People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University 20 Août 1955-SKIKDA
Faculty of Sciences
Department of Computer Science

Final dissertation for graduation

Master degree in computer science

Speciality: Systèmes Informatique (SI)

Theme

Formal Verification of Heterogeneous Systems using relative clock regions

Realized by :

- HARRAGA Hanane
- TAABANI Maroua

Supervised by :

LAYADI Saïd

Session: June 2023

Abstract

Currently, formal methods are increasingly being used to analyze the behavior of real-time systems. These systems are often part of critical applications, meaning that an error during their operation can have serious consequences in domains such as avionics, industrial process control and nuclear power plant control. For this reason, these systems need to be verified before their implementation.

Formal verification methods, or simply formal methods, are increasingly being utilized to meet the requirements imposed on this type of systems. In this study, we focused on the verification of heterogeneous real-time systems. This verification require the proposal of formal approaches that facilitate the development of verification tools. Formal methods offer effective techniques for verifying a given system.

The used model, called duration action timed automata with relative time (daTA-R), is an extension of standard timed automata. In this model, distributed real-time systems are composed of a set of daTA components. We assume that each component is characterized by a set of local clocks that evolve at a different frequency, but relative to the clocks of other components. To study the semantics of such systems, we have used the parameter called "slope," which represents the ratio between the clock frequencies.

Also, we have used the equivalence relation between clock valuations to move towards region graphs.

Finally, a tool has been developed to construct, for each daTA-R, the corresponding regions automaton.

Key words: heterogeneous systems , real-time systems, durational actions timed automata, daTA-R, regions graph.

Acknowledgements

The first to thank **Allah** Almighty, who enabled us to reach this high scientific stage, and paved the way for us the way to be among you today to discuss the Master's thesis, and then our parents for all their efforts since our birth until this moment and forever and to our families. You are everything for us we love you in**Allah**

We are pleased to extend our thanks to everyone who advised, guided and contributed to the preparation of this research by providing us the necessary references and sources at all stages.

And we thank in particular Dr **LAYADI Saïd** for the quality of his supervision exceptional, valuable advice, for his time, and patience, and for his rigor and willingness during our preparation this thesis.

We thank the jury members for their presence and interest reading our memory as well as the notes they will give us during this exam to improve our work

Dedication

On such a great day, I am pleased to dedicate this small work to my dear mother that her love continues to guide me and her presence is felt in every beat of my heart .May Allah have mercy on her. To the great woman l dedicate this small success, I will continue to make you honour and proud in all that I do until we meet again in "jannah " .

To whom God said about them, we will support you with your brother, "Sabrina, Nawal",
To my good company, Ranoush, and to every member of my family.

Hannane

Praise be Allah that his grace is just
I dedicate my graduation to my hope in life, the joy of my eye, and the secret of my success, my dear mother, to the one who taught me the meanings of fatherhood, to the one who illuminated the way to my future, my beloved father, Allah perpetuate them and prolong their lives,
To those who helped me in this life and shared groans with me, and who were my role models and guides, my sisters" Souad Salwa" and my absolutely great brothers "Hakim Muhammad Radwan and Youssef, and my twin Zakaria".

Maroua

Contents

List of Figures

Chapter 1

General Introduction

Sommaire

1.1 Work context

1.1.1 Formal verification of real time systems

Recently, a strong enthusiasm has been felt for real-time systems, as they increasingly manage our daily lives. Their most important characteristics are distribution and heterogeneity, which can be expressed in terms of mobility.
Mobile Ad hoc Networks (MANETs) are an example of complex distributed systems, consisting of wireless mobile nodes that can dynamically self-organize into arbitrary network topologies, in order to enable users and devices to communicate with each other under favorable conditions in spaces devoid of any pre-existing communication infrastructure [32].Flexibility and convenience are the reasons why their applications have been extended from the traditional domain (military) to a variety of commercial domains dealing with very specific problems, such as ambient intelligence [3], private networks [81] , and location-based services [18].
For example, with the emergence of autonomous combat drones, the complexity of drone coordination algorithms can make it difficult to provide assurance to a concerned public that these autonomous units are properly coordinating their actions with each other [72] . With a formal model, it would be possible to provide and prove correct algorithms for performing complex tasks.Nowadays, the verification of heterogeneous real-time systems poses a significant challenge. To address this, formal approaches need to be proposed, allowing for the development of verification tools. Formal methods offer efficient techniques for verifying a given system, leveraging mathematical rigor to demonstrate the validity of the underlying system.To perform this verification, it is necessary to introduce formal approaches that facilitate the development of verification tools. Formal methods offer efficient techniques to

verify a specific system, utilizing mathematical rigor to establish the system's validity.
In the last two decades, there has been significant research on various automata-based models, which encompass a wide range of distributed behavior aspects. Additionally, a solid grasp of the fundamental models is essential for developing automated verification techniques since many complex models are constructed by combining these basic ones. In general, time is measured using physical devices known as clocks, which exhibit nearly regular behavior over time. When specifying real-time systems, it is crucial to consider temporal properties, and clocks are explicitly utilized to express system constraints.
In distributed systems, there is no assumption that different timed components share a common time reference or evolve at the same cadence. Models that rely on a global time (i.e., a uniform rate of progression) for all system components fail to capture really aspects of distributed systems.
Our focus lies on modeling and verifying heterogeneous real-time systems, which are pervasive in computer applications and encompass the characteristics of distributed systems, such as reactivity, mobility, and adaptability to uncertain or unforeseen factors. The engineering of heterogeneous real-time systems, including their specification, development, management, and deployment, has become increasingly important. Consequently, verifying the validity of such systems poses a significant challenge.
We consider heterogeneous real-time systems where each participating component possesses its own physical clocks, operating at distinct frequencies of progression. In such scenarios, where a global clock is neither available nor desirable, it becomes imperative to model systems while taking into account the individual clock frequencies of the participating components. Thus, studying these systems based on their diverse temporal evolutions becomes a natural approach.The majority of systems explicitly incorporate constraints related to real-time aspects. These constraints are quantitative in nature and interpret delays or durations, such as response times or timeouts. Most classical models and automated verification techniques have been extended to handle such real-time systems.
In 1990, Alur and Dill introduced the model of timed automata to describe the behavior of real-time systems with quantitative constraints [6, 11].Verification techniques have been expanded to be compatible with this model [6, 46, 57]. Specification formalisms, such as temporal logics, have also been extended to interpret real-time properties characterizing systems [6, 46, 57]
There have been efforts to develop model-checking tools [79, 58], which have been successfully applied to real-world cases in the industrial domain [76, 22].
In[5, 4], the authors proposed a model for distributed timed systems where each component is a timed automaton with a set of local clocks that evolve at independent frequencies from the clocks of other components. A clock can be read by any component in the system but can only be reset at the level of the automaton it belongs to. For such systems, two main semantics have been discussed: universal semantics and existential semantics. The universal semantics captures behaviors that hold regardless of the individual components' clock frequencies. This is a natural choice when verifying that a system consistently satisfies positive specifications. However, when verifying if a system avoids negative specifications, it is preferable to use the existential semantics, which considers the set of behaviors that the system can potentially exhibit under different clock frequency choices.
The existential semantics always describes a regular set of behaviors. However, in the case of

universal semantics, the problem of determining if the set of behaviors is empty is undecidable. While the existential semantics allows us to verify negative specifications, the universal semantics is suitable when assessing whether a system exhibits a desirable behavior that remains robust despite clock variations. Unfortunately, determining the existence of such behaviors is an undecidable problem.

1.1.2 Work Motivations

In this work, we address the issue of heterogeneous system components. Heterogeneity is considered as a difference in the rates of component evolution. When a system consists of multiple elements, each element has its own clock frequency. Therefore, we propose an approach in this study for modeling and verifying distributed systems that incorporate local and relative clock frequencies. We describe the system using a timing mechanism where all clocks progress at the same frequency. However, clocks belonging to different processes (components) are allowed to advance at different but proportional rates. We assume that all processes can read the clock (use it), but only the process it belongs to can reset it. Defining the symbols associated with the above proposed model becomes challenging in the form of a timed transition system, as the number of configurations becomes infinite.
To solve this problem, we will review the proposed equivalence relations on clock valuations that aim to aggregate configurations. This implies that an equivalence class (referred to as a clock region) can represent a set of configurations [11].
Since the future behavior of any modeled system is determined by its current locality and the values of clocks in all processes, we will redefine several concepts, such as clock regions and the region automaton, with a specific focus on the impact of the relation between clock frequencies.
In this context,we will demonstrate that the problem of verifying temporal satisfaction among coordinated components in a formal verification of a heterogeneous systems can be reduced to a search on a graph of regions.
Therefore, we will first extend duration actions with timed automata (daTA-R) by introducing a characterization of relative time speeds associated with the components to address heterogeneity.

1.2 Contributions

In this work, we focused on the formal modeling and verification of real time and heterogeneous systems. The model of duration action timed automata has been extended with the notion of relative time in order to model such systems.Our contributions can be summarized in the formal modeling and verification of heterogeneous systems, aiming to verify their behavioral and temporal properties. The decidability of the accessibility property has been proven through the use of an abstraction of the system's behavior called "region". This abstraction requires redefining the necessary concepts to accommodate it.
We are interested in the formal verification of heterogeneous systems where each participating component has its own physical clocks with its own frequency of progression, even though there is no available or desirable global clock.

The model of Duration Action Timed Automata (daTA) is a semantic structure designed as a system of transitions. It is useful for expressing causality and temporal constraints that must be adhered to during the execution of actions.
We have proposed an extension of the Duration Action Timed Automata model with relative time (referred to as daTA-R) to address relative time speeds.
A daTA-R structure represents the potentially infinite behaviors of timed systems. This model focuses on the temporal constraints imposed by heterogeneous systems, taking into account the relative time speeds that differentiate the coordinated components.
In a daTA-R, the clocks belonging to different components evolve synchronously but under the relative rates assigned to these components. Each rate assigned to a component p represents the speed of p and depends on a certain reference of the global time, from which the local time of p will be derived.
To study the semantics of such systems, we have defined a parameter called "slope", which represents the ratio between the clock frequencies. This parameter enables us to assess and prove the decidability of accessibility in a daTA-R.
At any given point in time, the future behavior of the modeled system is determined by its current locality and the values of clocks in all processes. To address this, we have redefined the algorithm for constructing a region automaton, taking into account the effect of relativity between clock frequencies.
The semantics of a daTA-R is defined by two new relations: the equivalence relation on the set of all clock valuations and the successor relation on these equivalence classes. These two relations have allowed us to propose a method for constructing a finite region automaton to study and prove decidability. This region automaton captures the infinite set of behaviors implied by the daTA-R specification.
We have presented that the construction of the region automaton terminates with a finite space of regions.

1.3 Dissertation outline

Chapter 1 includes the general introduction.

Chapter2 Presents the state of the art on the real-time systems, the critical systems and heterogeneous systems with examples for each category of systems,

Chapter3 gives this chapter the famous model of durational actions Timed Automata (daTA). We present the basic notions of this model, namely the syntax, the semantics, the composition of several timed automata...; and we will insist on the use of this model in the formal verification with its different types: accessibility, inclusion of languages, properties logics, tests, behavioral equivalences... Then, we approach the extensions and the sub classes (the most important) of timed automata with emphasis on three points for each variant: the accessibility of a given state, the expressiveness and conciseness of the model. And we ends this chapter by giving some concepts on the practice of verification, in particular the two systems analysis approaches forward and backward, DBM structures, graphs of areas and a brief presentation on the Uppaal tool.

Chapter 4 Chapter 4 details the variant of durational actions timed automata model, on which we base our work in this thesis to address the issues and overcome the limitations mentioned above in the motivation of this dissertation. We started with the definition of durational actions timed automata with relative time model (daTA-R),we have explained the principle of maximality semantics by emphasizing its advantages over the semantics of interlacing. Then, we gave the basic concepts of this model, namely the syntax, semantics and product of daTA-R.

Chapter 5 presented the part of the realization of our application and the language of programming and the techniques and tools used.

And finally, a general conclusion.

Chapter 2

Critical and heterogeneous real-time systems

Sommaire

2.1 Introduction

In this chapter, will introduce real-time,critical and heterogeneous systems in general,with examples from each system.

2.2 Real-time systems

Computer systems consist of a combination of hardware, software, user and data. Such systems are constantly evolving and will become undoubtedly a constituent part of our daily lives. Real-time systems are defined as those systems in which the correctness of the system depends not only on the logical results of computations, but also on the physical time at which the results are produced [73]. In other terms, real-time systems are computing systems that have temporal constraints to meet and thus, are required to guarantee a response within specified timing constraints.Such systems need to achieve different requirements that can be structured into three main categories[48]: (i) functional requirements, (ii) temporal requirements, and (iii) dependability requirements. Functional requirements refer to the functions that the real-time system must perform.
In real-time systems,this attribute is specified using the concept of a deadline.Based on the consequence of missing a deadline , real-time systems are typically classified into three different classes:(i)Hard,(ii)firm,(iii)soft real-time systems[1].In hard deadlines, that is, deadlines

that if missed could result in severe consequences. On the other hand, for soft real-time systems the goal becomes to meet a certain subset of deadlines in order to optimize some specific criteria. Particularly, deadline violations result in degraded quality. However, the system keeps operating and may recover in the future[48].In a frim real-time system,deadline can be missed occasionally.

2.2.1 Example for real-time systems

- **Embedded system:**
  An embedded system can be defined as a computer system specifically designed to perform dedicated functions within a larger mechanical or electrical system. It is characterized by its integration into a larger system or product, where it operates as a component with a specific purpose. Embedded systems are typically embedded within physical devices or machines, often with limited resources and specific requirements. "An embedded system is a combination of computer hardware and software, designed to perform specific functions within a larger system, often with real-time constraints. It is typically embedded into a larger mechanical or electrical system, operates with limited resources, and is tailored to meet specific application requirements."[65]

- **Real-Time Video Surveillance Systems:**
  Real-time video surveillance systems are used for security and monitoring purposes, such as in airports, public spaces, and critical infrastructure. These systems process live video feeds in real-time, detect anomalies or security threats, and trigger immediate alerts or responses.[74]

2.3 Critical systems

Critical systems are systems where a failure or malfunction will result in significant negative consequences such as physical injury, property or environmental damage. These systems have stringent safety and security requirements to protect the user or other people. Alternatively, these systems may be mission critical, underpinning products, for profit or competitive advantage[41].A critical systems are typically classified into three different classes:(i)Safety-critical systems,(ii)Mission-critical systems,(iii)Business-critical systems. A safety-critical system is a system where safety is of importance.Basically, a safe system is a system that delivers a service free from occurrences of catastrophic consequences on the environment and user [17].On the other hand,for mission-critical systems are systems that are essential to the survival, operation, or mission of an organization or system, particularly those systems that are relied upon for safety, security, and economic stability. These systems include, but are not limited to, transportation, communication, energy, and healthcare systems[38].In critical business systems are information systems and software that support a company's operations and are considered vital to its continuity. These systems include financial transactions, business operations, and other core processes that maintain the company's operations. Many economic sectors rely on these systems to maintain productivity and effective performance[71].

2.3.1 Example of critical systems

- Communication systems such as telephone switching systems, aircraft radio systems, etc
- Embedded control systems for process plants, medical devices, etc.
- Embedded control systems for process plants, medical devices, etc.

2.4 Heterogeneous systems

Heterogeneous systems are inherently complex systems using a systems of systems approach [77]. These heterogeneous systems are comprised of subsystems with multiple interactions between these subsystems. Heterogeneous systems are different from classical software systems because subsystems in heterogeneous systems are functionaly independant and often exhibit hetreogeneity in terms of hardware, software and processes.The HSA foundation has defined a model for queues that enables a more precise control over scheduling the GPU workloads. These advances have led to numerous issues that can not be detected or studied with current tracers and profilers. Most tracers and profilers focus on the CPU, the operating system, or on one coprocessor and do not provide a global view of the system. This is an issue when all these components should work tightly together, and performance defects arise in some scenarios because of interaction problems. A performance diagnostic toolkit that supports heterogeneous hardware and can integrate and correlate tracing and profiling information from all levels is needed in this context.GPU simulators are sometimes used to analyze new hardware and software approaches to memory and scheduling issues [68, 63].The HSA is a low-level heterogeneous program- ming framework that is supported by multiple high-level GPU programming languages (e.g.,OpenCL and C++ AMP), hardware devices and device drivers [62].

2.4.1 Example of heterogeneous systems

1.Cloud Computing
The Cloud computing [66] is a paradigm that enables the acquisition of computational resources on demand in a pay-per-use model. Consumers of cloud computing resources can provision their own resources when they are needed, and can release them quickly once they are not required. Resources are kept in large pools, giving the consumers the notion of an infinite amount of available resources. Resource utilization is metered by the cloud service provider. The main principle behind this model is offering computing, storage, and software "as a service". There are many definitions of Cloud Computing, but they all seem to focus on just certain aspects of the technology [31].Cloud computing leverages the virtualisation of computing resources to allow end-users to provide them at an acceptable price. In increasing the computation demands, GPUs have been introduced in Cloud data centres because of their performance abilities and their suitability for some applications [33]. Moreover, GPU clusters will play an important role in the future of Cloud computing data centres since some compute-intensive applications need to involve GPUs with CPUs [78]. Cloud computing providers like Amazon , Microsoft Azure , IBM Bluemix , NIMBIX and recently Google have

enabled the users to access GPUs located in their Cloud data centres. Therefore, this recent situation is changing the taxonomy of the Cloud data centres and the methods of managing these resources.

2.Ad hoc networks

Ad hoc networks are new paradigm of networks offering unrestricted mobility without any underlying infrastructure. An ad hoc network is a collection of autonomous nodes or terminals that communicate with each other by forming a multihop radio network and maintaining connectivity in a decentralized manner. Each node functions as both a host and a router. More critically, the network topology is in general dynamic, because the connectivity among the nodes may vary with time due to node departures, new node arrivals, and the possibility of having mobile nodes. There are two major types of wireless ad hoc networks: Mobile Ad Hoc Networks (MANETs) and Smart Sensor Networks (SSNs) [36].

Chapter 3

durational actions Timed Automata (daTa)

Sommaire

3.1 Introduction

Timed automata is a theory for modeling and verification of real time systems. Examples of other formalisms with the same purpose, are timed Petri Nets, timed process algebras, and real time logicS [25]. Following the work of Alur and Dill [8, 12], several model checkers have been developed with timed automata being the core of their input languages e.g. [80]. It is fair to say that they have been the driving force for the application and development of the theory.The goal of this chapter is to provide a tutorial on timed automata with a focus on the semantics.

In the original theory of timed automata [8, 12], a timed automaton is a finite-state Büchi automaton extended with a set of real-valued variables modeling clocks. Constraints on the clock variables are used to restrict the behavior of an automaton, and Büchi accepting conditions are used to enforce progress properties. A simplified version, namely Timed Safety Automata is introduced in [47] to specify progress properties using local invariant conditions. Due to its simplicity, Timed Safety Automata has been adopted in several verification tools for timed automata e.g. UPPAAL [55] and Kronos [80]. In this context the model of Durational Actions Timed Automata (daTA*) has more interest. daTA* is a timed model, its semantic express the durations of actions and other notions for specifying the real-time systems such as urgency and deadlines [21]This model is based on a maximality semantic [7] and advocates the true concurrency; from this point of view it is well suitable for modeling real time, concurrent and distributed systems. But, the consideration of a dense time domain make the generation of space state infinite from which the impossibility to use modelchecking as methods for verification.

The rest of the chapter is organized as follows: In the next section, we describe the syntax and operational semantics of timed automata.

Example.1 The Figure 1 shows a daTA. This clock will be used in the construction of the timing constraints as guards of the transitions. This model is illustrated by the example of Figure 1 composed of two states S_0 and S_1 a transition labelled with an action a of duration 2 units of time. From the initial state of the illustrative daTA, the execution of the action a leads to a reset of the clockx_0 associated with it.The expression $x \leqslant 2$ in state S_1 represents a duration condition on the action a and means that a is potentially in execution until the clock a reaches the value . The action does not wait for the end of any other action, so the clock designated by the enabling domain of this action will be $C_\emptyset$. This enabling domain will be expressed by the guard and the deadline on the clock $C_\emptyset$ ($1 \leqslant C\_\emptyset \leqslant 3$) .

3.2 Syntax, semantics of timed automata (daTA)

The Durational Actions Timed Automata model daTA* [20] is a timed model defined as Timed Automata over an alphabet representing actions to be executed.In the 1990s, timed automata were introduced by R. Alur and D. Dill [9]. They extended classical automata by clocks which increment continuously and synchronously with time. Two news notions have been added to transitions:

- A guard defined on the value of the clocks describing the moment when the transition can be fired and a reset to zero for a set of clocks when drawing the transition
- Another new notion called invariant has been added to localities and expressed as a constraint on the clocks, this invariant can force the system to leave a locality to start the execution of a action by limiting the waiting time in this locality.

The time domain can be either the set of natural $\mathbb{N}$, or the set of positive rationals $\mathbb{Q}_{\geq 0}$ or the set of positive $\mathbb{R}_{\geq 0}$. In this section, we consider positive reals, however, note that there is not a big difference in the results if we consider$\mathbb{Q}_{\geq 0}$ or$\mathbb{N}$. Indeed, by considering $\mathbb{N}$, the granularity of the progression periods time can be adopted for the analysis of the properties of the system.

3.2.1 Preliminaries

We present in this section Timed Automata.In the following $\mathbb{R}_{\geq 0}$ is the set of nonnegative real numbers.A clock takes values from $\mathbb{R}_{\geq 0}$ or it is undefined, denoted by$\perp$. Without loss of generality, we write $R^+ \cup (\perp)$ where the set of nonnegative real numbers is extended with the special value$\perp$ Given a set X of clocks, a clock valuation over X is a function assigning a nonnegative real number to every clock.
The set of valuations of X, denoted $\{V_x\}$, is the set of total function from X to R^+ The valuation $v + d$ maps every clock y to $v(y) + d(d \in R^+)$.
Given a set λ of clocks, a reset is a subset of X. Given a valuation v and a reset λ, we define the valuation $v[\lambda \leftarrow 0]$ by $v(x) = 0$ for all x in λ and $v(x)$ if $x \notin \lambda$.
The set C_X of clock constraints over X ,where x , $t \in {R_\perp}^+$, and $*$ is a binary operator and $* \in \{=, <, \leq, >, \geq\}$. Clock constraints are evaluated over clock valuations. The satisfaction with respect to clock valuation function $v \in V_X$, he expression $\perp * \perp$ evaluates to true and all other comparisons that involve evaluate to false. We write $v \in C_X$ to denote that according the valuation function $v| = C_X$ evaluates to true. Let ACT be a set of label symbols.In general way timed automata are finite state machines whose transitions are decorated by clocks constraints.

3.2.2 Formal Syntax and Semantics

In this section, we present the syntax and the semantics of the model that we consider in this thesis, as well as the model of timed automata with durations of actions,which we abbreviate as daTA.

Definition 3.1

a daTA*A is a tuple (S, L_s, s_0, H, T_D) of the support ACT whereover:

- S is a finite set of locations,
- $l_s : S \rightarrow 2_f n^{\phi_t(H)}$ is a function which assigns to each state S the set F of ending conditions (duration conditions) of actions possibly in execution in S ,
- $S_0 \in S$ is the initial state, such that $L_s(s_0) = \emptyset$,

- H is a finite set of variables named clocks ,
- $T_D \subseteq S \times 2_{fn}^{\phi_t(H)} \times 2_{fn}^{\phi_t(H)} \times ACT \times H \times S$ is the set of transitions.
- A transition (s, G, D, a, x, s') represents a switch from state s to state s' by starting execution of action a and resetting clock x. G is the corresponding guard which must be satisfied to fire this transition. D is the corresponding deadline which requires, at the moment of its satisfaction, that action a must occur. (s, G, D, a, x, s') can be written $s \xrightarrow{G,D,a,x} s'$.

Figure 3.1 represents an example of daTA

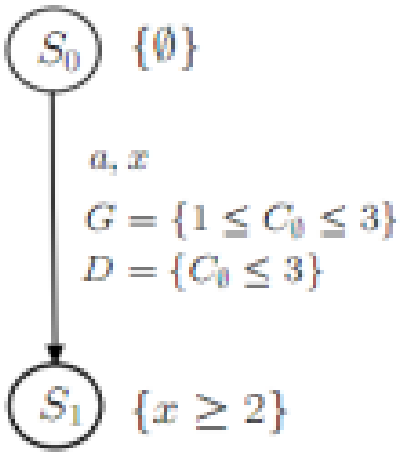

Figure 3.1: Example of daTA.

Definition 3.2

A timed transition system (TTS) over the alphabet A is a tuple $S = (Q, q_0, A, \rightarrow)$ where Q is a set of states, $q_0 \in Q$ is the initial state, A is a finite set of actions disjoint from $\mathbb{R}_{\geq 0}$, and $\rightarrow \subseteq Q \times (A \cup \mathbb{R}_{\geq 0}) \times Q$ is a set of edges.If $(q, e, q_0) \in \rightarrow$, we also write $q \xrightarrow{e} q_0$. Moreover, TTS should satisfy the classical time-related conditions where $d, d_0 \in \mathbb{R}_{\geq 0}$:

- Time determinism: $(q \xrightarrow{d} q') \wedge (q \xrightarrow{d} q") \Rightarrow (q' = q")$
- Time additivity: $(q \xrightarrow{d} q') \wedge (q' \xrightarrow{d'} q") \Rightarrow (q \xrightarrow{d+d'} q")$
- Null delay: $\forall q : q' \xrightarrow{0} q$
- Time continuity: $(q \xrightarrow{d} q_0) \Rightarrow (\forall d' \leq d, \exists q", q_{d'} \xrightarrow{d-d'} q")$

Let $S = (Q, q_0, A, \rightarrow)$ be a TTS. Let $\rightarrow^*$ be the reflexive and transitive closure of $\rightarrow$.We denote $\text{Reach}(q_0) = \{q \in Q \mid q_0 \rightarrow^* q\}$, the set of reachable states S .

Definition 3.3

The semantics of a daTA $A = (S, L_s, s_0, H, T_D)$ is defined by associating to it an infinite transitions system S_A over $ACT \cup R^+$. A state of S_A (or configuration) is a pair $<s, v>$ such that is a state of A and v is a valuation for H . A configuration $<v_0, s_0>$ is initial if s_0 is the initial state of$A$ and $\forall x \in H, v_0$(x)= 0 Two types of transitions between S_A configurations are possible, and which correspond respectively to time passing (rules $RA1$ and $RA2$) and the launching of a transition from A (rule RD).

$$\text{(RA1)}\ \frac{d \in \mathbb{R}^+\ \forall d \leq d,\ v + d' \notin \mathfrak{D}}{\langle s, v\rangle \xrightarrow{d} \langle s, v + d\rangle}$$

$$\text{(RA2)}\ \frac{\varepsilon \in \mathbb{R}^+\ \ v + \varepsilon \vDash \mathfrak{D} \wedge \varepsilon \in \eta}{\langle s, v\rangle \xrightarrow{\varepsilon} \langle s, v + \varepsilon\rangle}$$

$$\text{(RD)}\ \frac{(s, G, D, a, x, s') \in T_D\ \ v \vDash G}{\langle s, v\rangle \xrightarrow{a} \langle s', [\{x\} \to 0]v\rangle}$$

Where α is the smallest real quantity of time in which no action occurs (Belala, 2010).In RA rules,$D = V_{i \in I}\ D_i$ where $(S, G_I, D_I, a_i, x_i, s_i)_{i \in I}$ is the set of all transitions stemming from state S. Indeed, whenever a D_i holds, time cannot progress regardless of the otherD_i.
Note that if one wants to guarantee that at least a transition could be drawn starting from a state if time cannot progress any more within this state,one requires that the formula $D_i \to G$.

3.2.3 Parallel composition of timed automata

can be defined in a classical way, for example by constructing an n-ary synchronization with renaming. This type of operation allows the parallel execution of timed automata while synchronizing their actions. It is commonly used to model complex systems by decomposing them into smaller components that can be analyzed and tested independently before being combined into the the final system.

3.3 Decidability of the accessibility property

In this part, we want to describe a construction proposed in Alur and Dill (1994) [11]to decide whether a locality in a timed automaton is accessible or not. This construction is an abstraction of the behavior of the considered timed automaton in such a way that testing the accessibility of a locality in a timed automaton is reduced to testing whether a state (or a set of states) in another finite to decide if a locality in a timed automaton is accessible or not. This construction is then an abstraction of the behavior of the timed automaton considered in such a way so that the test of the accessibility of a locality in a timed automaton is reduced to testing if a state (or a set of states) in another finite automaton is accessible.

The key feature of this construction is the generation of a finite-state automaton in which a state can be an aggregation of an infinite number of states of the timed transition system. Therefore, any transition by action or time passage in the timed transition system finds its equivalence in the integrated automaton, and the reverse is always true. It should be noted that the precise waiting times in the individual locations may not be the same, so the equivalence relation with respect to time is a bisimulation relation that abstracts away from time and attempts to obtain a finite number of equivalence classes.

for $x \in H$, for any $v(x) \in R^+$,$fract(v(x))$ denotes the fractional part of v", and $[v(x)]$ denotes the integral part of v". For $x \in H$, let c_x be the largest integer c such that $(x \leq 0)$

or $c \leq x$ is a sub-formula of some clock constraint appearing in T .

- For all $x \in H$ either $[v(x)]$ and $[v'(x)]$ are the same, or both $v(x)$ and $v'(x)$ are greater than c_x.
- For all $xy \in H$ with $v(x) \leq c_x$ and $v(y) \leq c_y$, $fract(v(x)) \leq fract(v(y))$ iff $fract(v'(x)) \leq fract(v'(y))$.
- for all $x \in H$ with $v(x) \leq c_x$, $fract(v(x)) = 0$ iff $fract(v(x)) = 0$.

The equivalence relation $\approx$ is defined over the set of all clock interpretations for H. We will use V to denote the equivalence classes of $V(H)$ to which " belongs. It also clock regions denoted by:$[v] = v' \in V(H)|v \approx v'$

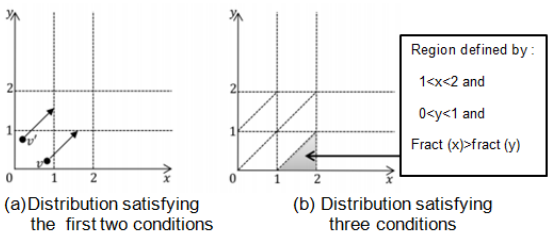

Figure 3.2: Region index for two clocks that have the same maximum value

Intuitively, the first two conditions express that if a clock valuation satisfies certain constraints of the timed automaton, then the other equivalent valuation also satisfies the same constraints. The third condition ensures the same order of arrival for new integer values when different clocks increment from two equivalent configurations. The equivalence relation $\sim$ defined above concerns clock valuations, and an equivalence class is therefore called a clock region.
The construction of clock regions is illustrated by Figure 3.2 where there are two clocks x and y and a maximum constant $C_x = C_y = 2$. If we apply the first two conditions of the equivalence relation $\sim$ on this example, we get the partition presented in figure 3.2 (a) that provides valid regions for all temporal constraints defined with constants smaller or equal to 2. However, the problem that arises is that we can find two valuations in the same region that are not equivalent with respect to the passage of time (see the third condition of the equivalence relation $\sim$ above). For example, let's take the two valuations v and v' (see Figure 3.2 (a)). If we let time elapse from v, we will first reach the integer value $x = 1$ before clock y reaches integer value 1, while from v', we will reach integer value $y = 1$ before clock x reaches 1. The possible behaviors from v and v' are therefore not the same. The third condition of $\sim$ refines the partition presented in Figure 3.2(a) by diagonal lines representing the passage of time and provides the partition shown in Figure 3.2 (b), which turns out to be a bisimulation relation with time abstraction. This equivalence relation enables the construction of a finite automaton from the initial timed automaton A as follows: the automaton states (configurations) are pairs (s, α) where s is a location of the timed automaton, and α is a clock region constructed according to the conditions of the equivalence relation α. The transitions are $(s, \alpha) \xrightarrow{a} (s', \alpha')$ if there exists:

- A transition $s \xrightarrow{G,a,R} s'$ in the timed automaton A,
- a valuation of clocks $v \in \alpha$ and $t0$ such that:
  - $v + t \models I(s)$,
  - $v + t \models G$;
  - $(v + t)[R] \models I(s')$ and
  - $(v + t)[R] \in \alpha'$.

The finite automaton $R(A)$ resulting from this construction associated with the initial timed automaton A is called the region automaton. The essential property of this region automaton is that it recognizes exactly the same language as that of the initial timed automaton, i.e., if $R(A)$ recognizes a word $a_1 a_2 \ldots$, then there exists a timed word $(a_1, t_1)(a_2, t_2) \ldots$ recognized by the initial timed automaton A. Thus, the emptiness test of the language recognized by a timed automaton A or the accessibility problems of a location in A can be reduced to an accessibility problem in its region automaton $R(A)$. This gives an algorithm for these two problems that is PSPACE-complete:

3.3.1 Theorem 3.1

The accessibility of a location in timed automata [9, 11]is a PSPACE-complete problem. To clarify these notions, we will construct the region automaton from the timed automaton

in Figure3.3(A). Figure 3.3(B) shows the resulting region automaton. In this example, to determine whether the location $s2$ in A is accessible, one needs to prove the accessibility of one of the states $(s3, \alpha)$ in the region automaton in Figure 3.3(B), where α is a clock region. In region automaton,there is a path leading to a state containing the locality $s3$,who is $(s_0, x = y = 0) \xrightarrow{a} (s_1, 0 = y < x < 1) \xrightarrow{c} (s_3, 0 < y < x < 1)$. This path tells us that there exists an execution in the timed automaton A of the form $(s_0, v_0) \xrightarrow{t_1} (s_0, v_0 + t_1) \xrightarrow{a} (s_1, v_1) \xrightarrow{t_2} (s_1, v_1 + t_2) \xrightarrow{c} (s_3, v_2)$,the transitions leading to the location s_3, or t_1 and t_2 are positive real numbers.

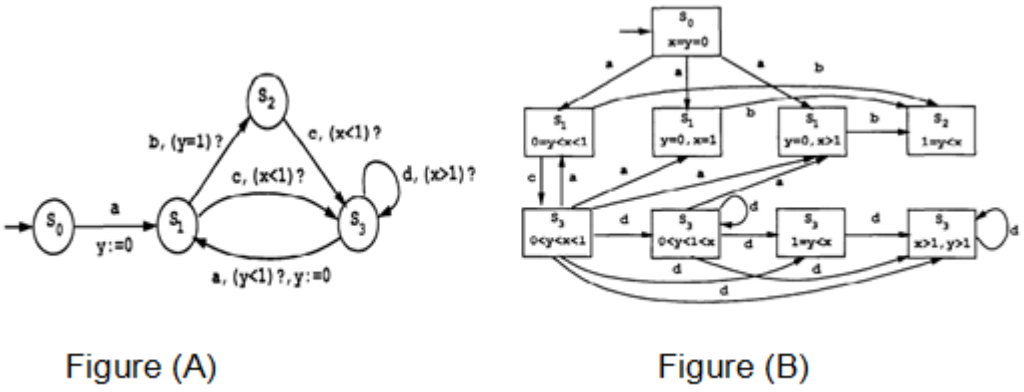

Figure 3.3: Automaton A and its region automaton

The computation of the complexity of the accessibility problem (evoked in the previous theorem) was done on the original paper [11].

- The PSPACE-hardness stems from the possibility of encoding, respecting the word w, the behavior of a Turing machine M that is linearly bounded in space. This implies the existence of a timed automaton such that the clock valuation of this automaton coincides with the state of the tape of M during execution. It should be noted that this encoding can be done using only three clocks.[37]
- The PSPACE-easy side comes from the possibility of considering a non-deterministic algorithm that can memorize the current symbolic state (a location and a clock region) and a successor state generated in a non-deterministic manner.

3.4 Verification by inclusion of timed languages

In the previous section, we showed that it is always possible to associate with an execution in a timed automaton a timed word that is described as a sequence of pairs (a, t) where a represents the label or action of the transition and t is the time instant at which action a was taken. An acceptance condition can be added to the model of timed automata (final locations, Büchi, Müller, etc.) to turn this model into a machine that accepts a language of timed words. If the correct behaviors of a system specification coincide with a language accepted by a timed automaton, then the verification problem reduces to studying the following question about timed languages: is the inclusion of the language of the studied system in that of

its specification verified? In untimed systems, this problem is solved by verifying that the intersection between the language of the studied system and the complement of the language provided by the specification is empty. In the case of timed systems, it is generally impossible to construct a timed automaton that recognizes the complement of a language accepted by a timed automaton, which makes it impossible to apply this method. Moreover, it has been proven in [11] that the inclusion problem mentioned above is undecidable, and verifying such timed properties can only be done for specific timed languages, such as those accepted by deterministic timed automata

3.4.1 Logic of timing

To extend time logic by quantitative constraints, we can

- or conditioning the classical modalities of these logics by constraints,
- or integrate watches (which relate to the formula of ownership) and operators to manage them.

The first option is to incorporate a constraint that has the form $\bowtie c$, such as $\bowtie \in \{=, <, \leq, \geq, >\}$ and $c \in \mathbb{N}$,at the level of the Until time operator noted by. Then we can say that a formula $\varphi \cup_{<c} \psi$ is true for an execution ρ if and only if there is a state such as :

- there are amounts of c units of time from the initial state to this state,
- He checked ψ and
- all previous visited states pending execution ρ verify φ .

More generally, it is possible to integrate an interval $[a; b]$ at the level of an operator $Until$. This way of expanding time logic is quite classic [49]. This extension was also proposed using CTL for the discreet time [40] and for the dense time [6] Logic Timed CTL (TCTL) was formally defined from the atomic propositions (such as labels on the system states to be checked), the Boolean operators $(\wedge, \vee, \neg)$ and the modalities $E\_\cup_{\bowtie c\_}$ and $A\_\cup_{\bowtie c\_}$.Property (1) is written as follows:

$$AG(problem \Rightarrow AF_{\leq 5}\ alarm)$$

Another approach to extending time logic by quantitative aspects [16] is to integrate.

- of clocks of formulas (all of these watches are rated H) that increase synchronously with time,
- a zero discount operator (in) and
- of simple constraints $x \bowtie c$ with $x \in H$.

The reduction to zero followed by a $x \bowtie c$ constraint captures the time that separates the two systems states. Formally, $TCTL_h$ logic was defined from CTL by adding $x \bowtie c$ constraints with $x \in H$ and $x \in H$ the operator $x\ \underline{in}\ \_$. Property (1) is written as follows:

$$AG(problem \Rightarrow (x\ \underline{in}\ (AF(x \leq 5 \wedge alarm))))$$

In this formula, the operator $\underline{in}$ reverses the clock x to zero when one encounters a state that checks "*problem*" and thus one must check that the watch x is less than or equal to 5 when one meets a state which checks the state "*alarm*" to ensure that the time separating these two states is less or equals 5.
It is clear that all operators of $TCTL$ can be expressed by using the clocks of formulas. We have the following equivalence for the formulas of $TCTL$ φ andψ :

$$E(\varphi \cup_{\bowtie c} \psi) \equiv x \ \underline{in} \ E(\varphi \cup (\psi \wedge x \bowtie c))$$

Very fine properties can be expressed by the $TCTL_h$ logic, it has been shown in the case of dense time that this logic is more expressive than $TCTL$ [29] : The proof is that this formula has no equivalent in $TCTL$:

$$x \ \underline{in} \ EF(P_1 \wedge x < 1 \wedge EG(x < 1 \Rightarrow \neg P_2))$$

The formula above expresses the fact that it is feasible to a state that verifies the P_1 property in less than 1 unit of time and from that state there is an execution during which there is no state verifying the P_2 property before the x clock is equal to 1.
We have the following result:

Theorem 3.2

The $TCTL$ or $TCTL_h$ logic[6] checking model for timed automates is a PSPACE-complete problem. Verification algorithms do the labeling of an extended automate of regions with special clocks that are used to check time constraints in logical formulas. Kronos is a tool used to check formulas written in $TCTL$ on parallel compositions of timed automates[79].

3.4.2 Logic of linear time

Linear time logic can also be extended so that properties can be formalized on the executions of timed automates.Then we write Property (1) as follows: $G(problem \Rightarrow F_{\leq 3}\ alarm)$.Among the linear time timed logic can be mentioned MTL [49, 14] where the $\cup$ modality has been extended by intervals, as well as a MTL restriction that does not allow the reduction of intervals to a single value (i.e. you cannot use the $\cup_{=c}$ modality).
The MTL checking model is an indecisible problem when considering a semantics for which the observation of the system is continuous: each atomic proposition has a truth value defined over time intervals during executions. On the other hand, the checking model is decisive when one considers a punctual semantics for which these atomic propositions are true at moments. For $MITL$, the model checking is an EXPSPACE-complete problem and becomes PSPACE -complete if the "*Until*" terms are used with the constraints $< c$ or $> c$ [13]. For more details on several quantitative time logic, we can refer to [15, 6, 14, 46, 69].

3.4.3 Modal Logic with Fixed Points

Modal logic can also be extended to enunciate real-time properties.Inspired by Hennessy and Milner's logic [43] one can describe the behavior of the system by defining the modalities that relate to the transition labels of the STT.There are two ways to express quantification:

- existential, written in the form $< a > \varphi$ to formalize "it is possible to draw a transition a and then check φ" and;
- universal, $[a]\varphi$ to formalize "after any transition labeled with a, φ is true."

For time transitions, we can use the symbol δ in the form of $< \delta > \varphi$ to express that it is possible to wait a certain time, without any action transition being drawn, until φ s verified. Fixed-point operators can also be used to formalize properties that involve unlimited behaviors[54, 75]. To measure the time that separates the actions from the system studied, formula clocks (such as those of$TCTL_h$ were used to deal with these quantitative aspects.
The checking model for timed modal logic with fixed points is an EXPTIME-complete problem [2].

3.4.4 Automated Testing

A test machine [1] is a T_φ timed machine built to describe an error scenario and that its verification can be carried out by applying an accessibility problem to the parallel composition of the automate that is checked with T_varphi.It is also possible to use timed modular logic to describe the correction property and automatically build the associated test automatic. Note that this approach is only applicable to a limited category of properties.

3.4.5 Behavioral Equivalence and bisimulation

Behavioral equivalences are useful for comparing timed systems. A widely used class of these equivalences is temporized strong bisimulation. It is said that the two statements (s_1, v_1) and (s_2, v_2) are heavily bisimilar and it is written $(s_1, v_1) \approx (s_2, v_2)$ if and only if:

- for any transition of $(s_1, v_1) \xrightarrow{a} (s'_1, v'_1)$ with $a \in Act$, there is a transition $(s_2, v_2) \xrightarrow{a} (s'_2, v'_2)$ such as $(S_1, v'_1). \approx (s_2, v_2)$, and vice versa, and
- for any transition of $(s_1, v_1) \xrightarrow{t} (s'_1, v'_1)$ with $t \in R_{\geq 0}$, there is a transition $(s_2, v_2) \xrightarrow{t} (s'_2, v'_2)$ such as $(s_1, v'_1). \approx (s'_2, v'_2)$, and vice versa.

The above definition of behavioral equivalence is very strong, and in particular, for a bisimulation relationship that respects the initial and final states, it is said that two timed automates are strongly bisimilar if they accept the same timed words.
It should be noted that the time-abstract bisimulation (used for the correction of the automate of regions) is much less powerful than the strong time-based bisimulation, because in the latter, the waiting times to leave a state must be the same from two equivalent states, which does not exist in the abstract time-bisimulation.
Deciding whether two timed automates are highly bisimilar is an exptime-complete problem[52].

3.5 Extensions of timed automata: high level languages

Having high-level descriptive languages is very useful to more easily specify systems. For this, several extensions of the timed automata model have been considered. During the presentation of these extensions, we will be interested in the following questions:

- Does the extension preserve the determinability of accessibility?

  If the verification of the systems we model is our goal,ining the determinability of the model is an essential point.

- Does the extension add expressiveness?

  It is important to have models that make it easy to describe more systems.

- Does the extension add concision to the models that can be built? Another important point is the ease of modeling: a smaller model is generally more readable and easier to modify.

3.5.1 The diagonal boundaries

The restrictions on the clocks allowed in the model of the timepieces we have presented are very simple and allow only a comparison between the value of a clock and a constant. Another type of constraint, called diagonal constraints, was mentioned in the original paper[16] and this allows for comparisons of types $x - y \bowtie c$ or$x$ and y, $\bowtie$ is a comparison symbol and c is an entire constant. This extension of timed automates has the following properties :

- Accessibility in timed automates with diagonal constraints is a PSPACE-complete problem [16].

- The use of diagonal constraints does not increase the expressiveness of timed automates [24].

- the use of diagonal constraints adds the concision [28].

The problem of accessibility of this extension was already proved decisive in the original paper [11],, it was also solved by the construction of an automate of regions, which refines the one presented in Section 3.3 and the complexity remains the same. For the second property, this extension was shown more expressively in [24].The principle of this demonstration is to replace one by one the diagonal constraints and to construct an automatic timed without diagonal restraints equivalent according to the strong timed bisimulation ratio.Figure **??** explains how to replace a diagonal tension (in this figure, the goal is to remove the diagonal force $x - y \leq c$ where c is a positive integer).The basic principle is as follows: the flow of time does not affect the truth value of a $x - y \leq c$ diagonal constraint, but this value can be changed if one of the two watches involved in the diagonal restraint is reset.So, two copies of the original automat were built, the $x - y \leq c$ constraint is verified in one copy, while in the other, it is the $X - y > c$ constrain that is checked.When the watch x or y is reset, we go according to the value of the other watch to either of the copies. For example, if the watch y is reset, then if $x \leq c$ is in the copy $x - yc$, whereas if $X > c$ we are in the other copy $X - y > c$. This transformation described above gives a double size of the automate, therefore we have an exponential construction to transform all diagonal constraints. Since timed automates with diagonal constraints are exponentially more concise than standard timed automates, this exponential cost cannot be avoided. This means that we can model systems with timed automates with diagonal constraints exponentially smaller than equivalent standard timed automates.

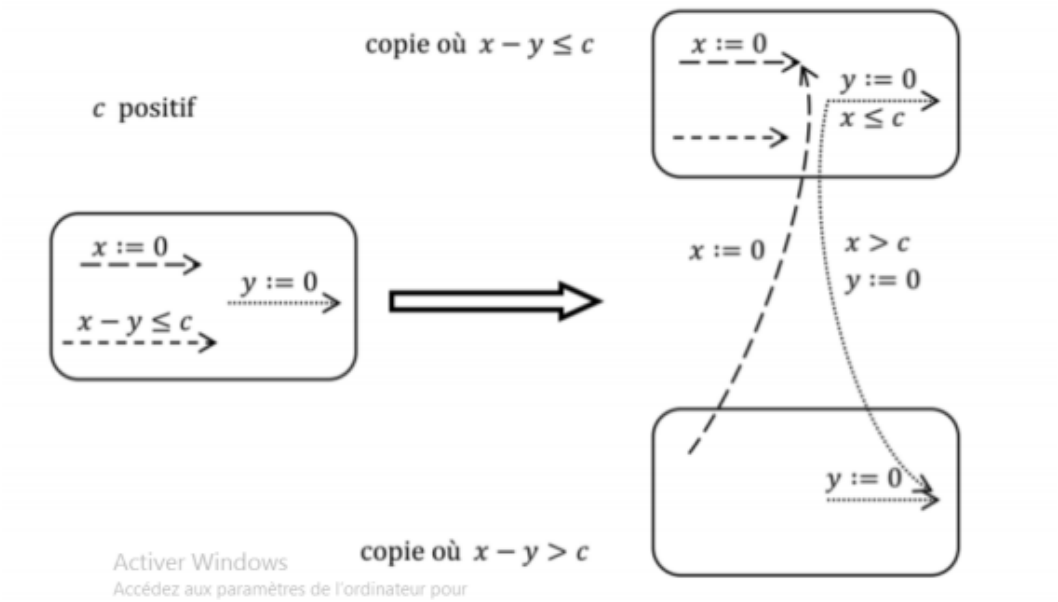

Figure 3.4: Suppression of diagonal constraints.

3.5.2 restricted additives

The timed automata can also be extended by other types of constraints, in this section we will consider the so-called additive constrains that have the shape $x + y \bowtie c$ where x and y are watches, is a comparison symbol and c is an integer positive. The authors of [23] have studied this extension of timed automates that accept languages not recognized by standard timed automates. For example, the Figure3.5 automate accepts a language that no standard timing automate can accept: actions are executed on dates $1/2,\ 3/4,\ 7/8,\ 15/16, ...$

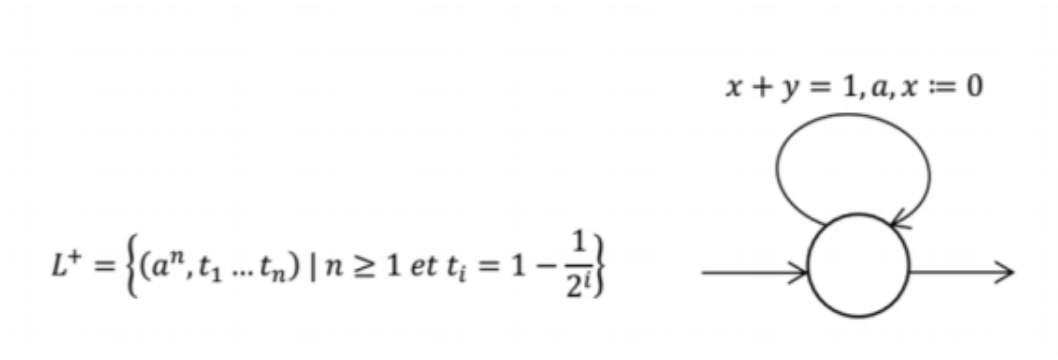

Figure 3.5: Suppression of diagonal constraints.

The properties satisfied by this model of timed automates with additive constraints are[23]:

- he problem of accessibility in timed automates with additive constraints using two clocks is decisive.
- The problem of accessibility in timed automates with additive constraints using at least four watches is unresolved.

The determinability of this model in the case of two watches derives from an extensive construction of an automatic of regions, the region index used is a refinement of the index of regions that we saw in the section 3.3 Figure 3.6 illustrates this construction. In the event that there are four watches, accessibility becomes indecisive. In proof of this latter case, the small automatic of Figure 3.5 has been used several times.

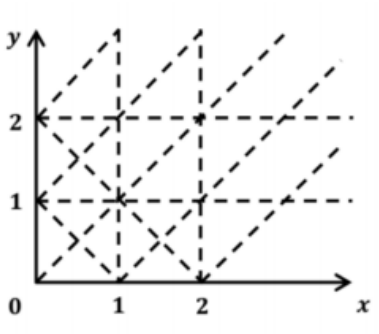

Figure 3.6: Index of regions on two clocks for timing automates with Additional restrictions.

Note 3.1

Regarding the case of three clocks in timing machines with additive constraints, and since it is impossible to build a finished region automat for them in the same way as for standard timing automates, the proof of accessibility remains an open issue.

3.5.3 The Internal Action

The elimination of internal actions, so-called ε-transitions, in finite machines has no influence on the language recognized by these structures, therefore this type of action does not add expressiveness to the finite automate model. On the other hand, in the timed framework, these silent actions add expressiveness to the model and always preserve the decisiveness of accessibility (the automatic of regions is constructed in the same way)[24].

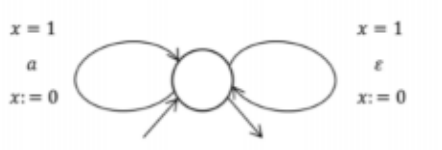

Figure 3.7: A language not recognised by standard timed automates.

he automated timing for Figure 3.7 accepts a language that is not supported by any other standard timing automates. This language contains the timed words on a letter a such that all dates are integers: at each unit of time this machine provides two possibilities of crossings, either the transition marked by a is chosen, or it is the other that is labeled by ε.

3.5.4 Updatable timed automata

In conventional timer machines, the only operation permitted on the clocks is the reset to zero. It is worth taking care of other slightly more complicated operations on the watches. The two forms of the operations of the upgrades are $x :\bowtie c$ and $X : \ bowtiey + c$ where x and$y$ are clocks, is a comparison symbol and c is an integer. For example, the update

- $x :\leq c$ assigns to the watch x a value less than or equal to the constant c in a non-deterministic way,
- $x := y - 1$ affects the watch x to the value of the clock y decremented by 1,
- $x := 0$ corresponds to the classic zero discounts.

A study of this type of timed automates that uses updates was done in [30]As it is possible to increase, decrease, or even set any clock to zero, these powerful types of updates make the accessibility of the overall model indecisive.However, accessibility has been shown to be decisive for several subclasses of this model, here are a few results for these sub-classes presented in [30] : accessibility in timed automates

- with updates of the form $x := c$ is decisive.
- with self-incrementation (c-to-d using updates of the form $x := x + 1$) and without diagonal constraint is decisive.
- with self-incrementation and diagonal constraints is indecisive.
- with self-decrementation (c-to-d that uses updates in the form $x := x - 1$) is indecisive.

The proof of determinability is always based on a construction of automatic regions with fairly fine region indices that refine those seen in the 3.3 section. Figure 3.8 displays a region index for a timed automate that uses diagonal clock constraints $x - y < 1, y > 1$ and $x := 0, y := 1$ updates.The distribution described by fat-pointed lines represents the classical index of regions. But when the update is used $y := 1$, this index will be incompatible with the model being studied because (among other things) the update $y := 1$ makes the clock region grey on Figure 3.8 overlaps several regions (without checking a condition of the time-abstract bisimulation relationship). Then you have to divide again this space of watch valuations by inserting a straight half $x = 2$ (presented by a line in fine dots on Figure 3.8).
It should be noted that it is possible to transform the classes of timed automates with updates that have been proved decisive into standard timed automates with equivalent internal actions according to a low bi-simulation ratio but that preserves the same languages [30]. In addition, these automates with updates are also exponentially more concise than standard timed automates
Finally, we can say that this type of clock operations called “updates” produce very practical macros for modeling real systems.For example, one can cite among the systems that naturally model with this type of operations: ordering problems [42].

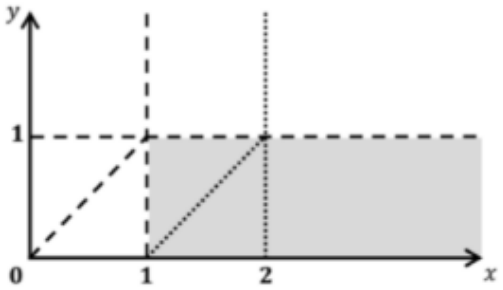

Figure 3.8: A new distribution of regions according to the $\{x - y < 1, y > 1\}$ constraints and the $\{x := 0, y := 1\}$ updates.

3.5.5 Linear hybrid automata

Linear hybrid automates extend the classical timed automates by:

- of general linear constraints, for example $2x - y + 3z < 85$,
- of updates more complex than simple reset operations: we can apply affinity functions to the watches of the Z set on each transition,
- Clocks have different slopes depending on the states: for this reason the clocks in this extension are called dynamic variables.

It has been seen above that the timed automate extensions with additive constraints and extended updates make all verification problems unresolved.The same goes for variables with different slopes: the problem of accessibility in this extension, where a variable can have two different slops, is indecisive. A survey on these concepts and even on decisive subclasses of linear hybrid automates is available in [45] and[70].
In addition, it should be noted the importance of the task of studying and searching for subclasses of linear hybrid automates. Therefore, this task allows us to distinguish:

- of decisive families such as the initialized rectangular automates [45], or
- There are groups in which optimization can be applied.

In addition, there are some subclasses that are of particular importance in the practical world, for example for the modeling of telecommunications protocols,one can use the p-automata family, or use the chronometer automate family, that is, those that have variables with slopes in $\{0, 1\}$, for the preemptive modeling problems of ordering.
So regarding model-checking and to check accessibility properties in linear hybrid automates, there are no general algorithms.But there is a possibility of proposing either semi-algorithms that may not complete the calculation,or approximations that ensure the completion of the computation but do not always give a result (for example, the answer may be "I don't know" or "the system checks the empty set").
The main point of these approaches is based on the use of linear constraints to determine the set of system states to be verified [7]. Polyhedral computing libraries have been used to

deploy semi-algorithms that verify accessibility. HyTech is a tool that can calculate step-by-step state spaces for these linear hybrid automates. As it can also do fixed point calculations (end is uncertain). A detailed description with examples is available in [44] .

3.6 The Subclasses of Timed Automates: Between Expressiveness and Efficiency

This section is reserved to present some restrictions on timed automates.In general, the idea behind these restrictions is to limit ourselves to less expressive models to ensure certain properties or make decision-making algorithms more effective.So the goal of this approach is to find a good compromise between a good model expressivity and an effective algorithm.

It should be noted that the two main reasons that have given importance to this approach to the restriction of timed automates are the parallel composition (the same problem was in the classical timeless framework) and timing (the integration of the clocks into the model), these two aspects have made the task of verifying these models more complicated, even theoretically, the verification of a parallel compound of time-free automates has the same degree of complexity as verifying a single timed automat (the complexity is still PSPACE).Although there are good data structures, called BDDs (Binary Decision Diagram), to curb the combinatory explosion due to parallel composition, and there are other data structure, called DBMs (Diference Bound Matrice), to facilitate the representation of real-time constraints, there are no data structure that gives us an effective solution to address the combination of these two types of explosions.

3.6.1 Automata with "event recording"

In this model, each letter a of the stock alphabet (which are labels for the timed automate transitions) has its own x_a clock such that this clock is reset to zero if a transition labeled by the a share is drawn.This model also implies that the automate can only have this type of clock. Therefore, during the shooting of any transition, there is only the clock associated with the action of that transition that will be reset to zero. So the value of the watch x_a coincides exactly with the time that has elapsed since the last execution of the a proprietary action.

The model of event-recording machines was proposed in [16], this paper also presented another version called the "event-predicting" machines where the clocks indicate the time remaining before the next launch of the a proprietary stock and not the time passed after the last execution of a proprietary action.Negative values were assigned to the watches each time to be able to manipulate this idea.

This subclass did not actually bring any improvements in complexity, so the vacuum test is still PSPACE-complete and the same for model-checking. In return, this subclass of timed automates is determinable, the languages of these models are closed by complementary and therefore the inclusion test between two languages becomes decisive.

3.6.2 Automata with one or two clocks

Accessibility for timed automates with more than three watches has been shown in [37] that it is a PSPACE-complete problem.For this purpose, the study of machines with one or two watches has gained importance in order to develop more efficient algorithms.
Clock automates (AT1H) form a important class as it checks the following properties [50]:

- the accessibility of a location is a NLOGSPACE-complete problem, i.e. it has the same complexity as that in a non-timed graph,
- model-checking for the AT1H that uses $TCTL_{\leq,\geq}$ formulas has a polynomial-time algorithm (so this $TCTL$ restriction does not allow " $= c$" constraints in the "$Until$" terms),
- model-checking of $TCTL$ is a PSPACE-complete problem.

For each (s, γ) configuration and for each (s, x) pair such as $x \in \gamma$, a $f_{s,\ Gamma}$ function is associated to give the minimum duration during taking a φ check path and before ψ is met.These $f_{s,\gamma}$ functions have the following three particular forms:

- $f_{s,\gamma}$ is constant at if the shortest path leading to ψ has a reset transition before any time transition,
- $f_{s,\gamma}$ is decreasing (with a slope equal to -1) when the shortest path has a time transition before any reset, or
- $f_{s,\gamma}$ combines the two above-cited forms, constant for the first interval of and after decreasing.

To define these functions $f_{s,\gamma}$, we must first determine their values at the limits of . Use an algorithm that calculates the shortest path. by Bellman-Ford[35].
Regarding the properties of timed languages, timed automates with a clock offer the determinability for the inclusion test of time-based languages comprising only the finite words [67], however this test remains indecisive for the endless words Abdulla2005.
For two-clock timing machines, a significant leap in complexity has been noticed: the accessibility of a location is a NP-hard problem and the model-checking of CTL (i.e. the formulas do not contain real-time constraints) is now PSPACE-complet [50].

3.7 Discrete-time automata

It is possible to Using natural numbers as a time domain instead of a dense domain is a viable option. However, for most of the problems encountered in timed automata and their restrictions, this change in the time domain does not significantly impact the complexity of verification algorithms. The accessibility and model checking of TCTL (Timed Computation Tree Logic) still remain PSPACE-complete.
Therefore, in order to achieve more efficient algorithms with this time domain change, it is necessary to explore simpler models. Various proposals have been put forward in the literature

to address this challenge. These proposals aim to find alternative modeling approaches that can yield more effective algorithms while utilizing natural numbers as the time domain.
First, we will consider models such as a duration of one unit of time is associated with each transition of the automate. Several studies have studied this simple notion of timing. Model-checking of $TCTL$ is polynomial for this restriction [40] and even for models where each transition lasts 0 or 1 unit of time [53]. This model-checking result for $TCTL$ is unique. If we use the $TCTL_h$ logic with formula clocks, we will have PSPACE-complete problems.
A spontaneous extension of the previous model is to consider transitions with full durations and obtain a model defined as cost automates. Within this framework, it is possible to have polynomial model-checking algorithms for $TCTL_{\leq,\geq}$, however model-checking for$TCTL$ is a Δ_2^p-complete [51] problem. This extension is implanted in the TSMV [64] tool which is an extended version of the NuSMV tool [34].

3.8 Verification in Practice

In order to check the timed machines, it is possible to use a few tools such as Uppaal and Kronos that implement verification algorithms that will be described in this section. As the notion of region, which groups the valuations of equivalent watches, is quite fine and contributes to the growth of inefficiency of algorithms, the construction of the automate of regions (which we presented in Section 3.3) was not useful for practice. So a second symbolic representation, called "zone," of the equivalence relationship has been used by tools that are based on flight algorithms. The interest of the use of zones in practice is at the expense of additional abstraction on behaviors.
To analyze flight systems, we can distinguish two main families of semi-algorithms:

- the first family uses forward analysis: from the initial configurations, this approach iteratively calculates the successors and tests whether the configuration we want to is calculated or not.
- The second uses a backward analysis: it can be said that it is the dual approach of the first, because it iteratively calculates the predecessors of the configurations that we want to and tests whether an initial configuration is among the set of calculated precursors.

These methods can be applied to all state-transition models in order to analyze the different systems, see for example [7] for an application on hybrid automates.
Before presenting these methods of analysis, we will first expose a concept widely used for the verification of timed systems that is symbolic representation.

3.8.1 The zones: a symbolic representation for the implementability of the verification

In order to verify a timed automate that has an infinite set of configurations, it is therefore necessary to have a symbolic representation in order to be able to manipulate these infinite sets of konfigurations. The zones are the most widely used symbolic representation: each zone represents a set of watch valuations that are obtained by a combination of simple time

constraints $x \bowtie c$ or $x - yc$ where x and y are watches, is a comparison symbol and c is an integer. In both forward and backward analysis approaches, the algorithms will manipulate the (s, Z) pairs where s is a localization of the timing machine and Z a zone of watch valuations. In these areas, several operations can be performed:

- the future of a Z zone is defined by $\overrightarrow{Z} = \{v + tannoov \in Z \; and$
Formulas of the first order are applied to the zones to define those operations that preserve the zones.
Now, we describe the simplest method that is backward analysis. Then we will present the forward analysis method that is favoured by most algorithms despite the difficulty of its implementation.

3.8.2 The backward analysis approach

The backward analysis is done as follows: from the states we want to, we calculate the preceding states in one step, then those in two steps,... and at each step we test whether the initial state is among the calculated states. If this is the case, it means that the desired states can be reached from the initial state. If you have completed the calculation and it is not, it means that you can never reach those desired states. The principle of methods based on backward analysis is illustrated in Figure 3.9.

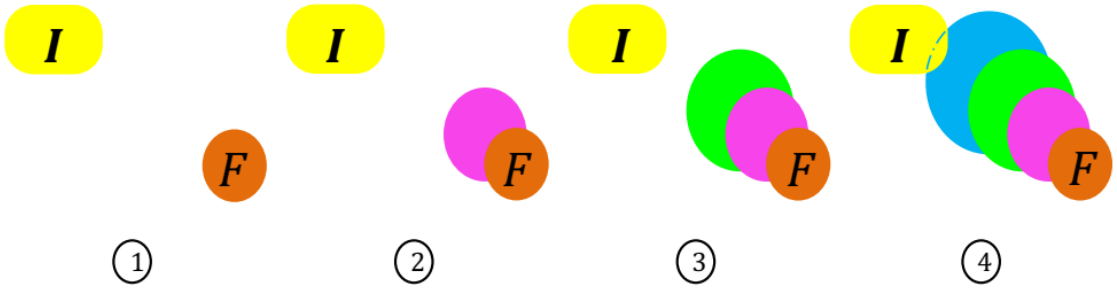

Figure 3.9: The calculation, step by step, of the predecessors of the final state in the backward analysis.

The use of zones makes it easy to calculate a step back from the analysis. If you take a transition from the timed automat $t = s \xrightarrow{G,a,Y} s'$ and if Z' is a zone, then the goal is to calculate the set of predecessors in a step of (s', Z') following the transition t which is exactly a set of configurations (s, v) where v is an valuation of watches belonging to the zone $Z = \overleftarrow{g \cap (Z' \cap Y = 0)[Y]^{-1}}$.
The completion of the iterative calculation of the backward analysis is a very important property. The proof of this property is based on the fact that a zone actually consists of several clock regions. This makes it easy to show that if Z' is a zone, then the Z zone we just described is itself a union of regions. As it has been shown that the number of regions associated with a timed automate is finite, then the amount of (s, Z) pairs that is being calculated in a backward analysis is itself finite.
Practically, forward analysis is implemented in most tools despite the undeniable advantages of backward analysis. The reasons can be summarized in:

- the forward analysis can only reach the system configurations that make sense because they are reachable by real executions.

- the analysis backwards is not suitable with the analysis of systems that use high-level data structures, such as the whole variables bordered or the pieces of C programs,... We find this data structure in tools that implement the analysis forward, for example the Uppaal tool that will be presented in the next section 3.9.

3.8.3 The forward analysis approach

The forward analysis is done as follows: from the initial state, the successive states are calculated in one step, then those in two steps,... and at each step we test whether the states sought are among the states calculated. If this is the case, it means that some of these desired states can be reached from the initial state. If you have completed the calculation and it is not, it means that you can never reach those desired states. The principle of methods based on forward analysis is illustrated in Figure 3.10.
The use of zones makes it easy to calculate a step forward from the analysis. If you take a transition from the timed automate $t = s \xrightarrow{G,a,Y} s$ and if Z is a zone, then the goal is to calculate the set of successors in a step of (s, Z) following the transition t which is exactly all the configurations of (s', v') where v' is an valuation of watches belonging to the zone $Z' = (g \cap \overrightarrow{Z})[Y]$.

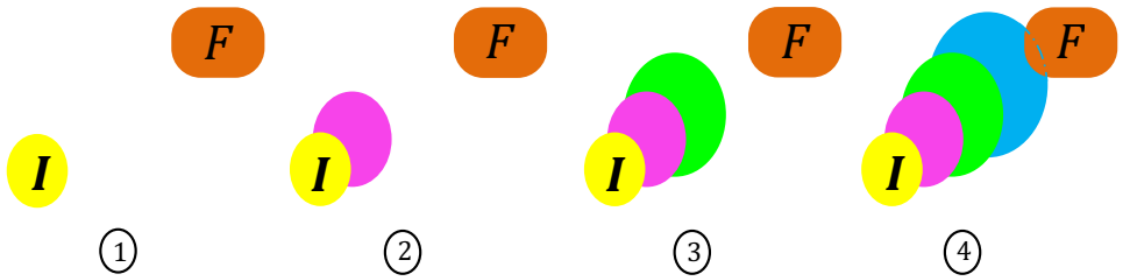

Figure 3.10: The calculation, step by step, of the successors of the initial state in the forward analysis.

Generally the iterative calculation of the forward analysis does not end, unlike the calculations of the backward analysis. Figure 3.11 and Figure 3.12 present a timed automate that illustrates this disadvantage of analyzing forward. The value of the x clock in this example increases by one unit of time for each iteration of the forward calculation, so this calculation will never end.
Usually an abstraction operator is used at each stage of the calculation to deal with this termination problem. This abstraction is called normalization or extrapolation. k is the highest constant with which a timepiece watch has been compared in restrictions. Now that Z is a zone, the extrapolation of Z versus k is the smallest area that includes Z and is defined by constraints that use only constants between $-k$ and $+k$. This extrapolation operation aims: Since all the clock constraints of the timing machine are bounded by k, then what is important is only to know that the value of a watch has exceeded the k boundary because

you don't really need its exact value. Note that using this extrapolation operation during the entire forward analysis ensures the completion of the iterative calculation because the number of zones defined by constraints that use only $[-k + k]$ constants is finite.
Furthermore, this forward analysis approach can create another problem with each calculation iteration: a calculated overapproximation of accessible states. So this iterative calculation can return to us states declared as accessible but in reality they are not.
It has been shown in [27] that this forward analysis approach is correct if we consider timed automates without diagonal constraints, however it is incorrect for the class of automates with diagonal restraints.

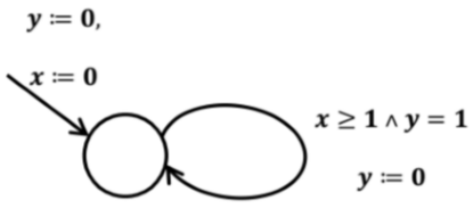

Figure 3.11: A timed automate that illustrates the non-finishment of the forward analysis.

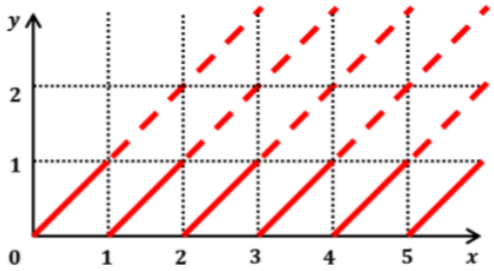

Figure 3.12: Iterative advance calculation.

3.8.4 DBMs: a daTA structure for the representation of zones

A DBM (Difference Bound Matrix) is a classical data structure that represents differential constraints systems[35]. Furthermore, this concept of DBM has gained particular interest in the verification of timed systems because it allows an adequate representation of a concept quite demanded by verification tools that is the zones. It was proposed in [26] for the analysis of temporal Petri networks, then it is heavily used to analyze the timed automates[39].
Which is n the cardinality of all the watches used, a DBM M is a square matrix of size $(n + 1) \times (n + 1)$. Typically, the coefficients of M are pairs of $(m, \prec)$ such that m is an integer and the symbol $\in \{<, \leq\}$. If $M = (m_{i,j})_{0 \leq i,jn}, \{x_i, i1i \leq n\}$ is the set of clocks and x_0 is a fictitious clock that is always worth 0, then for each constraint of the form $X_i - x_j m_{I,J}$ one can directly acquire the coefficient $M_{i,J}$ of the DBM. However, the coefficient $m_{i,0} = 7$ in the case of a $x_i \leq$ form coercion is 7 because this coertion can easily be rewritten in the

$X_i - x_0$form 7. The following example refers to a DBM defined by the constraints ($x_1 \geq 2$) $\wedge(x_2 \leq 4) \wedge (x_1 - x_2 - \leq 3)$and illustrated in Figure 3.13:

$$\begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{pmatrix} +\infty & -2 & +\infty \\ +\infty & \infty & 3 \\ 4 & \infty & +\infty \end{pmatrix} \end{array}$$

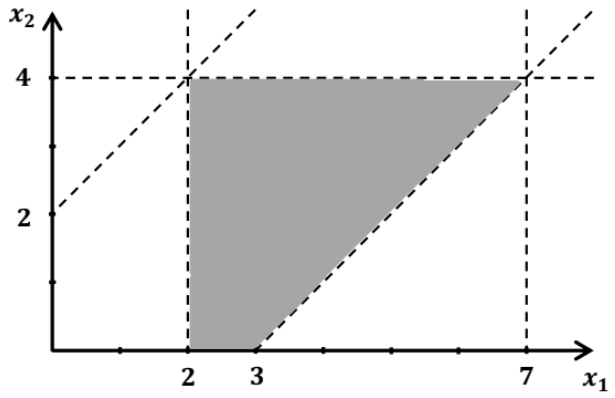

Figure 3.13: A zone defined by constraints $(x_1 \geq 2) \wedge (x_2 \leq 4) \wedge (x_1 - x_2 \leq 3)$.

If a coefficient is equal to $+\infty$, it means that there is no constraint on the difference between the two watches linked to the coefficent. The value of the $m_{0,1} = -2$ coefficient is determined from the $x_1 \geq 2$ coercion that must be rewritten in the form of another equivalent coertion using the fictional clock $x_0 - x_1 \leq -2$.
Every DBM characterizes a zone, and each zone can be characterized by one or more DBMs. In the above DBM, if you change the $+\infty$ value of the $m_{1,0}$ coefficient to 9, this does not change the drawn area and then you get a second DBM of the same zone. A shorter path algorithm, such as Floyd-Warshall [35], can be applied to provide DBMs in a normal form. These algorithms choose the strongest constraints the DBMs that characterize the zones. If we put the previous example of a DBM in normal form, we will have the following DBM equivalent:

$$\begin{pmatrix} 0 & -2 & 0 \\ 7 & 0 & 3 \\ 4 & 2 & 0 \end{pmatrix}$$

SO, DBMs represent the basic data structures that are used to determine the areas for the implementation and manipulation of configuration sets of timed automates. However, these structures have undergone several improvements, such as minimizing DBMs [56] and using Clock Difference Diagrams (CDDs) to compact the representation of DBM unions [59].

3.9 Example of verification tools: Uppaal

Uppaal is a tool developed by the University of Uppsala (Sweden) and Aalborg University (Danish)[56].It is used to verify temporary systems. This tool is free at http://www.uppaal.com.

It is designed to do verification on a variant of the timed automates. This variant is wider syntaxically because it allows a direct representation of the

- Emergency constraints: in the case of a transition that must be taken immediately, without waiting,
- Atomic constraints: When one is forced to instantly draw a sequence of transitions,

These modelling facilities improve the concision and readability of the original model but do not add expressiveness.

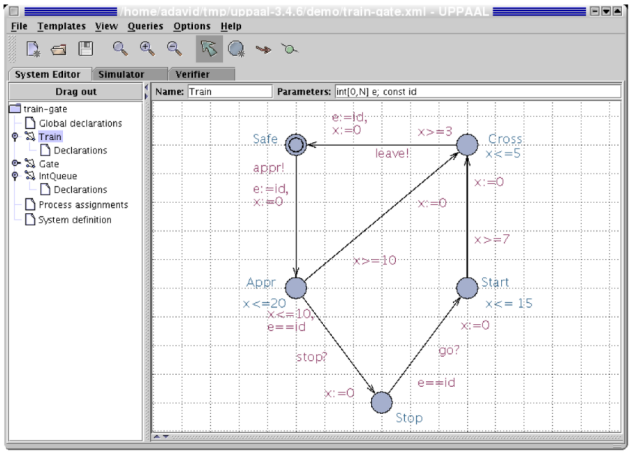

Figure 3.14: An example of modeling by Uppaal.

The Uppaal tool offers the ability to verify a subset of $TCTL_h$ that mainly gathers the properties of accessibility, security and also the responsive properties. For more details on using this tool, a tutorial is available at the web address mentioned above [19].

Chapter 4

Construction of regions automaton for a daTA model with relative time

Sommaire

4.1 Introduction

A Duration Action Timed Automata with Relative Time (daTA-R) is an extension of the traditional Timed Automata framework that incorporates relative timing constraints. To represent the duration of actions, each edge in the automaton is annotated with clock constraints that encompass them. These clock constraints implicitly define the temporal boundaries of the actions, taking into account those actions that have already started. A single clock is reset at each edge, signifying the initiation of an event or action.
The completion of an action is captured through temporal formulas associated with the locations of the automaton, specifically at the destination location. By examining the temporal formulas at the destination location, one can determine whether the action has concluded or if it is still ongoing. If there is a dependency between successive actions, the duration of an action is typically specified within the constraint of the following edge. However, if there is no direct dependency, the duration information is usually encoded in the subsequent locations, indicating that the action is still in progress.

This elegant approach of utilizing clock constraints and temporal formulas enables the daTA-R structure to effectively represent and capture the durations of actions, all while adhering to the principles of maximality semantics.An example of a daTA is shown in figure4.1.The automaton consists of two localities S_0,S_1 and one clock x. A transition from $S_0$to S_1 represents the start of action a (indicating the beginning of its execution).

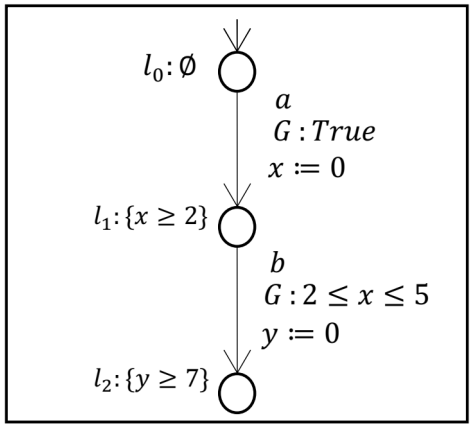

Figure 4.1: daTA-R

Assuming a time granularity of seconds, the automaton A starts in locality l_0. As soon as the value of y is less than or equal to 4, the automaton can make an a transition to l_1 and reset the x clock to 0. On the locality l_1 the temporal formula $\{x \geq 2\}$ represents information about the duration of the action a (it is important to differentiate it from invariant in timed automata). When x is at most 5 and is at least 2, transition to l_2 can be started (b executed) and y is reset. In the same logic the temporal formula $\{y \geq 7\}$ represents duration of the action a.

4.1.1 Preliminaries

In the following we consider $\mathbb{R}_{\geq 0}$ a set of nonnegative real numbers. Clocks are real variables take values from $\mathbb{R}_{\geq 0}$. Let M, ranged over $x, y, \ldots$, be a set of clocks and consider $H \subset M$. A clock valuation over H is a function that assigns a nonnegative real number to every clock. V_H is the set of total valuation functions from H to $\mathbb{R}_{\geq 0}$. A valuation is noted $v \in V_H$, and for $d \in \mathbb{R}_{\geq 0}$, $v + d$ maps every clock x to $v(x) + d$. For $\lambda \subseteq H$, $v[\lambda := 0]$ indicates the valuation for H which assigns the value 0 to each $x \in \lambda$, and agrees with the valuation v for the other clocks of H.
The set $C(X)$ of clock constraints C is defined by the grammar:

$$C ::= true \mid false \mid x \sim c \mid C \wedge C,$$

where $x \in X$, $c \in \mathbb{R}_{\geq 0}$ and $\sim \in \{<, >, =, \leq, \geq\}$. We write $v \models C$ when the valuation v satisfies a clock constraint C over X iff C evaluates to *true* according to the values given by v.
We also use a subset of constraints where only the atomic form of clocks comparison is allowed. This set is defined by $C_d(H)$ by the grammar: $C ::= x \geq c$, where $x \in H$ and $c \in \mathbb{R}_{\geq 0}$. This subset represents condition duration over the set of actions noted by Act.

4.2 Timed automata with action durations and relative speed clocks model

In this section, we present the basic model called duration action timed automata on which we introduce the notion of relativity between clock frequencies. This is achieved by the asynchronous product of these automata.

4.2.1 Definition 4.1

a data-R is a tuple (S, L_s, s_0, H, T_D) of the support ACT where over:

- S is a finite set of locations,
- $l_s : S \rightarrow 2_{fn}^{\phi_t(H)}$ is a function which assigns to each state S the set F of ending conditions (duration conditions) of actions possibly in execution in S ,
- $S_0 \in S$ is the initial state, such that $L_s(s_0) = \emptyset$,
- H is a finite set of variables named clocks ,
- $T_D \subseteq S \times 2_{fn}^{\phi_t(H)} \times 2_{fn}^{\phi_t(H)} \times ACT \times H \times S$ is the set of transitions.
- A transition (s, G, D, a, x, s') represents a switch from state s to state s' by starting execution of action a and resetting clock x. G is the corresponding guard which must be satisfied to fire this transition. D is the corresponding deadline which requires, at the moment of its satisfaction, that action a must occur. (s, G, D, a, x, s') can be written $s \xrightarrow{G,D,a,x} s'$.

Definition 2 (daTA with relative time rates)

A daTA with relative time rates (daTA-R) over the set of agents Ag is a structure $A = (\mathcal{A}, \pi)$ where $\mathcal{A} = (S, s_0, H, T, L_S)$ is a daTA and π is a distribution of the clocks of H over the agents, defined as a subjective mapping from H to Ag (for each $p \in Ag$, we have $\pi^{-1}(p) \subseteq H$). Observe that several clocks can be assigned to each agent, said to belong to that agent. The evolutions of the clocks of an agent behave synchronously. In a daTA with relative time rates, the clocks that belong to different agents evolve synchronously but under the relative time rates assigned to these agents. Each such rate assigned to the agent p represents the speed of p and depends on some global time reference, from which the local time of p is derived. Note that each rate can be given by the function $\tau_p : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ with $\tau_p(0) = 0$ and $\tau_p(t)$ returns the local time in each agent $p \in Ag$ for the instant t of absolute time. Moreover, τ is a tuple of local time rate functions such that $\tau = (\tau_p)_{p \in Ag}$. The function τ_p is the p local time rate. For a time value t, the mapping function $\tau : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}^{Ag}$ assigns the tuple $(\tau_p(t))_{p \in Ag}$ to $\tau(t)$.

4.2.2 Semantical Rules to Build a daTA-R

In the study by Alur and Dill [10], they propose an equivalence relation to group states of a timed transition system, where each equivalence class represents a set of states. These classes are referred to as clock regions, capturing different valuations of clocks within the system. Achieving coherence in the design of a multi-agent system requires that all components share a common perception of time, as emphasized by Lenzen [61]. This shared perception of time is crucial for maintaining consistency in the system. It becomes particularly significant when calculating the semantic graph of the model, as it affects the interpretation of states and transitions. Consequently, it becomes necessary to redefine concepts like clock regions and region automata to account for this unified perception of time. In the subsequent discussion, we will focus on the impact of relative clock speeds.

4.2.3 Equivalence Classes of Clock Valuations

Let $A = ((S, L_S, s_0, H, T), \pi)$ be a daTA-R over Act and a set of agents Ag.

Definition 4 ($slope_{xy}$)

Let x (resp. y) be a clock that belongs to the agent p (resp. q) and evolves according to the rate function τ_p (resp. τ_q). We define $slope_{xy}$ as the ratio of local-time rate functions τ_q and τ_p, noted $slope_{xy} = \tau_q/\tau_p$.(see Figure 4.2).

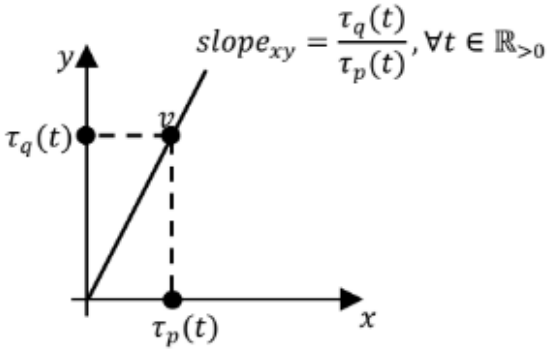

Figure 4.2: Two clocks evolution with different speeds.

Given a pair of clocks, x and y (within respectively agent p and q), their owner speeds will make them diverging from time reference at a certain speed. That is equal to the ratio between their owner rates. It represents the slope of the straight line in Figure 4.2. As there are only finitely many clock constraints on clock x, we can determine the largest integer $c_x \in \mathbb{N}$ with which x is compared in some clock constraint (guard) of the daTA-R A. In the remainder and for every pair of clocks x and y, the parameter $slope_{xy}$ is assumed be an integer constant whatever the value of time t.

Definition 5 (equivalence relation $\sim$)

We define the equivalence relation $\sim$ over the set of all clock valuations[60],$v \sim v'$ iff all the following conditions hold:

1. For all $x \in H$, either $v(x)$ and $v'(x)$ are the same, or both $v(x)$ and $v'(x)$ are greater than c_x.

2. For all $x, y \in H$ with $v(x) \leq c_x, v(y) \leq c_y$ and x (resp. y) evolves according to τ_p (resp. τ_q):

   (a) $c \cdot \frac{1}{slope_{xy}} \leq v(x) \leq (c+1) \cdot \frac{1}{slope_{xy}}$ iff $c \cdot \frac{1}{slope_{xy}} \leq v'(x) \leq (c+1) \cdot \frac{1}{slope_{xy}}$ $for\ c \in \mathbb{N}$

   (b) $fract(slope_{xy} v(x)) \leq fract(v(y))$ iff $fract(slope_{xy} v'(x)) \leq fract(v'(y))$

An equivalence class of clock valuations induced by $\sim$ is a clock region of A.

4.2.4 The Representation of Clock Regions

Each equivalence class of clock valuations can be specified by a finite set of clock constraints it satisfies. The notation $[v]$ represents the clock region to which v belongs.

Example

We consider two clocks x and y which evolve at different rates such that $slope_{xy} = 2$, $c_x = 2$ and $c_y = 2$. The clock regions obtained by the Definition 5 (equivalence relation) are depicted in Figure 4.3 . Thus, we have 15 corner points (e.g. $[x = 0, 5 \wedge y = 2]$), 38 open line segments (e.g. $[0 < 2x = y < 1]$) and 23 open regions (e.g. $[0 < 2x < y < 1]$).

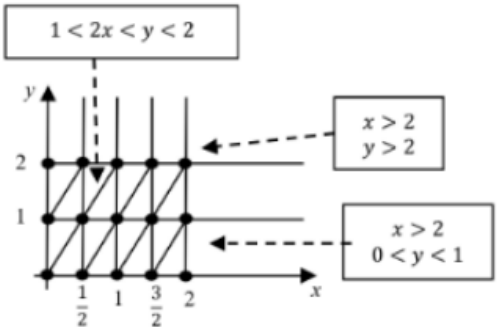

Figure 4.3: Clock regions deducted by the relation $\sim$.

Definition 6 ($slope_{max}$)

For each clock $x \in H$, we define $slope_{max(x)}$ as the largest value of $slope_{xy}$ for all $y \in H$. Reconsider the example above:

- $slope_{max(x)}$=$max(slope_{xy}, slope_{xx})$=$max(\frac{2}{1}, 1)$=2.

- $slope_{max(y)}$=$max(slope_{yx}, slope_{yy})$=$max(\frac{1}{2}, 1)$=1.

Note that if x is the fastest clock in H then $slope_{max(x)} = slope_{xx}$=1, moreover $1/slope_{max(x)}$ is the smallest amount of time in which x cannot stay in the same region. In the example, the clock x changes the region each half unit of time corresponding to $1/slope_{max(x)}$ =1/2, when y do this change each one unit of time (except for regions represented by points).
The representation of a clock region accords with the two following points:

- For each clock x which evolves according to τ_p, there is one clock constraint taken from
$$\left\{x = c \mid c = 0, \frac{1}{slope_{xy}}, 2 \cdot \frac{1}{slope_{xy}}, \ldots, 1, 1 + \frac{1}{slope_{xy}}, 1 + 2 \cdot \frac{1}{slope_{xy}}, \ldots, c_x \quad for\ all\ y \in H\right\}$$
the set: $$\bigcup \left\{ \bigwedge_{y \in H} \left(c - \frac{1}{slope_{xy}} < x < c\right) \mid c = \frac{1}{slope_{xy}}, 2 \cdot \frac{1}{slope_{xy}}, \ldots, 1, 1 + \frac{1}{slope_{xy}}, 1 + 2 \cdot \frac{1}{slope_{xy}}, \ldots, c_x \right\}$$
$$\bigcup \{x > c_x\}. \quad (1)$$
- For each pair of clocks x and y which evolve respectively according to τ_p and τ_q such that $c < x < c + \frac{1}{slope_{max(x)}}$ and $d < y < d + \frac{1}{slope_{max(y)}}$ appear in (1) for some c and d, whether $slope_{xy}(x - c)$ is less than, equal to or greater than $y - d$.

4.2.5 The time-successors of clock regions

In the following, we introduce the time-successor relation over clock regions. When time advances from any clock valuation v in the region α, we will reach all its time-successors α'. Formally, we say that α' is a time-successor of the region α if there are v in α, v' in α', $t \in \mathbb{R}_{>0}$ such that $v' = v \bigoplus \tau(t)$, with $v \bigoplus \tau(t) = (v(x) + \tau_p(t))_{\pi^{-1}(x)=p}$. For example, in Figure 4.3 the five time-successors of the region $\alpha = [(1.5 < x < 2), (1 < y < 2x - 2)] are : itself, [(x = 2), (1 < y < 2)], [(x > 2), (1 < y < 2)], [(x > 2), (y = 2)] and [(x > 2), (y > 2)]$. These regions are those covered by a line drawn from any point in α parallel to the line $y = slope_{xy} \cdot x = 2x$ (in the upwards direction). To compute each time-successor of a region α, we must give:

[label=()] For every clock x, a constraint of the form $(x = c)$ or $\left(c < x < c + \frac{1}{slope_{max(x)}}\right)$ or$(x > c_x)$ and For every pair x and y such that $\left(c < x < c + \frac{1}{slope_{max(x)}}\right)$ and $\left(d < y < d + \frac{1}{slope_{max(y)}}\right)$ appear in 1, the ordering relationship between $slope_{xy}(x - c)$ and $y - d$.

To compute the possible time-successors, three cases are distinguished:

First case

Each clock x in the region α satisfies the constraint $(x > c_x)$, so α has only one time-successor, itself.
This is the case of region $[(x > 2), (y > 2)]$ in Figure 4.3.

Second case

This case is considered when there is at least, in the region α, one clock x which satisfies the constraint $x = c$ for some $c \leq c_x$. The set H_0 contains all clocks appearing in similar

constraint form as x. The clock region α will be changed immediately when the time advances, because the fractional part of each clock in H_0 becomes different from 0. The clock regions α and β have the same time-successors where β is specified by:

2. A set of clock constraints which can be given as follows:
   [label=For each clock $x \in H_0$:[label=]
   (a) i. If α satisfies $(x = c_x)$ then β satisfies $(x > c_x)$;
   ii. If α satisfies $(x = c)$ then β satisfies $\left(c < x < c + \frac{1}{slope_{max(x)}}\right)$.
   (b) For each clock $x \notin H_0$, the clock constraint in α remains the same in β.
   (c) The ordering relationship between $slope_{xy}(x - c)$ and $y - d$ of each pair of clocks x, y in α is the same as that in β, such that $x < c_x$ and $y < c_y$ hold in the region α.

For example, in Figure 4.3 the time-successors of the region $[(x = 0), (0 < y < 1)]$ are the same as the time-successors of the region $[0 < 2x < y < 1]$.

Final case

If the first and the second case do not apply, then let H_0 be the set of clocks x for which the region α satisfies two constraints, $c < x < c + \frac{1}{slope_{max(x)}}$ and $slope_{xy}(x - c) \geq y - d$, for all clocks y for which the region α satisfies $d < y < d + \frac{1}{slope_{max(y)}}$. Thus, when time advances, clocks in H_0 take the values $\frac{1}{slope_{max(x)}}$. Therefore, the time-successors of the region α are α, β and all the time-successors of β which is specified by :

1. A set of clock constraints which can be given as follows:

   [label=.]For each clock $x \in H_0$, if α satisfies $\left(c < x < c + \frac{1}{slope_{max(x)}}\right)$ then β satisfies $\left(x = c + \frac{1}{slope_{max(x)}}\right)$; For each clock $x \notin H_0$, the clock constraint in α remains the same in β.

2. For each pair of clocks x and y such that $\left(c < x < c + \frac{1}{slope_{max(x)}}\right)$ and $\left(d < y < d + \frac{1}{slope_{max(y)}}\right)$ appear in (1.2), the ordering relationship between $slope_{xy}(x - c)$ and $y - d$ in α remains the same in β.

For example, in Figure 4.2 the time-successors of the region $[0 < 2x < y < 1]$ include itself, $[(0 < x < 0,5), (y = 1)]$ and all time-successors of $[(0 < x < 0,5), (y = 1)]$.

4.3 The region automaton

The above definitions about equivalence relation over the set of all clock valuations and the time-successor relation over these equivalence classes allow defining the region automaton. The region automaton $R(A)$ of the daTA-R A is formed by a set of configurations and a transition relation over this set. Each configuration $< s,\alpha >$ records a location s of the

automaton A and a clock region α of current values of the clocks. The initial configuration of $R(A)$ is $< s_0, [v_0] >$ with s_0 is the initial location of A and $[v_0]$ is the region of the initial valuation of clocks that maps each clock in H to 0. The transition relation of $R(A)$ joins $< s$,α¿ and $< s', \alpha' >$ with the label a iff there is a transition labeled with a from location s with the clock valuation $v \in \alpha$ to location s' with the clock valuation $v' \in \alpha'$ in the daTA-R A. For a clock region α and a set of clocks $\lambda \subseteq H$, the region $\alpha[\lambda := 0]$ denotes the reset of all clocks of λ in any valuation v of α. Formally $\forall v \in \alpha, \forall x \in H$:

if $x \in \lambda$ then $v(x) = 0$ in $\alpha[\lambda := 0]$
else $v(x)$ in $\alpha[\lambda := 0]$ is the same in α.

4.3.1 Algorithm of construction of regions automaton

Let $A = ((S, s_0, H, L_S, T), \pi)$ be a daTA-R over the set of agents Ag. The region automaton $R(A)$ is an automaton over the alphabet Act such that:

- The configurations of $R(A)$ are of the form $< s|\alpha >$ where s is a location of A and α is a clock region.

- The initial configuration is of the form $< s_0|[v_0] >$ where $v_0(x) = 0$ for every $x \in H$.

- A transition of $R(A)$, from the configuration $< s|\alpha >$ to $< s'|\alpha' >$, is labeled by $a \in Act$ iff there is a transition (s, G, a, x, s') in T and a clock region α'' which satisfies

  - α'' is a time-successor of the region α,
  - α"$\models$G and
  - α'=[$\{x\}$:=0] α".

Figure 4.4 highlights a part of the regions graph of the frame example.

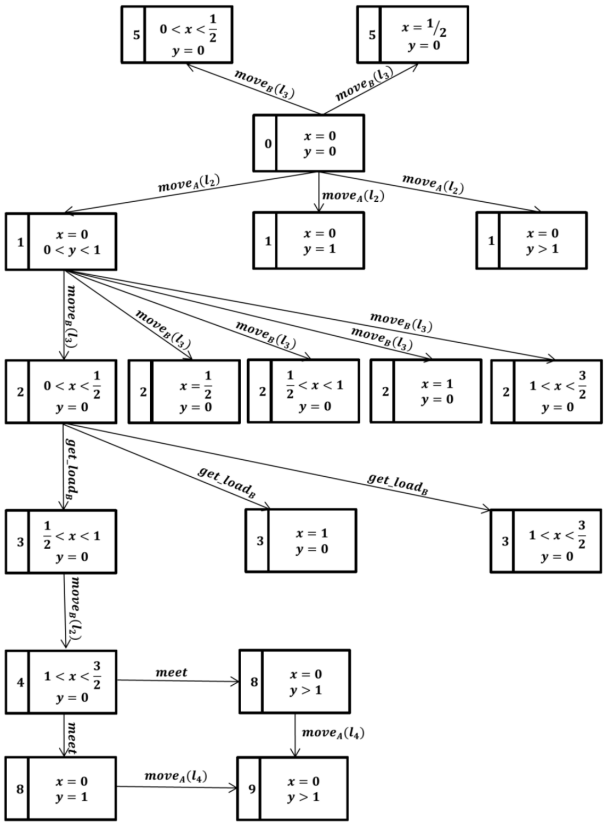

Figure 4.4: A part of regions automaton.

4.4 Algorithm for calculating time-successors of clocks region

Var:
β , α
Pile-région; pile-ordre ; pile-CT :pile
Pile-vide (pile-région) = Faux
$\alpha \leftarrow$dépilé (pile-région)
if *'=' ∈ operator (α. pile-CT)* **then**
  α appartient au 2ème cas;
else
  if *operator (α. pile-CT) =*{>} **then**
    α Appartient au 1èr cas;
  else
    α Appartient au 3éme cas;
    if $\alpha \in$ {*région du 1èr cas*} **then**
      $\beta = \alpha$;
    end
    if $\alpha \in$ { *région du 2ème cas*} **then**
      β. pile-ordre = α. pile-ordre;
      while *CTA ← dépiler (α. pile-CT* **do**
        CTA ← dépiler (α. pile-CT)
        if *operators (CTA)≠ { = } alors* **then**
          Empiler (CTA, β. pile-CT);
        else
          if *val (CTA) ValMax (clock Id (CTA))* **then**
            CTA ← clock Id (CTA) ¿ ValMax (clock Id (CTA));
            Empiler (CTA, β. pile-CT);
          else
            CTA ← Val (CTA) $\prec$ clock Id (CTA)$\prec$ Val (CTA) + $\frac{1}{slopeMax(clockId)}$;
          end
        end
      end
    end
  end
  if $\alpha \in$ {*région de 3ème cas*} **then**
    β. pile-ordre = α. pile-ordre;
    X ←clockIdMax (dépiler (α. pile-ordre));
    while *(pile-vide (α. pile -CT) =faux)* **do**
      CTA ← dépiler (α. pile-CT;
      if *clockId (CTA) ≠ X* **then**
        Empiler (CTA, β. pile-CT ;
      else
        CTA ← clockId (CTA) =val-sup (CTA);
        Empiler (CTA, β. pile-CT);
      end
    end
  end

The functions used in the algorithm

- **Operators**:returns the set of operators present in alpha and pile-CT.
- **ClockId**:returns the set of operators present in alpha and stack-CT.
- **ClockIdMax**:returns the maximum CTA clockid.
- **pile-vide**:function to check if a pile is empty or not.
- **val**:returns the value associated with the CTA.
- **valmax**:returns the mmaximum value associated with the clock identify.
- **valsup**:returns the successor value of the clock identify.

4.5 Conclusion

In this chapter, an approach for the analysis of timed systems has been proposed where the clocks evolve with relative frequencies. The proof of decidability has was based on an abstraction, called region, of the behavior of the system, for this, the necessary concepts are redefined. As a continuation of this contribution, it remains to study the power of expressiveness of duration action timed automata with relative velocities time and to study once again the effect of the other relations between these frequencies of clocks. This can be achieved by exploring the different forms and values of the parameter slope other direction consists in exploring the capacities of this model for describe the semantics of ambient system specifications.

Chapter 5

Implementation and realization

Sommaire

5.1 Introduction

In this chapter, we will present class diagrams for modeling daTA-R and the region automaton as well as the java development language with the development environments required for the development and creation of our app.

5.2 Class Diagram

A class diagram is a type of UML (Unified Modeling Language) diagram used to model the static structure of a software system, with emphasis on classes, attributes, methods, and relationships between classes. It provides an overview of the system architecture and facilitates understanding of the class hierarchy, associations and inheritance. The class diagram is widely used in software development to plan, design, and communicate the structure of an object-oriented system.

5.2.1 Class diagram of daTA-R

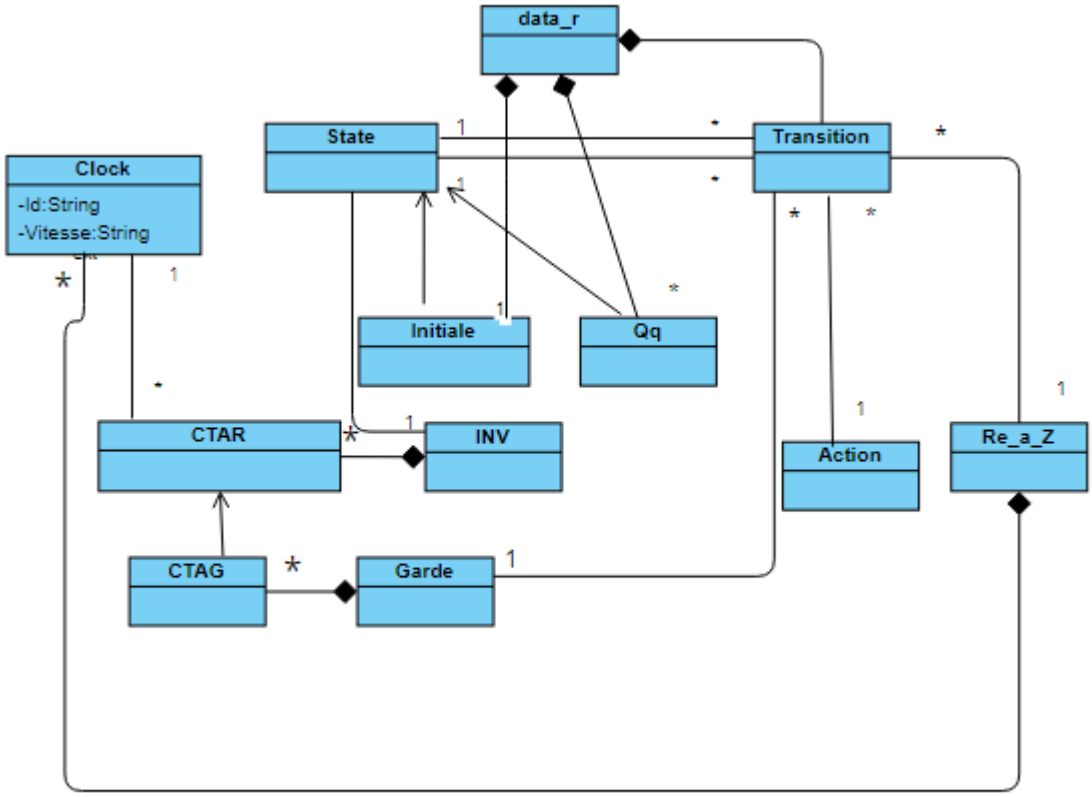

Figure 5.1: Class diagram of daTA-R

5.2.2 Class diagram of Region automaton

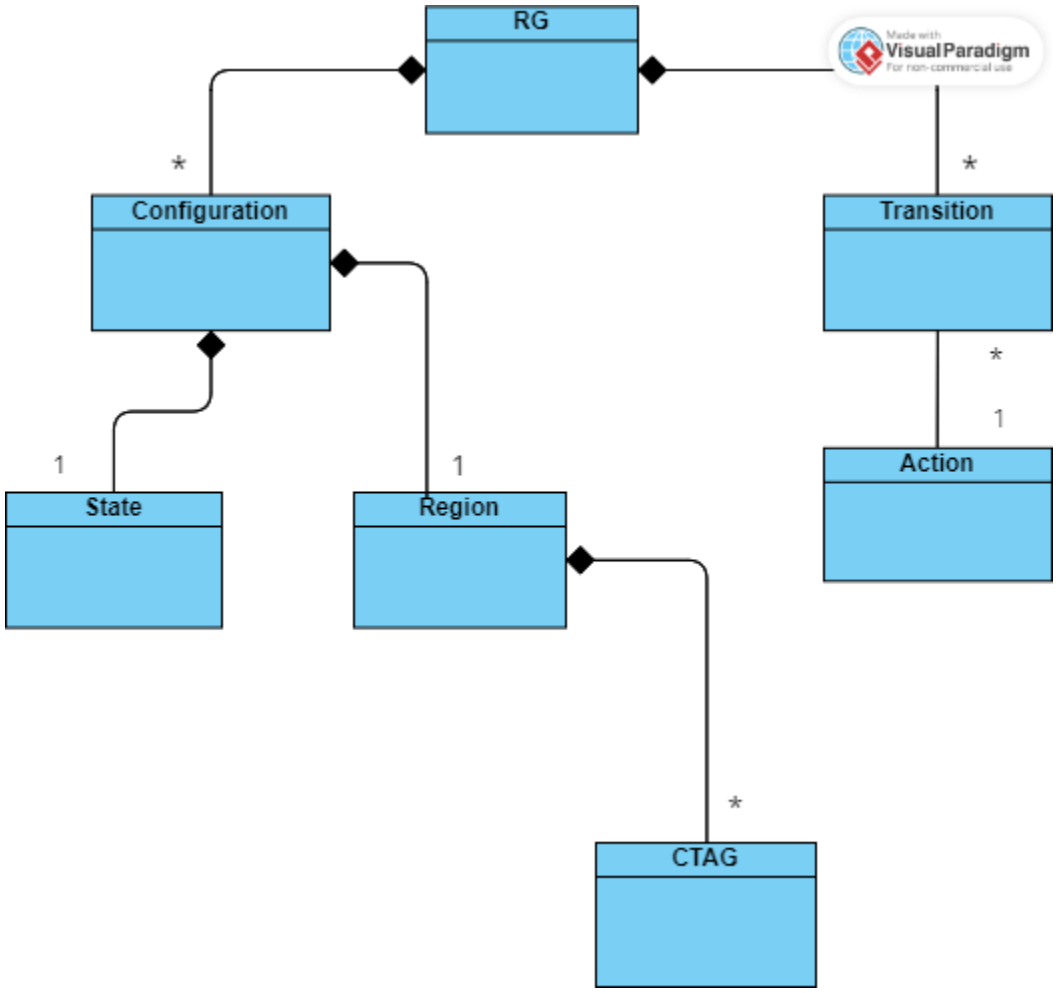

Figure 5.2: Class diagram of Regions automaton

5.3 Software tools

5.3.1 Tools for implementation

- **Overleaf:**
  Overleaf is a collaborative cloud-based LaTeX editor used for writing, editing and publishing scientific documents. The official journal partners with a wide range of scientific publishers to provide LaTeX templates and direct submission links.

Figure 5.3: Logo of Overleaf

- **JAVA language:**
  Java is an object-oriented language, used for the development of various types of application. It is characterized by its probability due to the use of a virtual machine the JVM (Java Virtual Machine) which interfaces between the program and the operating system.

Figure 5.4: Logo of JAVA

- **IntelliJ IDEA:**
  IntelliJ IDEA is an Integrated Development Environment designed specially for the Java language by the company Jetbrains

Figure 5.5: IntelliJ IDEA Logo

- **Visual Paradigm:**
  Visual Paradigm is a UML modeling tool supporting UML 2 and Business Process Modeling Notation (BPMN). In in addition to modeling support, it provides capabilities for report generation and code engineering, including the generation of coded. In this project we used Visual Paradigm for modeling class diagrams.

Figure 5.6: Visual Paradigm Logo

- **Graphstream:**
  Graphstream is a Java graph management library. His goal principal is the modeling of dynamic interaction networks of different sizes.The purpose of the Library is to provide a means of representing graphics and work on them.

Figure 5.7: Graphstream Logo

5.4 Source code

5.4.1 Source code structure

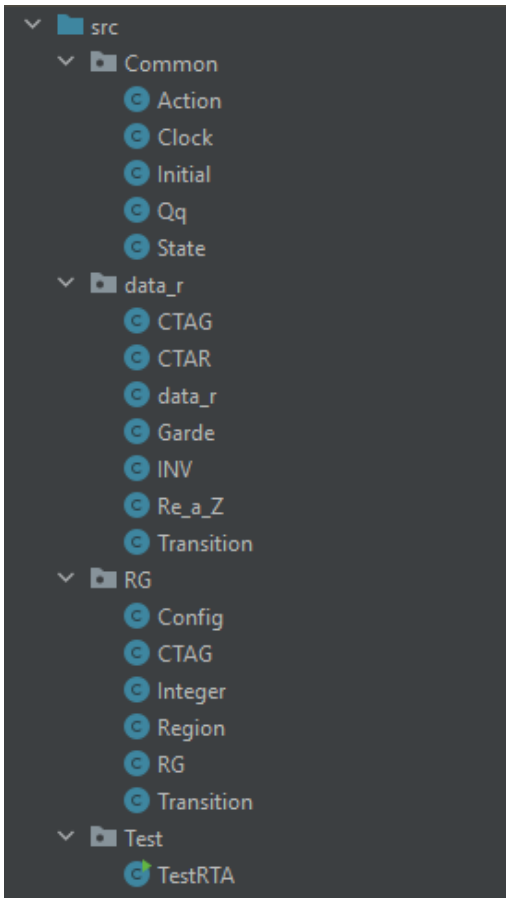

Figure 5.8: Source code structure

5.4.2 Main interface

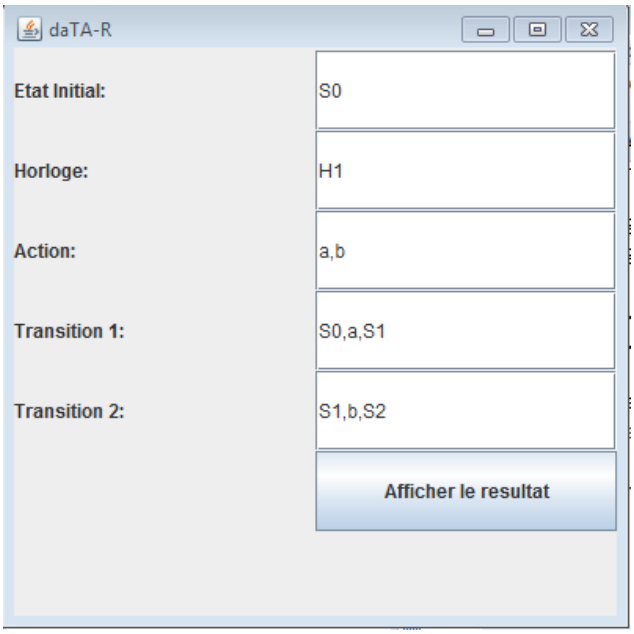

Figure 5.9: Main interface

5.4.3 Transform source code

```
class AutomataGraph {
public static void drawAutomata(String etatInitial, String horloge, String action,
    String transition1, String transition2) {
        Graph graph = new SingleGraph("Region Automata");
        graph.setStrict(false);
        graph.setAutoCreate(true);
        Node s0 = graph.addNode("S0");
        Node s1 = graph.addNode("S1");
        Node s2 = graph.addNode("S2");
        Edge e1 = graph.addEdge("S0S1", "S0", "S1");
        Edge e2 = graph.addEdge("S1S2", "S1", "S2");
        e1.setAttribute("ui.label", transition1);
        e2.setAttribute("ui.label", transition2);
        Viewer viewer = graph.display();
        viewer.setCloseFramePolicy(Viewer.CloseFramePolicy.HIDE_ONLY);
        String result = "Etat Initial: " + etatInitial + "\n"
                + "Horloge: " + horloge + "\n"
                + "Action: " + action + "\n"
                + "Transition 1: " + transition1 + "\n"
                + "Transition 2: " + transition2;
JOptionPane.showMessageDialog(null, result, "Affichage", JOptionPane.INFORMATION_MESSAGE);
```

Figure 5.10: Method to schematize the daTA-R structure

5.5 Execution example

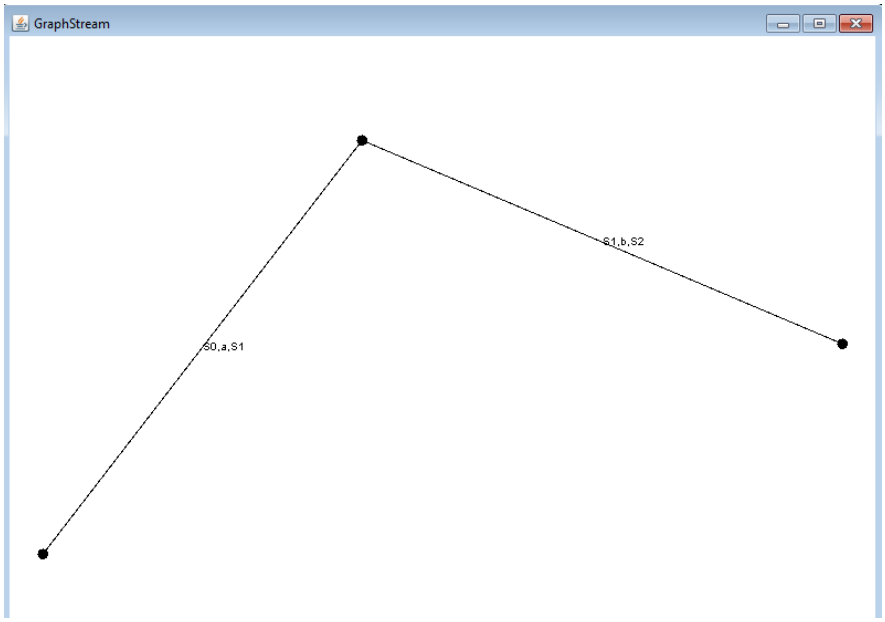

Figure 5.11: Execution example

5.6 Conclusion

In this chapter, we have presented the java programming language with the tool of Intellij IDEA development. We have also presented the two diagrams of classes modeled by a graphic design software which is Visual Paradigm.

Chapter 6

General conclusion

The main objective of this work was to participate in the development of models and algorithms for the formal verification of heterogeneous systems where the notions of reactivity, heterogeneity, distribution must be considered.These approaches must take temporal properties into consideration.We have applied our approach in the field of critical and heterogeneous real-time systems.This field is still present in computer applications.The contributions of this work can be summarized as follows:

We have proposed an approach for the analysis of critical and heterogeneous real-time systems.The clocks associated with different processes evolve at different frequencies but relative.

We have shown the decidability of reachability using a new abstraction of system behaviors. This abstraction required the redefinition of several necessary concepts.

In heterogeneous and coordinated real-time systems, although the components may be heterogeneous, we have worked on the daTa-R model to interpret the temporal behaviors.

To study the semantics of such systems, we have used the parameter "slope". This parameter allowed us to redefine the equivalence relation on the valuations clocks of an daTA-R.
We have presented that the number of clock regions associated with a daTA-R is finite.Therefore, we redefine the successor relation on these clock regions to be able to build the region automaton.

To have a construction of a region automaton that terminates, we must have a finite index of regions and a successor relation on this index. Indeed, this automaton of regions specifies the same behaviors as those implied by the specification daTA-R.

Bibliography

[1] Aceto, L., Burgueno, A., and Larsen, K. G. (1998). Model checking via reachability testing for timed automata. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 263–280. Springer.

[2] Aceto, L. and Laroussinie, F. (2002). Is your model checker on time? On the complexity of model checking for timed modal logics. *The Journal of Logic and Algebraic Programming*, 52:7–51.

[3] Ahola, J. (2002). Ambient Intelligence: Plenty of Challenges by 2010. In *Advances in Database Technology — EDBT 2002*, pages 14–14. Springer Berlin Heidelberg.

[4] Akshay, S., Bollig, B., Gastin, P., Mukund, M., and Kumar, K. N. (2014). Distributed Timed Automata with Independently Evolving Clocks.

[5] Akshay, S., Bollig, B., Gastin, P., Mukund, M., and Narayan Kumar, K. (2008). Distributed Timed Automata with Independently Evolving Clocks. In *CONCUR 2008-Concurrency Theory*, chapter Distribute, pages 82–97. Springer Berlin Heidelberg.

[6] Alur, R., Courcoubetis, C., and Dill, D. (1993). Model-checking in dense real-time. *Information and computation*, 104(1):2–34.

[7] Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T., Ho, P.-H., Nicollin, X., Olivero, a., Sifakis, J., and Yovine, S. (1995). The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34.

[8] Alur, R. and Dill, D. (1990a). Automata for modeling real-time systems. In *Automata, Languages and Programming: 17th International Colloquium Warwick University, England, July 16–20, 1990 Proceedings 17*, pages 322–335. Springer.

[9] Alur, R. and Dill, D. (1990b). Automata for modeling real-time systems. In *International Colloquium on Automata, Languages, and Programming*, pages 322–335. Springer.

[10] Alur, R. and Dill, D. (1992). The theory of timed automata. In *Real-Time: Theory in Practice: REX Workshop Mook, The Netherlands, June 3–7, 1991 Proceedings*, pages 45–73. Springer.

[11] Alur, R. and Dill, D. L. (1994a). A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235.

[12] Alur, R. and Dill, D. L. (1994b). A theory of timed automata. *Theoretical computer science*, 126(2):183–235.

[13] Alur, R., Feder, T., and Henzinger, T. A. (1996). The benefits of relaxing punctuality. *Journal of the ACM (JACM)*, 43(1):116–146.

[14] Alur, R. and Henzinger, T. (1993). Real-Time Logics: Complexity and Expressiveness. *Information and Computation*, 104(1):35–77.

[15] Alur, R. and Henzinger, T. A. (1991). Logics and models of real time: A survey. In *Workshop/School/Symposium of the REX Project (Research and Education in Concurrent Systems)*, pages 74–106. Springer.

[16] Alur, R. and Henzinger, T. A. (1994). A really temporal logic. *Journal of the ACM (JACM)*, 41(1):181–203.

[17] Avižienis, A., Laprie, J.-C., and Randell, B. (2004). Dependability and its threats: a taxonomy. In *Building the Information Society: IFIP 18th World Computer Congress Topical Sessions 22–27 August 2004 Toulouse, France*, pages 91–120. Springer.

[18] Basagni, S., Chlamtac, I., and Syrotiuk, V. (2000). Location Aware One-to-Many Communication in Mobile Multi-Hop Wireless Networks. In *Proceedings of the 51st IEEE Semiannual Vehicular Technology Conference, VTC 2000 Spring*, volume 1, pages 288–292.

[19] Behrmann, G., David, A., and Larsen, K. G. (2004). A Tutorial on Uppaal. In *Formal Methods for the Design of Real-Time Systems*, volume 3185, pages 200–236.

[20] Belala, N. (2010). *Modèles de temps et leur intérét à la vérification formelle des systèmes temps-réel*. PhD thesis, Université Mentouri de Constantine, Algérie.

[21] Belala, N., Sadouni, D., Boukharrou, R., Chaouche, A.-C., Seraoui, A., and Chachoua, A. (2013). Time petri nets with action duration: a true concurrency real-time model. *International Journal of Embedded and Real-Time Communication Systems (IJERTCS)*, 4(2):62–83.

[22] Bengtsson, J., Griffioen, W. O. D., Kristoffersen, K. J., Larsen, K. G., Larsson, F., Pettersson, P., and Yi, W. (2002). Automated verification of an audio-control protocol using UPPAAL. *The Journal of Logic and Algebraic Programming*, 52:163–181.

[23] Bérard, B. and Dufourd, C. (2000). Timed automata and additive clock constraints. *Information Processing Letters*, 75(1):1–7.

[24] Bérard, B., Petit, A., Diekert, V., and Gastin, P. (1998). Characterization of the expressive power of silent transitions in timed automata. *Fundamenta Informaticae*, 36(2, 3):145–182.

[25] Berthomieu, B. and Diaz, M. (1991). Modeling and verification of time dependent systems using time petri nets. *IEEE transactions on software engineering*, 17(3):259.

[26] Berthomieu, B. and Menasche, M. (1983). An enumerative approach for analyzing time Petri nets. In *Proceedings IFIP 9th World Computer Congress*. Citeseer.

[27] Bouyer, P. (2004). Forward analysis of updatable timed automata. *Formal Methods in System Design*, 24(3):281–320.

[28] Bouyer, P. and Chevalier, F. (2005). On conciseness of extensions of timed automata. *Journal of Automata, Languages and Combinatorics*, 10(4):393–405.

[29] Bouyer, P., Chevalier, F., and Markey, N. (2005). On the expressiveness of TPTL and MTL. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 432–443. Springer.

[30] Bouyer, P., Dufourd, C., Fleury, E., and Petit, A. (2004). Updatable timed automata. *Theoretical Computer Science*, 321(2-3):291–345.

[31] Buyya, R., Garg, S. K., and Calheiros, R. N. (2011). Sla-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions. In *2011 international conference on cloud and service computing*, pages 1–10. IEEE.

[32] Chlamtac, I., Conti, M., and Liu, J. J.-N. (2003). Mobile ad hoc networking: imperatives and challenges. *Ad hoc networks*, 1(1):13–64.

[33] Choi, H. J., Son, D. O., Kang, S. G., Kim, J. M., Lee, H.-H., and Kim, C. H. (2013). An efficient scheduling scheme using estimated execution time for heterogeneous computing systems. *The Journal of Supercomputing*, 65:886–902.

[34] Cimatti, A., Clarke, E., Giunchiglia, F., and Roveri, M. (2000). NuSMV: a new symbolic model checker. *International Journal on Software Tools for Technology Transfer*, 2(4):410–425.

[35] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to algorithms*, volume 6. MIT press Cambridge.

[36] Corson, S. and Macker, J. (1999). Mobile ad hoc networking (manet): Routing protocol performance issues and evaluation considerations. Technical report.

[37] Courcoubetis, C. and Yannakakis, M. (1992). Minimum and maximum delay problems in real-time systems. *Formal Methods in System Design*, 1(4):385–415.

[38] Curtis, P. M. (2011). *Maintaining mission critical systems in a 24/7 environment*, volume 61. John Wiley & Sons.

[39] Dill, D. L. (1989). Timing assumptions and verification of finite-state concurrent systems. In *International Conference on Computer Aided Verification*, pages 197–212. Springer.

[40] Emerson, E. A., Mok, A. K., Sistla, A. P., and Srinivasan, J. (1992). Quantitative temporal reasoning. *Real-Time Systems*, 4(4):331–352.

[41] Ertugrul, N., Soong, W., Dostal, G., and Saxon, D. (2002). Fault tolerant motor drive system with redundancy for critical applications. In *2002 IEEE 33rd Annual IEEE Power Electronics Specialists Conference. Proceedings (Cat. No. 02CH37289)*, volume 3, pages 1457–1462. IEEE.

[42] Fersman, E., Pettersson, P., and Yi, W. (2002). Timed Automata with Asynchronous Processes: Schedulability and Decidability. In *Lecture Notes in Computer Science*, pages 67–82.

[43] Hennessy, M. and Milner, R. (1985). Algebraic laws for nondeterminism and concurrency. *Journal of the ACM (JACM)*, 32(1):137–161.

[44] Henzinger, T. A., Ho, P.-H., and Wong-Toi, H. (1997). HYTECH: a model checker for hybrid systems. *International Journal on Software Tools for Technology Transfer*, 1(1-2):110–122.

[45] Henzinger, T. A., Kopke, P. W., Puri, A., and Varaiya, P. (1998). What's Decidable about Hybrid Automata? *Journal of Computer and System Sciences*, 57(1):94–124.

[46] Henzinger, T. A., Nicollin, X., Sifakis, J., and Yovine, S. (1994a). Symbolic model checking for real-time systems. *Information and computation*, 111(2):193–244.

[47] Henzinger, T. A., Nicollin, X., Sifakis, J., and Yovine, S. (1994b). Symbolic model checking for real-time systems. *Information and computation*, 111(2):193–244.

[48] Kopetz, H. and Steiner, W. (2022). *Real-time systems: design principles for distributed embedded applications*. Springer Nature.

[49] Koymans, R. (1990). Specifying real-time properties with metric temporal logic. *Real-time systems*, 2(4):255–299.

[50] Laroussinie, F., Markey, N., and Schnoebelen, P. (2004). Model checking timed automata with one or two clocks. In *International Conference on Concurrency Theory*, pages 387–401. Springer.

[51] Laroussinie, F., Markey, N., and Schnoebelen, P. (2006). Efficient timed model checking for discrete-time systems. *Theoretical Computer Science*, 353(1):249–271.

[52] Laroussinie, F. and Schnoebelen, P. (2000). The state explosion problem from trace to bisimulation equivalence. In *International Conference on Foundations of Software Science and Computation Structures*, pages 192–207. Springer.

[53] Laroussinie, F., Schnoebelen, P., and Turuani, M. (2003). On the expressivity and complexity of quantitative branching-time temporal logics. *Theoretical Computer Science*, 297(1):297–315.

[54] Larsen, K. G. (1990). Proof systems for satisfiability in Hennessy-Milner logic with recursion. *Theoretical Computer Science*, 72(2):265–288.

[55] Larsen, K. G., Larsson, F., Pettersson, P., and Yi, W. (1997a). Efficient verification of real-time systems: Compact data structure and state-space reduction. In *Proceedings Real-Time Systems Symposium*, pages 14–24. IEEE.

[56] Larsen, K. G., Larsson, F., Pettersson, P., and Yi, W. (1997b). Efficient verification of real-time systems: compact data structure and state-space reduction. In *Real-Time Systems Symposium, 1997. Proceedings., The 18th IEEE*, pages 14–24. IEEE.

[57] Larsen, K. G., Pettersson, P., and Yi, W. (1995). Model-checking for real-time systems. In *International Symposium on Fundamentals of Computation Theory*, pages 62–88. Springer.

[58] Larsen, K. G., Pettersson, P., and Yi, W. (1997c). Uppaal in a nutshell.

[59] Larsen, K. G., Yi, W., and Pearson, J. (1999). Clock Difference Diagrams. *Nordic Journal of Computing-NJC*, 6(December):271—-298.

[60] Layadi, S., Ilie, J.-M., Kitouni, I., and Saidouni, D.-E. (2015). Relative timed model for coordinated multi agent systems. In *Computer Science and Its Applications: 5th IFIP TC 5 International Conference, CIIA 2015, Saida, Algeria, May 20-21, 2015, Proceedings 5*, pages 15–27. Springer.

[61] Lenzen, C., Locher, T., and Wattenhofer, R. (2010). Tight bounds for clock synchronization. *Journal of the ACM (JACM)*, 57(2):1–42.

[62] Liang, C.-Y., Fu, S.-Y., Liu, Y.-P., and Hsu, W.-C. (2018). Automatically migrating sequential applications to heterogeneous system architecture. In *2018 International Conference on High Performance Computing & Simulation (HPCS)*, pages 114–121. IEEE.

[63] Malhotra, G., Goel, S., and Sarangi, S. R. (2014). Gputejas: A parallel simulator for gpu architectures. In *2014 21st International Conference on High Performance Computing (HiPC)*, pages 1–10. IEEE.

[64] Markey, N. and Schnoebelen, P. (2004). Symbolic model checking for simply-timed systems. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 102–117. Springer.

[65] Marwedel, P. (2021). *Embedded system design: embedded systems foundations of cyber-physical systems, and the internet of things*. Springer Nature.

[66] Mell, P., Grance, T., et al. (2011). The nist definition of cloud computing.

[67] Ouaknine, J. and Worrell, J. (2004). On the language inclusion problem for timed automata: Closing a decidability gap. In *Logic in Computer Science, 2004. Proceedings of the 19th Annual IEEE Symposium on*, pages 54–63. IEEE.

[68] Power, J., Hestness, J., Orr, M. S., Hill, M. D., and Wood, D. A. (2014). gem5-gpu: A heterogeneous cpu-gpu simulator. *IEEE Computer Architecture Letters*, 14(1):34–36.

[69] Raskin, J.-F. (1999). *Logics, automata and classical theories for deciding real time*. PhD thesis.

[70] Raskin, J.-F. (2005). An introduction to hybrid automata. In *Handbook of networked and embedded control systems*, pages 491–517. Springer.

[71] Rittinghouse, J., Ransome, J. F., CISM, C., et al. (2011). *Business continuity and disaster recovery for infosec managers*. Elsevier.

[72] Sarkar, S. K., Basavaraju, T., and Puttamadappa, C. (2007). *Ad Hoc Mobile Wireless Networks: Principles, Protocols, and Applications*. CRC Press.

[73] Stankovic, J. A. (1988). Misconceptions about real-time computing: A serious problem for next-generation systems. *Computer*, 21(10):10–19.

[74] Stauffer, C. and Grimson, W. E. L. (1999). Adaptive background mixture models for real-time tracking. In *Proceedings. 1999 IEEE computer society conference on computer vision and pattern recognition (Cat. No PR00149)*, volume 2, pages 246–252. IEEE.

[75] Stirling, C. (2001). *Modal and Temporal Properties of Processes*. Texts in Computer Science. Springer New York, New York, NY.

[76] Tripakis, S. and Yovine, S. (1998). Verification of the fast reservation protocol with delayed transmission using the tool kronos. In *Real-Time Technology and Applications Symposium, 1998. Proceedings. Fourth IEEE*, pages 165–170. IEEE.

[77] Turner, R., Ingold, D., Lane, J. A., Madachy, R., and Anderson, D. (2012). Effectiveness of kanban approaches in systems engineering within rapid response environments. *Procedia Computer Science*, 8:309–314.

[78] Yang, C.-T., Liu, J.-C., Wang, H.-Y., and Hsu, C.-H. (2014). Implementation of gpu virtualization using pci pass-through mechanism. *The Journal of Supercomputing*, 68:183–213.

[79] Yovine, S. (1997a). Kronos: A verification tool for real-time systems. *International Journal on Software Tools for Technology Transfer (STTT)*, 1(1):123–133.

[80] Yovine, S. (1997b). Kronos: A verification tool for real-time systems. *Int. J. Softw. Tools Technol. Transf.*, 1(1-2):123–133.

[81] Zimmerman, T. G. (1996). Personal Area Networks: Near-field intrabody communication. *IBM Systems Journal*, 35(3.4):609–617.