicate with the CEP using the MQTT protocol.

To demonstrate the advantages of this solution, case studies in which it was applied will be presented in Chapter 5, generating a CEP from a set of rules for device control in the contexts of Smart Home and Smart Farming.

## 5 CASE STUDIES

This chapter presents two case studies. The main objective is to evaluate the Control-as-a-Service architecture proposed in this work applied to different scenarios. Throughout the chapter, we describe how the proposed solution can be applied to these scenarios. In the first case study presented in Section 5.1, we present a scenario applied to the context of Smart Homes. The second case study (Section 5.2), presents the application of this solution in the control of an agricultural greenhouse. In this study, we implemented a simulator capable of modeling the behavior of climate variables as the CEP acts in the control of actuators.

### 5.1 CASE STUDY I: SMART HOME

Sensing systems, in the context of Smart Home, allow collecting environmental measurements inside and outside environment, serving as parameters to control various types of devices such as air conditioners, heaters, or garden watering systems settings according to weather conditions. Presence sensors can provide data for the security system (e.g., coordinating surveillance camera activities), power management (e.g., activating or deactivating the lighting system), and water consumption (e.g., coordinating the use of water reservoirs).

One of the main issues in the context of and Smart Homes is the growing number of home appliances in residential environments, which represents a significant demand for electricity (JOO; CHOI, 2017). Among the different types of appliances, lighting, heating and cooling systems account for most of the energy consumption (CANDANEDO; FELDHEIM; DERAMAIX, 2017). Given this, we present a use case of the proposed solution in this paper to meet the demand for a power management policy in the context of and Smart Homes.

In this use case, we propose the control of a residential environment consisting of a lighting system and a cooling system. For simplicity, these systems consist of a lamp and an air conditioner, respectively. We also consider a window that can act in the lighting system or climate control. Illumination can be adjusted based on window opening and lamp lighting. For the thermal control of the environment, it is possible to use the air conditioner or open/close the window.

The environment sensing is made from a presence sensor that indicates if there is at least one person present at the place; a temperature sensor provides data to measure the user's thermal comfort; a device is used to define day and night. In addition to these sensors, we have added a noise detection system to measure ambient sound comfort.

We define four rules based on the devices and sensors available in the environment.

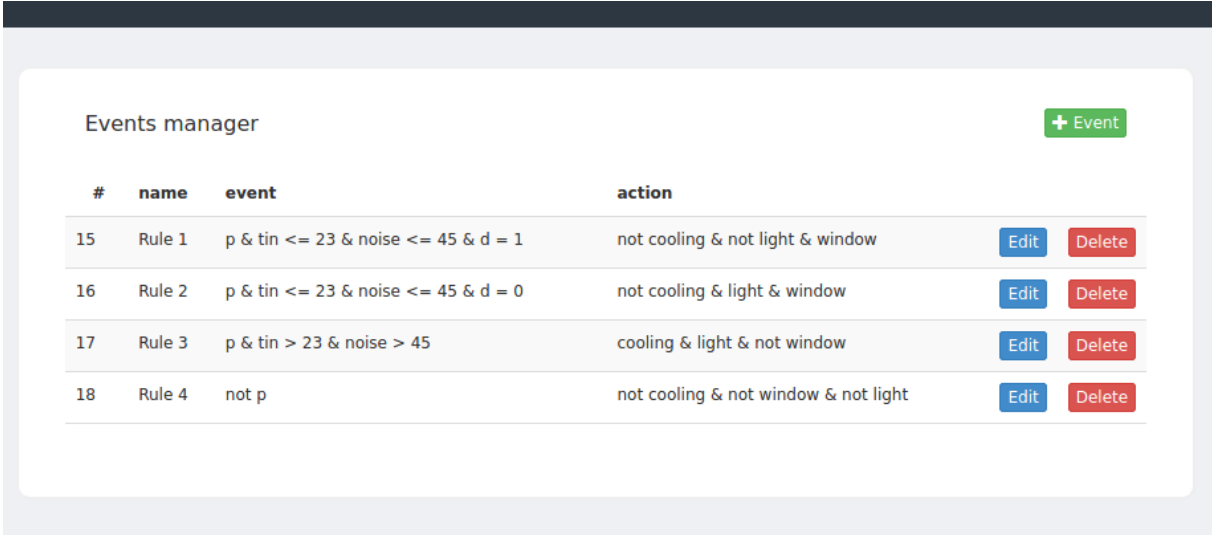
In this case, three factors were used to define each of the rules: environmental safety, energy-saving, and user comfort. Note that some rules may override the factors – i.e., the same rule may meet the energy-saving and user comfort factors.

Table 4 shows each rule, formally. The first rule establishes the case for natural cooling and lighting, promoting energy savings. In Rule 1, if there is some person in the room ( $p$ ), temperature ( $tin$ ) is less than or equal to  $23^\circ$ , there is no noise ( $noise$  under 45 dB), and it is daytime ( $d = 1$ ), then the air conditioner ( $cooling$ ) and lamp ( $light$ ) must be turned off, and the window ( $window$ ) must be opened. Complementary, when it is night ( $d = 0$ ), Rule 2 triggers artificial lighting.

Table 4: Events designed for smart home controller.

	Event	Action
Rule 1	$p \ \& \ tin \leq 23 \ \& \ noise \leq 45 \ \& \ d = 1$	not cooling & not light & window
Rule 2	$p \ \& \ tin \leq 23 \ \& \ noise \leq 45 \ \& \ d = 0$	not cooling & light & window
Rule 3	$p \ \& \ tin > 23 \ \& \ noise > 45$	cooling & light & not window
Rule 4	not $p$	not cooling & not window & not light

Rule 3 treats the case when thermal user comfort can not be achieved by natural means and it enforces the environment to turn the air conditioner on, the lamp off, and to close the window. Finally, when there is no presence, Rule 4 determines all devices to be turned off and the window to be closed for user safety and energy saving. Figure 9 shows how these rules are defined in the RE.



#	name	event	action		
15	Rule 1	$p \ \& \ tin \leq 23 \ \& \ noise \leq 45 \ \& \ d = 1$	not cooling & not light & window	Edit	Delete
16	Rule 2	$p \ \& \ tin \leq 23 \ \& \ noise \leq 45 \ \& \ d = 0$	not cooling & light & window	Edit	Delete
17	Rule 3	$p \ \& \ tin > 23 \ \& \ noise > 45$	cooling & light & not window	Edit	Delete
18	Rule 4	not $p$	not cooling & not window & not light	Edit	Delete

Figure 9: Events designed for smart home controller in the Rules Engine.

To coordinate the collecting, preprocessing, and sending data to CEP, we use a microcontroller capable of sending and receiving data using MQTT protocol. In this case, we choose the Esp8266 NodeMCU-12E microcontroller. This microcontroller coordinates the

collection of temperature, noise measurements, time of day, performs preprocessing of all these data, and sends it to the CEP. This microcontroller is also the device responsible for receiving the control signals given by the CEP and retransmitting it to control the window, air conditioner, and lamp. In this use case, we assume that the microcontroller uses a three-channel relay to communicate with these devices, turning each device on/off according to the received control signals.

Hosted on a Fog node, the CEP responds with the states of each actuator. Listing 5.1 and 5.2 shows the format of the messages received by the CEP (Listing 5.1) and the format of messages send by the CEP (Listing 5.2).

```

2 {
  data: [
4     {sensor: "tin", value: 22},
      {sensor: "noise", value: 31},
6     {sensor: "p", value: 1},
      {sensor: "d", value: 1},
8   ]
}
```

Listing 5.1: Data from device.

```

2 {
  data: [
4     {device: "cooling", value: 0},
      {device: "light", value: 0},
6     {device: "window", value: 1},
      ]
8 }
```

Listing 5.2: Data from CEP.

The message presented in Listing 5.2 shows the control signals for each actuator. Processed by the microcontroller, the control signals are relayed to the actuators.

## 5.2 CASE STUDY II: SMART FARMING

In the context of Smart Farming, it is possible to identify several scenarios in which the application of Control-as-a-Service can offer farmers ideal conditions to optimize their production. In greenhouse-based production, for example, farmers need to monitor the environment and control various devices – i.e., heater, air humidifier, exhaust fans, irrigation system, among others – to match air temperature, relative humidity, and soil moisture according to crop needs. These climate variables must be controlled to meet the demands of each crop type, increasing productivity and allowing farmers to meet the off-season demands (SCHREINEMACHERS et al., 2016).

This section presents a case study aimed at analyzing the efficiency of the control service applied to an agricultural greenhouse. In this sense, control rules are presented to synthesize a CEP that should act in the control of devices inside the greenhouse, analyzing environmental data and redefining the states of actuators based on the requirements of each crop. For this task, mathematical models were used for the development of a greenhouse simulator. Also, meteorological data were collected for use in the simulation and control process. Section 5.2.1 presents the mathematical models for air temperature and absolute humidity. Section 5.2.2 presents the data, control rules, and settings used in the greenhouse. Finally, Section 5.2.4 presents the results obtained for this case study.

### 5.2.1 Mathematical model

The simulation uses a mathematical model that describes the environmental conditions inside a greenhouse with homogeneous properties of air temperature and absolute humidity using first-order differential equations presented by Fitz-Rodríguez et al. (2010). This model is based on the energy and mass balance equations, following the models proposed by Takakura (1976) and Takakura e Fang (2002).

There are several more sophisticated models for simulation of greenhouses. However, these models tend to be complex to implement and are not generic enough to analyze the behavior of different greenhouse designs under different weather conditions. Besides, the model presented by Fitz-Rodríguez et al. (2010) presents parameters that represent the equipment capable of acting in environmental control, making it possible to actuate a controller that automatically redefines these parameters from the input of weather data generated by the simulation process, being appropriate to the objectives of this work.

The rate of change of the air temperature inside the greenhouse depends on the heat produced by solar radiation in the greenhouse and the heat gain and loss ratios that are directly influenced by the greenhouse structure, type of cover, ventilation, cooling, heating, and plant evapotranspiration effect (FITZ-RODRÍGUEZ et al., 2010). Based on these characteristics, Equation 5.1 describes the air temperature inside the greenhouse:

$$\frac{dT_{in}}{dt} = \frac{1}{C_p \cdot \rho \cdot H} \cdot Q_{GRin} + Q_{Heater} - L \cdot E - (T_{in} - T_{out}) \times (q_v \cdot C_p \cdot \rho + w \cdot k) \quad (5.1)$$

where  $C_p$  is the specific heat of air ( $J \cdot kg^{-1} \cdot K^{-1}$ );  $H$  is the greenhouse height ( $m$ );  $T_{in}$  ( $^{\circ}C$ ) and  $T_{out}$  ( $^{\circ}C$ ) are the air temperature inside and outside of the greenhouse, respectively;  $Q_{GRin}$  ( $W \cdot m^{-2}$ ) is the solar radiation inside the greenhouse;  $k$  ( $J \cdot m^{-2} \cdot ^{\circ}C^{-1} \cdot s^{-1}$ ) is the heat transmission coefficient of glazing;  $E$  is the evapotranspiration rate inside the greenhouse;  $L$  ( $J \cdot kg^{-1}$ ) is the latent heat of vaporization of water;  $q_v$  is the ventilation rate ( $m^3 \cdot m^{-2} \cdot s^{-1}$ );  $Q_{Heater}$  ( $W \cdot m^{-2}$ ) is the heat flux from the heating system;  $w$  is the ratio of glazing surface to floor surface; and  $\rho$  ( $kg \cdot dry\ air \cdot m^{-3}$ ) is the specific mass of air.

The solar radiation absorbed inside the greenhouse ( $Q_{GRin}$ ) is a variable that acts on the production of heat, which depends on the external solar radiation and the type of material that covers the greenhouse structure. The material used in the coverage defines the values of transmittance and reflectance of the solar radiation on the floor surface (FITZ-RODRÍGUEZ et al., 2010). Thus, the radiation inside the greenhouse is estimated by

$$Q_{GRin} = \tau_c \cdot (1 - \rho_g) \cdot Q_{GRout}, \quad (5.2)$$

where  $\tau_c$  is the transmittance of the type of material used in the greenhouse cover,  $\rho_g$  is the light reflection capacity given by the chosen material, and  $Q_{GRout}$  ( $W \cdot m^{-2}$ ) is the

external solar radiation. It is important to highlight that  $\tau_c$  and  $\rho_g$  are dimensionless constants and act to define the amount of light inside the greenhouse and are specific to each type of material that surrounds the environment.

The presence of plants exposed to the local environmental conditions directly influences the evapotranspiration variable that acts on the heat exchange coefficient.  $E$  is the combination of the processes of water loss by evaporation from the soil and transpiration from vegetation. In this simulation,  $E$  is estimated based on the value of external solar radiation and the transmittance of the material (FITZ-RODRÍGUEZ et al., 2010). Thus, evapotranspiration is defined by the Equation 5.3:

$$E = 0.00006 \cdot \tau_c \cdot Q_{GRout} + 0.0004. \quad (5.3)$$

Note that  $Q_{GRin}$  and  $Q_{Heater}$  act by heating the greenhouse. On the other hand, the variables  $L$ ,  $E$  and the difference between  $T_{in}$  and  $T_{out}$  work on heat reduction. Importantly, the variables  $H$ ,  $q_v$ ,  $C_p$ ,  $\rho$ ,  $w$ ,  $Q_{Heater}$ , and  $k$  are defined based on the greenhouse settings, which will be presented in Subsection 5.2.3.

The absolute humidity model for the greenhouse is based on the base relationship between absolute humidity and water vapor flow (FITZ-RODRÍGUEZ et al., 2010). In this context, we consider evapotranspiration as a source of greenhouse water vapor gain and ventilation rate as the sole cause of water vapor loss. Thus, absolute air humidity is defined by Equation 5.4:

$$\frac{dW_{in}}{dt} = \frac{1}{\rho \cdot H} \cdot (E - (W_{in} - W_{out}) \cdot q_v \cdot \rho), \quad (5.4)$$

where  $W_{in}$  and  $W_{out}$  are the absolute humidity of the air inside and outside the greenhouse, respectively.  $E$  and  $q_v$  are the evapotranspiration and ventilation rate, and  $\rho$  is the light reflection capacity of the material used to cover the greenhouse.

## 5.2.2 Data and method

For the simulator implementation, the greenhouse settings were defined based on the most common practices of protected cultivation in Brazil. These settings involve choosing the shape of the structure used, the type of material, and the selection of actuators. These characteristics directly influence the volume of the existing air mass and the amount of solar radiation absorbed inside the greenhouse. Also, data collected from a weather station in the four seasons served as input to the model. Every minute, air temperature and solar radiation data were sent to the controller that returned with the status of each actuator.

### 5.2.2.1 Data

The data used in this experiment present hourly measurements of air temperature ( $^{\circ}\text{C}$ ), absolute humidity of air ( $g_{water}kg_{dry}^{-1}$ ), relative humidity of air (%), and solar radiation



( $\text{W} \cdot \text{m}^{-2}$ ) for the days mar/21/2018, jun/22/2018, set/24/2018, and dec/23/2018 in the latitude  $-8.059280$  and longitude  $-34.959239$ , geographic coordinates of Recife city, Pernambuco, in the Northeastern of Brazil. These data provide the outside environmental conditions as input to the simulation and allow analyzing the efficiency of the generated controller according to Recife's climate variation during the year.

The measurements of air temperature, relative humidity of air, and solar radiation were obtained from National Institute of Meteorology (Instituto Nacional de Meteorologia - INMET)(INMET, 2011) for one day (24 hours) at each of the four seasons in the Southern Hemisphere, i.e., autumn, winter, spring, and summer, respectively.

Importantly, INMET weather data service does not provide absolute humidity measurements. In this sense, it was necessary to calculate this measure from the data already collected. For this task, we used Equation 5.5 (LI et al., 2019):

$$AH = \frac{6.112 \times e^{\left(\frac{17.67 \times T}{T+243.5}\right)} \times RH \times 2.1674}{273.15 + T}, \quad (5.5)$$

which is accurate within 0.1% in the temperature range from  $-30^\circ\text{C}$  to  $+35^\circ\text{C}$ .  $AH$ ,  $RH$ , and  $T$  are the absolute humidity, relative humidity and air temperature, respectively.

Finally, hourly data were interpolated using Lagrange's polynomial interpolation method (RULES, 1995) to obtain measurements per unit of second.

#### 5.2.2.2 Rules

The set of rules defined for this case study were based on the thermal requirements of small plants adapted to the hot climate, allowing, through the existing actuators in the greenhouse, to maintain the temperature at the better range for this type of crop. Cultures such as cantaloupe, cucumber, and squash watermelon require temperature ranges between  $27$  to  $38^\circ\text{C}$  during the day and  $24$  to  $27^\circ\text{C}$  overnight (FERNANDES; COSTA; LEMOS, 2018). Considering these intervals, Table 5 defines the rules used in this case study. The variables  $tin$  and  $srou$  are the air temperature inside the greenhouse and solar radiation outside the greenhouse, respectively.

Table 5: Events designed for greenhouse controller.

	Event	Action
Rule 1	$tin < 24$	heater & not fan
Rule 2	$tin > 27$ & $srou < 100$	not heater & not fan
Rule 3	$tin > 38$	not heater & fan

Note that when the internal temperature is below  $24^\circ\text{C}$  (Rule 1), we assume the need to turn the heater on and to turn off the exhaust fan to maintain the temperature inside of the greenhouse between  $24^\circ\text{C}$  and  $27^\circ\text{C}$ . This way, we can maintain the air temperature

inside the greenhouse overnight, which is the period of the day when the temperatures fall below to 24°C in Recife.

Overnight ( $\text{srout} < 100$ ), if the temperature exceeds 27°C then all actuators must be switched off (Rule 2), causing only natural ventilation to act. We consider that the first minutes of dawn and dusk (up to 100  $\text{Wm}^{-2}$  for radiation) can still be considered night. This rule considers that this low level of solar radiation will cause the internal and external air temperature to be equal. During the day, when the temperature exceeds 38°C (Rule 3), the controller turns on the exhaust fan and turns off the heater, in order to maintain the internal temperature between 27°C and 38°C.

Figure 10 presents the definition of these rules by the user using the Rules Engine.



The screenshot shows a web interface titled "Events manager" with a green "+ Event" button in the top right corner. Below the title is a table with four columns: "#", "name", "event", and "action". There are three rows of data, each with "Edit" and "Delete" buttons to its right.

#	name	event	action	
12	Rule 1	tin < 24	heater & not fan	<a href="#">Edit</a> <a href="#">Delete</a>
13	Rule 2	tin > 27 & srout < 100	not heater & not fan	<a href="#">Edit</a> <a href="#">Delete</a>
14	Rule 3	tin > 38	not heater & fan	<a href="#">Edit</a> <a href="#">Delete</a>

Figure 10: Events designed for greenhouse controller in Rules Engine.

### 5.2.3 Greenhouse setup

It is important to note that from the mathematical model used it is possible to define a number of scenarios to experiment. However, setting all possible scenarios requires analyzing all possible control combinations for the mathematical model in question, requiring analysis of a total of 311,040 possible combinations (FITZ-RODRÍGUEZ et al., 2010). Since this is not the main objective of this work, we chose to analyze a specific scenario.

A greenhouse with A-frame structure and polyethylene cover has its height, the ratio of glazing surface to the floor surface, the heat transmission coefficient of glazing, and transmittance defined as  $H = 4$ ,  $w = 2.2$ ,  $k = 6.2$ ,  $\tau_c = 0.87$ , respectively (FITZ-RODRÍGUEZ et al., 2010). In addition,  $\rho_g = 0.5$  is presented as the light reflection capacity for polyethylene toppings.

The actuators in the environment are the main elements responsible for the control of climate variables. In this context, we consider the existence of ventilation devices and heaters, allowing the exchange of heat with the external environment or acting in the

generation of heat in the internal environment, respectively. For the scenario in question, we do not consider the existence of air cooling systems.

The settings for the ventilation are based on their ability to change the ventilation rate ( $q_v$ ) of the greenhouse, representing the air exchanges per hour (FITZ-RODRÍGUEZ et al., 2010). Thus, we consider that when the actuators are off, only natural ventilation occurs, being  $q_v = 0.014$  for structures in the A-frame format. When powered on, each actuator is able to modify  $q_v$  at a rate of 0.086. In this sense, we define  $q_v = 0.086$ , corresponding to the performance of one exhaust fan. Finally, a heater (30,000 W) was included in the simulation. Each heater influences the environment based on the relationship between the potency and the area of the greenhouse (FITZ-RODRÍGUEZ et al., 2010). Thus, considering a single heater, the variable  $Q_{Heater}$  is given by:

$$Q_{Heater} = \frac{H_{cap}}{A_{fl}}, \quad (5.6)$$

where  $H_{cap}$  is the potency of the heater, and  $A_{fl}$  is the area of the greenhouse floor surface. In this context,  $H_{cap} = 30,000$ , and  $A_{fl} = 300$ .

## 5.2.4 Results

In this Section, the simulation results obtained from the application of the controller to the greenhouse are presented. From the numerical solution of Equations 5.1 and 5.4, it was possible to obtain a dynamic response of the temperature and absolute humidity conditions inside the greenhouse based on the configurations presented in Section 5.2.2. As the simulation requests the controller service, the service returns the status of each actuator according to the rules defined.

The greenhouse simulation was performed in an Intel Core i5 4200U, 8GB RAM 1600MHz, and 1TB in a Python script. The CEP deployment was done on a virtual machine with 1.60GHz single-core CPU and 1GB RAM. For each second of simulation, the measurements of air temperature and absolute humidity were recorded.

Figure 11 presents the results for simulations at 24-hour intervals in different seasons of the year. The red lines represent the air temperature inside the greenhouse considering that the air changes are performed only by natural ventilation, without the devices acting. The green lines represent the air temperature in the scenario where the heater and exhaust fan are operated according to the rules shown in the Table 5. Finally, the blue lines represent the temperature of the air outside the greenhouse, obtained from INMET.

Note that at all seasons of the year, in the absence of control, for most of the daytime - between 8:00 and 17:00 -, the air temperature inside the greenhouse ranges from 40°C to 65°C, which is above the upper threshold defined as ideal (38°C).

Observing the conditions inside the greenhouse in the controlled scenario, the air temperature inside of the greenhouse maintains within the ideal range, producing a much

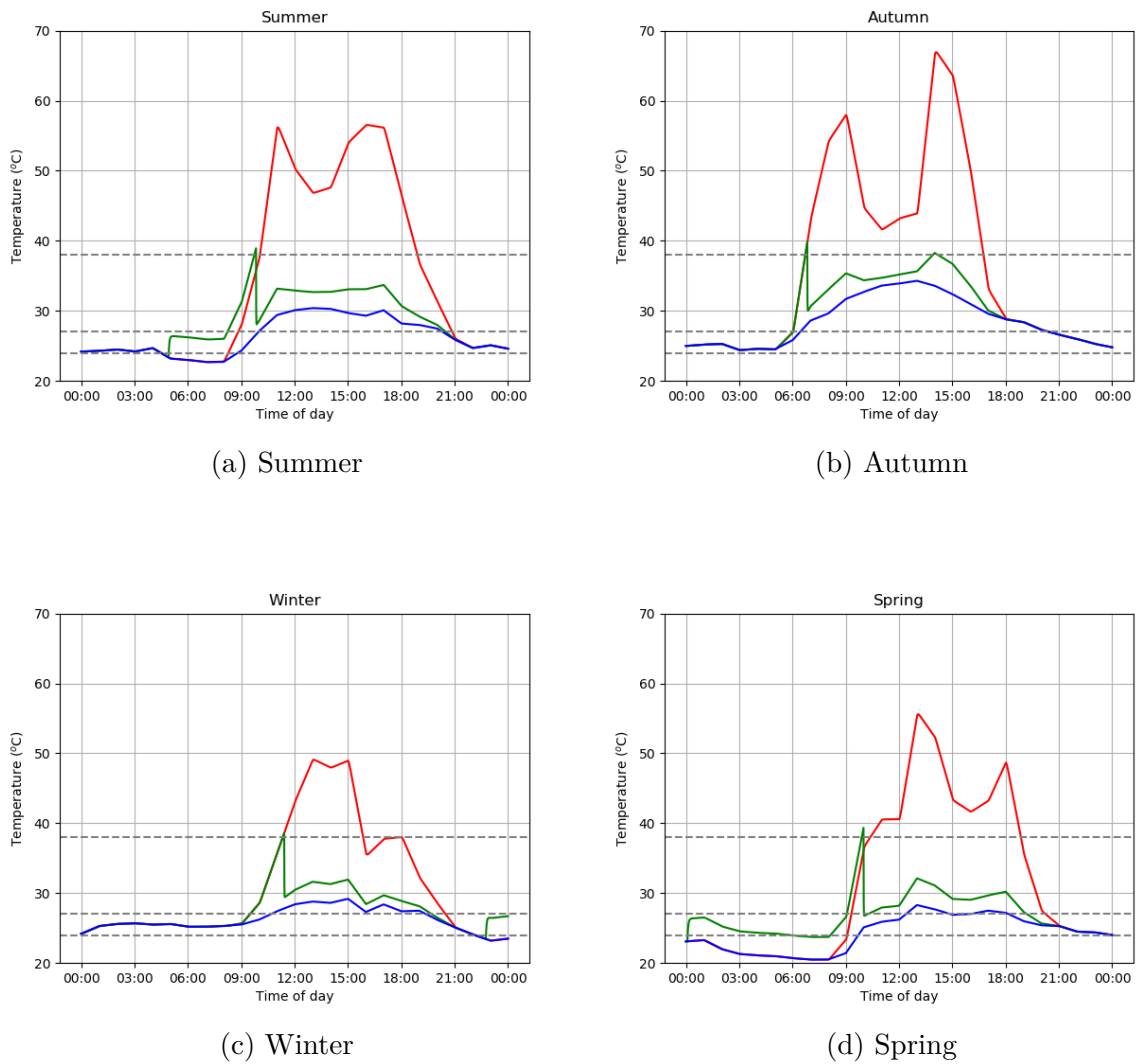


Figure 11: Simulation results for air temperature: The red lines indicates the temperature inside the greenhouse without control; the green lines represent the temperature inside the greenhouse when the CEP is acting; and the blue lines present the air temperature outside the greenhouse.

more comfortable climate for the protected crop. Another interesting fact is that in the absence of radiation the internal temperature tends to be equal to external temperature. However, the external temperature is not always within the ideal thresholds for the crop. Note that in Figure 11 (d) the outside air temperature between 00:00 and 10:00 is less than ideal. The same can be observed at a smaller interval in Figure 11 (a) (between 5:00 and 9:00) and (d) (between 22:00 and 00:00). In these cases, when there is no control, the inside air temperature tends to be lower than the lower threshold, following the outside air temperature. Thus, the activation of the heater allows the temperature at night to also remain at the expected range.

Figure 12 presents the absolute humidity measurements obtained as a result of the simulation. As in Figure 11, the red lines represent the absolute humidity inside the

greenhouse considering that the air changes are performed only by natural ventilation, without the devices acting. The green lines represent the absolute humidity in the scenario where the heater and exhaust fan are operated. Finally, the blue lines represent the absolute humidity outside the greenhouse, obtained from Equation 5.5.

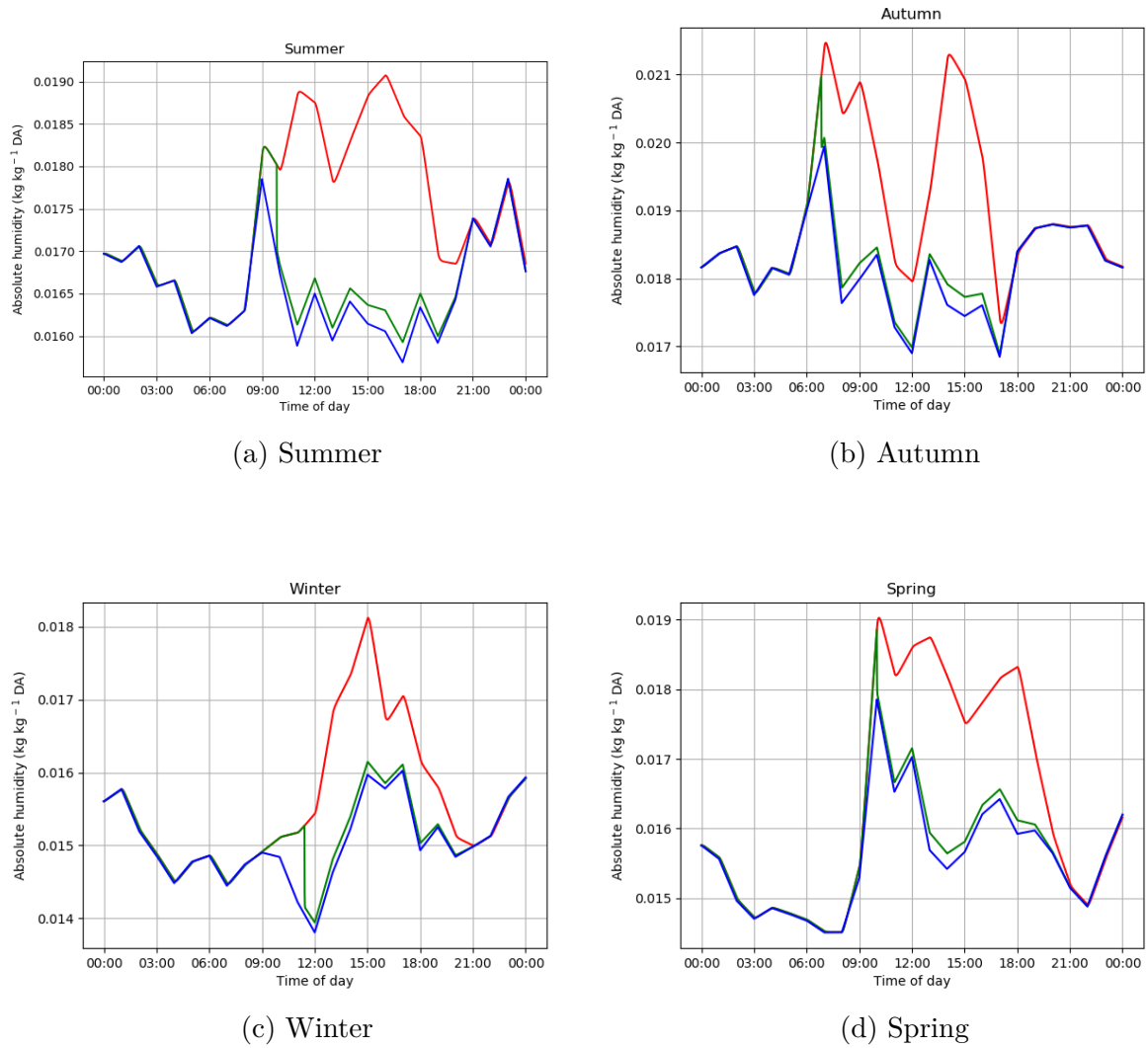


Figure 12: Simulation results for absolute humidity: the red lines indicates the absolute humidity inside the greenhouse without control; the green lines represent the absolute humidity inside the greenhouse when the CEP is acting; and the blue lines present the absolute humidity outside the greenhouse.

Note that CEP's role in controlling the temperature inside the greenhouse also directly influences absolute humidity measurements. In the uncontrolled scenario, we can see that at warmer times (between 8:00 and 17:00), the increase in water loss due to plant evapotranspiration is significant. With the action of the CEP (green lines), water loss is controlled, peaking during the early morning hours in summer, autumn and spring due to the incidence of solar radiation after a long period without exposure to light - i.e., at night.

Knowing the absolute humidity and temperature data obtained in the simulation, it is possible to define the relative humidity inside the greenhouse using Equation 5.5. Figure 13 presents the results obtained for relative humidity. As in Figure 12, the red lines represent the relative humidity inside the greenhouse, considering that the air changes are only performed by natural ventilation, without the devices acting. The green lines represent the relative humidity in the scenario where the heater and exhaust fan are operating. The blue lines represent the relative humidity outside the greenhouse.

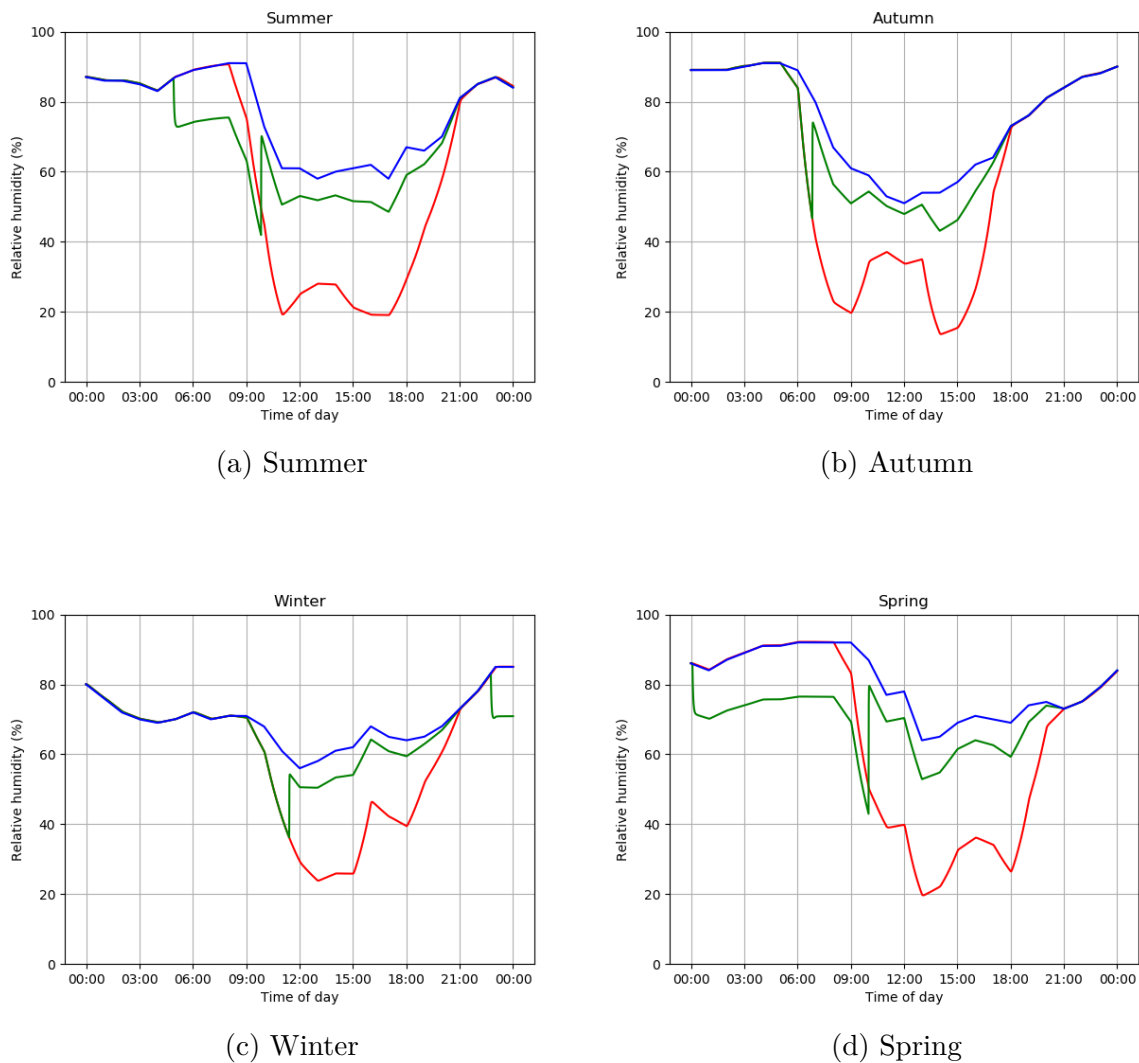


Figure 13: Simulation results for relative humidity: the red lines indicates the relative humidity inside the greenhouse without control; the green lines represent the relative humidity inside the greenhouse when the CEP is acting; and the blue lines present the relative humidity outside the greenhouse.

It is important to note that observing the behavior of the relative humidity and absolute humidity variables inside the greenhouse is extremely important because it is only possible to obtain the best plant growth within a specific humidity range. Very high relative humidity favors fungal growth, causing disease in the crop. A dry environment slows

down plant growth. The most favorable relative humidity depends on the cultivated plant species, with a typical range ranging from 60% to 90% (PONCE et al., 2014). The relative humidity measurements meet the requirements of crops such as cucumber (70% to 90%) and watermelon (65% to 75%) (PONCE et al., 2014). In this regard, note that the results obtained from CEP's performance kept the relative humidity within the acceptable thresholds.

Although there is a correlation between relative humidity and absolute humidity, the temperature variation significantly affects the relative humidity inside the greenhouse, acting on this variation more than the absolute humidity. Another interesting fact is that the higher the temperature, the lower the degree of relative humidity, and the higher the intensity of evapotranspiration, as can be seen comparing the Figures 12 and 13.

### 5.3 CONCLUDING REMARKS

The case studies presented in this chapter demonstrate how the control-as-a-service architecture can be applied to the Smart Home and Smart Farming scenarios, providing a CEP hosted on a device at Fog to meet the control demands of these environments. Based on rules defined for each scenario, a CEP was synthesized and deployed on a device in the Fog.

Note that in both scenarios, the devices to which each CEP was hosted have low computational power. However, these configurations are suitable for the operation of the CEP, demonstrating that this solution can be applied using a low-cost infrastructure. This feature extends to the microcontroller, sensors, and actuators presented in the Smart Home scenario. By using an MQTT communication protocol, any device capable of communicating using the same protocol can be used to implement the solution.

In the Smart Farming scenario, the results demonstrate the efficiency of the generated CEP. Based on the control strategy adopted in the experiment, CEP maintained the temperature within limits required by the type of crop. Besides, this strategy ensured that the relative humidity measurements were adequate for plant development. In the same sense, the absolute humidity measured by the simulator points to a reduction in evapotranspiration measures compared to the uncontrolled scenario, being strategic for reducing the consumption of water for irrigation.

## 6 CONCLUSION

This work proposes a Control-as-a-Service architecture that automatically synthesizes controllers for a smart environment using the techniques from the DCS area. Each controller is a Fog Computing-based CEP, which promotes scalable and reliable communication. In this context, while the RE allows the user-defined rules to be generated on the Cloud, the CEP is responsible for processing the data that arrives at the service in the Fog, responding with control actions associated with the rules previously defined.

Given the network limitations of Cloud-based IoT services, we presented an analysis of how a constrained network impacts control applications that need to send a large amount of data to Cloud-based services. Our analysis shows that the number of packets lost is considerable and represents a relevant problem in the face of the current demand for IoT applications that requires increasing event processing.

To demonstrate the main advantages of our architecture, we presented possible scenarios in the context of Smart Farming and Smart Home, to which the proposed architecture can be applied. In the Smart Farming context, this architecture was implemented and applied to the control of agricultural greenhouses, defining the states of the actuators based on the input data. A simulator allowed to analyze different scenarios to verify the behavior of the generated CEP.

The results presented in the control of agricultural greenhouses allowed us to analyze the efficiency of the generated CEP. Based on the set of defined rules, CEP maintained the temperature inside the greenhouse according to the limits required by the type of crop. The control strategy adopted in this scenario also positively influenced the measures of absolute humidity and relative humidity, directly influencing the adequate development of the plants. These results also imply the correctness analysis of the generated CEP, which met the rules defined in the RE.

In both scenarios, CEP deployment was proposed in devices with low computational power. Besides, the use of the MQTT communication protocol allows different types of devices to be used in applications. These characteristics demonstrate that the solution can be applied in low-cost scenarios in terms of infrastructure, allowing deployment in environments where there is little or no Internet availability.

### 6.1 LIMITATIONS

The modeling process requires some assumptions to be made that limit the results and may not fully represent reality. The CEP synthesis process, for example, assumes that all actuators used in control scenarios have only two states: on or off. To meet scenarios in which devices have multiple states, it is possible to integrate the proposed Rules Engine



with IoT platforms capable of describing multistate devices. From the description of these devices, new Heptagon/BZR models can be modeled automatically by Model Engine, allowing new rules to consider all states that describe the functioning of the device.

## 6.2 CONTRIBUTIONS

As the main contributions of this work, we may highlight:

- Propose a Fog Computing-based Control-as-a-Service architecture that matches Complex Event Processors from a set of rules defined in a Rules Engine.
- Present the application of a rules check mechanism for the synthesis of correct controllers.
- A greenhouse simulator implementation based on a mathematical model that describes the environmental conditions inside a greenhouse with homogeneous properties of air temperature and absolute humidity using first-order differential equations.
- An analysis, through simulations, of the control service application in the Smart Farming scenario.
- An analysis of the impact of low bandwidth and availability of the Internet for Cloud Computing-based IoT applications.

## 6.3 PUBLICATIONS

During the dissertation, some articles and collaborations were developed. As first author, a paper in the 10th Workshop of Applied Computing for the Management of the Environment and Natural Resources (WCAMA) 2019 presented a study of the availability of all meteorological stations of the National Institute of Meteorology - INMET installed in the Brazilian territory in the year 2017. This paper was selected as the Best Paper in WCAMA 2019 and published also in 2019, in an extended form, in the *Revista Brasileira de Computação Aplicada (RBCA)*.

In the context of this dissertation, two articles were produced and submitted to, respectively, the *Computers and Electronics in Agriculture* journal and the *Simulation Modelling Practice and Theory* journal. As non-first author, it is worth to mention the collaboration in the development of a cloud service for spatial data interpolation registered by the National Institute of Industrial Property (Instituto Nacional da Propriedade Industrial - INPI) under the Process number BR512019002705-1.

## 6.4 FUTURE WORKS

Future work will address the challenge of synthesizing multistate CEPs from the rules defined in the RE. In this context, the evolution of the rules mechanism developed in this work is necessary and can even be integrated with IoT platforms capable of describing multistate devices.

Performance comparison of the synthesized CEP against the proposal presented by Mazon-Olivo et al. (2018) can also be studied. In this study, the computational cost of the synthesis process of CEPs will also be analyzed, seeking information on the cost of implementing the synthesis service.

Finally, it is proposed to apply the proposed architecture in a real scenario, allowing us to analyze all the implementation difficulties.

## REFERENCES

- ABOUBEKR, S.; DELAVAL, G.; PISSARD-GIBOLLET, R.; RUTTEN, E.; SIMON, D. Automatic generation of discrete handlers of real-time continuous control tasks. *IFAC Proceedings Volumes*, Elsevier, v. 44, n. 1, p. 786–793, 2011.
- AKBAR, A.; KHAN, A.; CARREZ, F.; MOESSNER, K. Predictive Analytics for Complex IoT Data Streams. *IEEE Internet of Things Journal*, v. 4, n. 5, p. 1571–1582, out. 2017. ISSN 2327-4662.
- AL YAMI, M.; SCHAEFER, D. Fog computing as a complementary approach to cloud computing. In: *2019 International Conference on Computer and Information Sciences (ICCIS)*. [S.l.: s.n.], 2019. p. 1–5.
- AN, X.; RUTTEN, E.; DIGUET, J.-P.; GRIGUER, N. L.; GAMATIÉ, A. Discrete control for reconfigurable fpga-based embedded systems. *IFAC Proceedings Volumes*, Elsevier, v. 46, n. 22, p. 151–156, 2013.
- ARIAS, S.; BOUDIN, F.; PISSARD-GIBOLLET, R.; SIMON, D. ORCCAD, robot controller model and its support using Eclipse Modeling tools. In: *5th National Conference on Control Architecture of Robots*. Douai, France: [s.n.], 2010. Disponível em: <<https://hal.inria.fr/inria-00482559>>.
- ASGHARI, P.; RAHMANI, A. M.; JAVADI, H. H. S. Internet of Things applications: A systematic review. *Computer Networks*, v. 148, p. 241–261, jan. 2019. ISSN 1389-1286.
- BECK, H.; DAO-TRAN, M.; EITER, T. Lars: A logic-based framework for analytic reasoning over streams. *Artificial Intelligence*, Elsevier, v. 261, p. 16–70, 2018.
- BONOMI, F.; MILITO, R.; ZHU, J.; ADDEPALLI, S. Fog computing and its role in the internet of things. In: ACM. *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. [S.l.], 2012. p. 13–16.
- BOYLAND, P. *Ecuador, July 2019, Mobile Network Experience [WWW Document]*. URL <<https://www.opensignal.com/reports/2019/07/ecuador/mobile-network-experience>> (accessed 10.4.2019). 2019.
- BRANDIN, B. A.; CHARBONNIER, F. The supervisory control of the automated manufacturing system of the aip. In: IEEE. *Proceedings of the Fourth International Conference on Computer Integrated Manufacturing and Automation Technology*. [S.l.], 1994. p. 319–324.
- CANDANEDO, L. M.; FELDHEIM, V.; DERAMAIX, D. Data driven prediction models of energy use of appliances in a low-energy house. *Energy and buildings*, Elsevier, v. 140, p. 81–97, 2017.
- CANO, J.; DELAVAL, G.; RUTTEN, E. Coordination of ECA Rules by Verification and Control. In: KÜHN, E.; PUGLIESE, R. (Ed.). *Coordination Models and Languages*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. v. 8459, p. 33–48. ISBN 978-3-662-43375-1 978-3-662-43376-8.

- CARDUCCI, C. G. C.; MONTI, A.; SCHRAVEN, M. H.; SCHUMACHER, M.; MUELLER, D. Enabling esp32-based iot applications in building automation systems. In: *2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0 IoT)*. [S.l.: s.n.], 2019. p. 306–311. ISSN null.
- CASSANDRAS, C. G.; LAFORTUNE, S. *Introduction to discrete event systems*. [S.l.]: Springer Science & Business Media, 2009.
- CASTILLO-CARA, M.; HUARANGA-JUNCO, E.; QUISPE-MONTESINOS, M.; OROZCO-BARBOSA, L.; ANTÚNEZ, E. A. FROG: A robust and green wireless sensor node for fog computing platforms. *Journal of Sensors*, Hindawi Limited, v. 2018, p. 1–12, 2018.
- CHEN, C. Y.; FU, J. H.; SUNG, T.; WANG, P.; JOU, E.; FENG, M. Complex event processing for the internet of things and its applications. In: *2014 IEEE International Conference on Automation Science and Engineering (CASE)*. [S.l.: s.n.], 2014. p. 1144–1149. ISSN 2161-8089.
- CHIANG, M.; ZHANG, T. Fog and iot: An overview of research opportunities. *IEEE Internet of Things Journal*, v. 3, n. 6, p. 854–864, Dec 2016. ISSN 2372-2541.
- CUDA. *CUDA Zone, September 2019, CUDA Zone [WWW Document]*. URL <<https://developer.nvidia.com/cuda-zone>> (accessed 15.1.2020). 2019.
- CUGOLA, G.; MARGARA, A. Low latency complex event processing on parallel hardware. *Journal of Parallel and Distributed Computing*, Elsevier, v. 72, n. 2, p. 205–218, 2012.
- DELAVAL, G.; RUTTEN, E.; MARCHAND, H. Integrating discrete controller synthesis into a reactive programming language compiler. *Discrete Event Dynamic Systems*, v. 23, n. 4, p. 385–418, Dec 2013. ISSN 1573-7594.
- FARDBASTANI, M. A.; ALLAHDADI, F.; SHARIFI, M. Business process monitoring via decentralized complex event processing. *Enterprise Information Systems*, Taylor Francis, v. 12, n. 10, p. 1257–1284, 2018.
- FERNANDES, M. B.; COSTA, B. A.; LEMOS, J. M. Hydroponic Greenhouse Crop Optimization. In: *2018 13th APCA International Conference on Control and Soft Computing (CONTROLO)*. Ponta Delgada: IEEE, 2018. p. 270–275. ISBN 978-1-5386-5346-3.
- FITCHARD, K. *Brazil, July 2019, Mobile Network Experience [WWW Document]*. URL <<https://www.opensignal.com/reports/2019/07/brazil/mobile-network-experience>> (accessed 10.4.2019). 2019.
- FITZ-RODRÍGUEZ, E.; KUBOTA, C.; GIACOMELLI, G. A.; TIGNOR, M. E.; WILSON, S. B.; MCMAHON, M. Dynamic modeling and simulation of greenhouse environments under several scenarios: A web-based application. *Computers and Electronics in Agriculture*, v. 70, n. 1, p. 105–116, jan. 2010. ISSN 0168-1699.
- GUILLET, S.; BOUCHARD, B.; BOUZOUANE, A. Correct by Construction Security Approach to Design Fault Tolerant Smart Homes for Disabled People. *Procedia Computer Science*, v. 21, p. 257–264, 2013. ISSN 1877-0509.

- HUANG, C.; LU, R.; CHOO, K. R. Vehicular fog computing: Architecture, use case, and security and forensic challenges. *IEEE Communications Magazine*, v. 55, n. 11, p. 105–111, Nov 2017. ISSN 1558-1896.
- IBGE. *Censo Agropecuário 2017 - Resultados Definitivos*. 2018. 1-105 p. Disponível em: <<https://sidra.ibge.gov.br/pesquisa/censo-agropecuário/censo-agropecuário-2017>>.
- INMET. *Rede de Estações Meteorológicas Automáticas do INMET*. [S.l.], 2011. Disponível em: <<http://www.inmet.gov.br/>>.
- JOO, I.; CHOI, D. Distributed optimization framework for energy management of multiple smart homes with distributed energy resources. *IEEE Access*, v. 5, p. 15551–15560, 2017. ISSN 2169-3536.
- JUNIOR, M. R.; OLIVIERI, B.; ENDLER, M. Dg2cep: a near real-time on-line algorithm for detecting spatial clusters large data streams through complex event processing. *Journal of Internet Services and Applications*, Springer, v. 10, n. 1, p. 8, 2019.
- KAMIENSKI, C.; KLEINSCHMIDT, J.; SOININEN, J.; KOLEHMAINEN, K.; ROFFIA, L.; VISOLI, M.; FILEV MAIA, R.; FERNANDES, S. Swamp: Smart water management platform overview and security challenges. In: *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. [S.l.: s.n.], 2018. p. 49–50. ISSN 2325-6664.
- LI, H.; ROSEBROCK, C. D.; RIEFLER, N.; WRIEDT, T.; MÄDLER, L. Experimental investigations on the effects of water vapor and oxygen concentrations in the ambience on the burning constant, lifetime and residuals of single isolated xylene, isobutanol and ethanol droplets. *Experimental Thermal and Fluid Science*, Elsevier, v. 109, p. 1–9, 2019.
- MADUMAL, M. B. A. P.; ATUKORALE, D. A. S.; USOOF, T. M. H. A. Adaptive event tree-based hybrid cep computational model for fog computing architecture. In: *2016 Sixteenth International Conference on Advances in ICT for Emerging Regions (ICTer)*. [S.l.: s.n.], 2016. p. 5–12. ISSN 2472-7598.
- MAZON-OLIVO, B.; HERNANDEZ-ROJAS, D.; MAZA-SALINAS, J.; PAN, A. Rules engine and complex event processor in the context of internet of things for precision agriculture. *Computers and Electronics in Agriculture*, v. 154, p. 347–360, nov. 2018. ISSN 0168-1699.
- MIHAI, V.; DRAGANA, C.; STAMATESCU, G.; POPESCU, D.; ICHIM, L. Wireless sensor network architecture based on fog computing. In: *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*. [S.l.: s.n.], 2018. p. 743–747.
- MOHAMED, N.; AL-JAROODI, J.; JAWHAR, I. Service-oriented big data analytics for improving buildings energy management in smart cities. In: *2018 14th International Wireless Communications Mobile Computing Conference (IWCMC)*. [S.l.: s.n.], 2018. p. 1243–1248. ISSN 2376-6506.
- MORENO, N.; BERTOIA, M. F.; BURGUEÑO, L.; VALLECILLO, A. Managing measurement and occurrence uncertainty in complex event processing systems. *IEEE Access*, v. 7, p. 88026–88048, 2019. ISSN 2169-3536.

- MUANGPRATHUB, J.; BOONNAM, N.; KAJORNKASIRAT, S.; LEKBANGPONG, N.; WANICHSOMBAT, A.; NILLAOR, P. IoT and agriculture data analysis for smart farm. *Computers and Electronics in Agriculture*, v. 156, p. 467–474, jan. 2019. ISSN 0168-1699.
- MUTLAG, A. A.; GHANI, M. K. A.; ARUNKUMAR, N. a.; MOHAMED, M. A.; MOHD, O. Enabling technologies for fog computing in healthcare iot systems. *Future Generation Computer Systems*, Elsevier, v. 90, p. 62–78, 2019.
- OKAY, F. Y.; OZDEMIR, S. A fog computing based smart grid model. In: *2016 International Symposium on Networks, Computers and Communications (ISNCC)*. [S.l.: s.n.], 2016. p. 1–6. ISBN 978-1-5090-0284-9.
- PERALTA, G.; IGLESIAS-URKIA, M.; BARCELO, M.; GOMEZ, R.; MORAN, A.; BILBAO, J. Fog computing based efficient iot scheme for the industry 4.0. In: *2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM)*. [S.l.: s.n.], 2017. p. 1–6. ISBN 978-1-5090-5582-1.
- PONCE, P.; MOLINA, A.; CEPEDA, P.; LUGO, E.; MACCLEERY, B. *Greenhouse design and control*. [S.l.]: CRC Press, 2014. ISBN 978-1-315-77155-7.
- PULIAFITO, C.; MINGOZZI, E.; LONGO, F.; PULIAFITO, A.; RANA, O. Fog computing for the internet of things: A survey. *ACM Transactions on Internet Technology (TOIT)*, ACM, v. 19, n. 2, p. 18, 2019.
- RAHMANI, A. M.; GIA, T. N.; NEGASH, B.; ANZANPOUR, A.; AZIMI, I.; JIANG, M.; LILJEBERG, P. Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach. *Future Generation Computer Systems*, Elsevier, v. 78, p. 641–658, 2018.
- RAMADGE, P. J.; WONHAM, W. M. The control of discrete event systems. *Proceedings of the IEEE*, IEEE, v. 77, n. 1, p. 81–98, 1989.
- RAMPEREZ, V.; SORIANO, J.; LIZCANO, D. A multidomain standards-based fog computing architecture for smart cities. *Wireless Communications and Mobile Computing*, Hindawi, v. 2018, 2018.
- RILEY, G. F.; HENDERSON, T. R. The ns-3 network simulator. In: *Modeling and tools for network simulation*. [S.l.]: Springer, 2010. p. 15–34.
- RULES, P. Department of computer science and engineering the chinese university of hong kong shatin, hong kong. *Future Directions of Fuzzy Theory and Systems*, World Scientific, p. 129, 1995.
- SCHREINEMACHERS, P.; WU, M.-h.; UDDIN, M. N.; AHMAD, S.; HANSON, P. Farmer training in off-season vegetables: Effects on income and pesticide use in Bangladesh. *Food Policy*, v. 61, p. 132–140, maio 2016. ISSN 0306-9192.
- SHIRAZI, S. N.; GOUGLIDIS, A.; FARSHAD, A.; HUTCHISON, D. The extended cloud: Review and analysis of mobile edge computing and fog from a security and resilience perspective. *IEEE Journal on Selected Areas in Communications*, v. 35, n. 11, p. 2586–2595, Nov 2017. ISSN 1558-0008.

- SINCHE, S.; RAPOSO, D.; ARMANDO, N.; RODRIGUES, A.; BOAVIDA, F.; PEREIRA, V.; SILVA, J. S. A survey of iot management protocols and frameworks. *IEEE Communications Surveys Tutorials*, p. 1–1, 2019. ISSN 2373-745X.
- SOURI, A.; HUSSIEN, A.; HOSEYNINEZHAD, M.; NOROUZI, M. A systematic review of iot communication strategies for an efficient smart environment. *Transactions on Emerging Telecommunications Technologies*, Wiley Online Library, 2019.
- STOJMENOVIC, I.; WEN, S. The fog computing paradigm: Scenarios and security issues. In: *2014 Federated Conference on Computer Science and Information Systems*. [S.l.: s.n.], 2014. p. 1–8.
- SUN, Y.; WU, T.-Y.; ZHAO, G.; GUIZANI, M. Efficient Rule Engine for Smart Building Systems. *IEEE Transactions on Computers*, v. 64, n. 6, p. 1658–1669, jun. 2015. ISSN 0018-9340.
- SYLLA, A. N.; LOUVEL, M.; RUTTEN, E. Design framework for reliable and environment aware management of smart environment devices. *Journal of Internet Services and Applications*, v. 8, n. 1, dez. 2017. ISSN 1867-4828, 1869-0238.
- SYLLA, A. N.; LOUVEL, M.; RUTTEN, E.; DELAVAL, G. Modular and Hierarchical Discrete Control for Applications and Middleware Deployment in IoT and Smart Buildings. In: *2018 IEEE Conference on Control Technology and Applications (CCTA)*. Copenhagen: IEEE, 2018. p. 1472–1479. ISBN 978-1-5386-7698-1.
- TAIVALSAARI, A.; MIKKONEN, T. A roadmap to the programmable world: Software challenges in the iot era. *IEEE Software*, v. 34, n. 1, p. 72–80, Jan 2017. ISSN 1937-4194.
- TAKAKURA, T. *Development of VETH chart using computer*. [S.l.], 1976.
- TAKAKURA, T.; FANG, W. *Climate under cover*. [S.l.]: Springer Science & Business Media, 2002.
- TALAVERA, J. M.; TOBÓN, L. E.; GÓMEZ, J. A.; CULMAN, M. A.; ARANDA, J. M.; PARRA, D. T.; QUIROZ, L. A.; HOYOS, A.; GARRETA, L. E. Review of IoT applications in agro-industrial and environmental fields. *Computers and Electronics in Agriculture*, v. 142, p. 283–297, nov. 2017. ISSN 0168-1699.
- WANG, D.; RUNDENSTEINER, E. A.; ELLISON, R. T. Active complex event processing over event streams. *Proceedings of the VLDB Endowment*, v. 4, n. 10, p. 634–645, jul. 2011. ISSN 2150-8097.
- WANG, Y.; GAO, H.; CHEN, G. Predictive complex event processing based on evolving Bayesian networks. *Pattern Recognition Letters*, v. 105, p. 207–216, abr. 2018. ISSN 0167-8655.
- ZHAO, M.; PRIVAT, G.; RUTTEN, E.; ALLA, H. Discrete control for the internet of things and smart environments. In: *Presented as part of the 8th International Workshop on Feedback Computing*. San Jose, CA: [s.n.], 2013.
- ZHOU, K.; FU, C.; YANG, S. Big data driven smart energy management: From big data to big insights. *Renewable and Sustainable Energy Reviews*, Elsevier, v. 56, p. 215–225, 2016.