UNIVERSITAT DE BARCELONA

Master Thesis

Master in Pure and Applied Logic

On interval logics and stopwatches in model-checking real-time systems

Marina López Chamosa

Supervisors

Joost J. Joosten

Moritz Müller

Year

2021/22

Abstract

Our thesis focuses on the *model-checking problem*, which is at the heart of both formal verification of software and algorithmic law. In general, this computational problem consists of deciding whether a given structure fulfills a given property expressed by a sentence in a logic[1]. These structures and logics can take many forms.

We speak of *algorithmic law* whenever the application of that particular law is intended to be performed by a computer on a data set representing a real case. In the field of algorithmic law one needs an algorithm to decide whether a particular real case is legal or not. For a model-checking approach, the law is formalized by a sentence in some logic, whereas a case is viewed as a word structure.

In the field of formal verification of software, whose goal is to test whether a program works correctly, the verification task is naturally formalized as a model-checking problem by associating a structure to every program, and a sentence in a suitable logic to every desired property of the program [4].

The model-checking framework often allows to transform a complex and informal question into the formally precise computational problem of whether $K \models \varphi$, where the input K is in some class of structures $\mathcal{K}$ and the input φ is in some language L. As a result, it is of practical interest in many real-world applications, providing both simple procedures and mathematical proofs of correctness. Thus, the computational complexity of the mentioned problem is of central importance.

In our thesis, we discuss different formalisms as inputs of the model-checking problem to analyze their complexity. In particular, the model-checking problem of linear-temporal properties is studied, both in the presence of discrete and continuous time, with an automata-theoretic approach. The strategy in this setting is to reduce questions about models and sentences, to questions about automata, and then provide an answer using standard decision procedures for automata.

[1] In our setting we view a *logic* abstractly as given by sets of *sentences* and a *satisfaction relation* between structures and sentences. For example, sentences could be certain automata, the structures certain words or timed words, and the satisfaction relation given by automaton acceptance. Thereby, we follow the tradition of abstract model-theory and refer to [11] for a formal set-up. In this thesis we use the term "logic" only informally and refrain from giving a definition.

We start by introducing a conceptual framework that is used throughout the thesis. In particular, we define the Metric Interval Temporal Logic (MITL) and its intepretation over real-time words called *signals*. Next, we give a presentation of the translation from MITL formulas to a variant of timed transducers from [14], where we elaborate many claims and sketches that occur in the paper without further detail. This recent construction from Ferrere et al. stands out for its simplicity and elegance, as opposed to cumbersome constructions based on alternating automata or tableaux [15, 17]. Further, an study of the complexity of the translation is offered, and a complexity analysis of MC(TA, MITL) for Timed Automata (TA) over MITL sentences is provided.

The desire for even more expressive logics leads us to the class of Stopwatch Automata (SWA), an extension of Timed Automata. Although the expressive capacity of SWA is very appealing in the context of reasoning about durations, the emptiness problem for SWA was unfortunately shown to be undecidable in [18]. In [18], Henzinger et al. designed a very abstract setting, yielding general results that provided little insight in the particular case of SWA. In the final chapter we revisit [18] providing a simpler undecidability proof.

Acknowledgements

I would like to thank many people for supporting me throughout this thesis. First of all, I am deeply grateful to my supervisors. I want to thank Joost for his experienced advice and encouragement; and Moritz for his patience, involvement and ideas. I have learnt so much from you.

I'd like to thank Catalina as well for becoming my first friend when I came to Barcelona and being by my side until today. Also, I thank my friend Mireia for always being willing to listen and giving the best advice, and Albert because he is the best study and adventure partner and a constant source of inspiration.

Finally, I am grateful to my family for their support and love: without you this thesis wouldn't exist. Especially, I want to thank my mother for her help and all the effort she puts into understanding and appreciating everything I do.

Contents

Introduction

An algorithm is a set of precise instructions designed to perform a specific task. Similar to a cooking recipe, algorithms are the basis of every executable code. From smartphones and airplanes to medical devices and power plants, all types of devices rely heavily on algorithms to perform automated processes that allow them to function. Public administrations are increasingly using algorithmic systems to speed up their decision-making processes and (hopefully) minimize the risk of human errors. What taxes a person must pay, who is eligible for financial aid and social services, and which school a child will attend are all questions often answered by an algorithm with a level of efficiency that exceeds human capabilities.

However, there is a catch: algorithms are intricate and, like any human creation typically prone to contain some errors. Even minor programming errors can lead to unexpected outcomes and systematic failures [9]. Due to the the large presence of software in our lives, the role of software verification has become critical in error prevention and quality assurance. The so-called *formal verification techniques*, which aim to provide mathematical proofs of software correctness, are among the most reliable and recognized techniques for software verification. Its most prominent exponent, the *model-checking* technique, was introduced in the 1980s as a novel framework in the formal verification of concurrent systems by Clarke and Emerson [7] and, independently, by Queille and Sifakis [25].

The challenge of these computer scientists was to find simple yet rigorous methods of testing software correctness. To face this challenge, they established a new theoretical framework where they conceived programs as finite structures from a class of structures $\mathcal{K}$, representing their possible computations, and desirable properties (or *specifications*) as sentences in a formal language L. The final ingredient was a model-checker, an algorithm decision whether the structure was a model of the sentence or not, through systematic inspection of all states of the model.

In the field of law, the use of algorithms plays a fundamental role in so-called *algorithmic laws*, whose application to a given case is executed by an algorithm. The algorithm takes as input both a case and a law, and decides whether the given case is legal according to the given law or not. To be presented as inputs of an algorithm, both the case and the law have to be suitably formalised. As before, we formalize a set of cases by a class of finite structures $\mathcal{K}$ and a law is

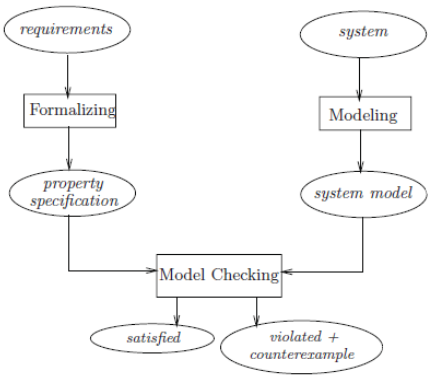


Figure 0.0.1: Schematic view of model-checking in software verification [4].

translated into some formal language L.

Therefore, the model-checking approach to software verification and the problem of algorithmic law, can be mathematically formalised into the same class of computational problems, denoted by $MC(\mathcal{K}, L)$:

Input: $K \in \mathcal{K}$, $\varphi \in L$

Problem: $K \vDash \varphi$?

The automata-theoretic approach

The study of the correspondence between logics and classes of automata started in the 60s with the work of Büchi [6], Elgot [12], McNaughton [23] and Rabin [26], among others. Motivated by decision problems in mathematical logic, they investigated relations between variants of second-order logic and automata over infinite objects. In one of his most famous theorems, Büchi proved that a fragment of second-order logic is expressively equivalent to finite automata, by defining, for every formula φ, an automata $\mathbb{A}_\varphi$ that accepted exactly all the models of φ. With this result, he paved the way to solving many decision problems in mathematics and logic.

Over the last decades, establishing relations between logics and classes of automata to solve computational problems has proven to be a fruitful strategy. In the context of software verification, the behaviour of a program can be represented by the set of all its computations. Moreover, computations are sequences of states, whence they can be formalised as words over the alphabet of propositions holding at each state. In this view, a program can be seen as a formal language. Similarly, a specification is represented by a formula in some logic, inducing the set of words that are models of the formula. We obtain that formal languages are mathematical models of both programs and specifications.

Within this framework, the standard model-checking technique relies on a translation from every formula φ in a suitable logic to an automaton $\mathbb{A}_\varphi$ accepting precisely the words that satisfy the formula [31]. To test a certain property

φ for a given system S, it is enough to compare the language of $\mathbb{A}_{\neg\varphi}$ to the set of behaviours of the system, represented by $\mathbb{A}_S$. If any word in $L(\mathbb{A}_S)$ happens to be accepted by $\mathbb{A}_{\neg\varphi}$, then the system doesn't meet the specification and such a word is a counterexample.

The above perspective allows to reduce the model-checking problem between a system and a specification to a decision problem about automata. This approach can be applied in the context of algorithmic law too as we explain below, with the help of an example.

The European transport Regulation 561 [13] limits driving times of european truck drivers and imposes resting periods of various durations. The activities of a driver are recorded by tachographs. Regulation 561 distinguishes three activities that a driver can perform: *driving*, *resting* and *other work*. The time that a driver can spend doing the same activity is regulated by a number of complex quantitative rules, such as Article 6 (see Figure 0.0.2).

Because the size of the data set given by a tachograph can easily become untractable to humans, Regulation 561 is a clear example of a law that needs the help of an algorithm to be applied.

Article 6

1. The daily driving time shall not exceed nine hours.

However, the daily driving time may be extended to at most 10 hours not more than twice during the week.

2. The weekly driving time shall not exceed 56 hours and shall not result in the maximum weekly working time laid down in Directive 2002/15/EC being exceeded.

3. The total accumulated driving time during any two consecutive weeks shall not exceed 90 hours.

Figure 0.0.2: Example of algorithmic law from Regulation 561 [13].

If one wishes to check the compliance of a given article from Regulation 561 by a particular driver, one needs to formalize both the set of activities performed by the driver, and the article. The first is naturally formalized as a sequence of propositions from the set $P = \{\text{drive}, \text{rest}, \text{other work}\}$ i.e. a finite word over the alphabet $\Sigma = \{d, r, w\}$. For example the word $ddddrrw$ describes 4 minutes of driving, followed by 2 minutes of resting and then 1 minute of doing other work.

A given article is typically formalised into a sentence in some logic L over P. In general, we choose L depending on the particular features of the law to be expressive enough while remaining computably tractable. Furthermore, every sentence in the language of L should have a straightforward interpretation over words as above. Therefore every sentence from the law can be viewed as a set

of allowed words, i.e. a language over P.

Then, testing the legality of a case with respect to a law is equivalent to checking whether the word w formalizing the case belongs to the language of the automaton $\mathbb{A}_\varphi$ where $\varphi \in L$ formalizes the law.

Model-checking real-time systems

The problem of model-checking linear-time properties in the presence of discrete time has been widely studied [22, 4, 5, 7, 8]. In the discrete case, $\mathbb{N}$ is commonly chosen as time domain and ω-words as semantic structures. To express properties of these structures, logics such as Linear Temporal Logic (LTL hereafter) and Computation Tree Logic (CTL) are employed and a variety of efficient translations from LTL formulas to finite automata have been proposed [15, 16, 28, 19].

However, the situation in the case where $\mathbb{R}_0^+$ is chosen as time domain is rather different [3, 21, 29]. Many variants of real-time logics and timed automata have been studied but correspondences among them are not so simple and natural as in the discrete case.

We focus on one of the most prominent real-time extensions of LTL, Metric Interval Temporal Logic (MITL) [2], which arises as a syntactic restriction of Metric Temporal Logic (MTL) [20]. In MTL, real time is introduced in the syntax of a linear temporal logic by replacing the unrestricted temporal operators by time-bounded versions. For example, the bounded operator $\Diamond_{[2,4]}$ is interpreted as "eventually, within 2 to 4 time units". To obtain MITL from MTL, one avoids singular intervals of the form $[a, a]$ in the subscripts[2].

The aim of this thesis is to give a comprehensive presentation of [14], whose main contribution concerns an efficient translation from MITL to a variant of timed automata. To this end, we rewrite formulas using a restricted language, into the so-called *standard form*. Then we explicitly construct four basic automata that can be combined using what we call *parallel-composition* and *sequential-composition*, yielding temporal testers for any MITL formula.

This construction is they key to show that the model-checking problem of MITL formulas against Timed Automata is decidable, by adapting the proof from [1] for signal-based automata.

The Regulation 561 [13] is then used to illustrate the difficulties of MITL when reasoning about accumulated durations of alternating activies. This brings us to the class of Stopwatch Automata [10], a variant of timed automata whose clocks can be running or paused at states. This enables overall durations to be measured by allowing a clock to stop when the activity is paused and then reactivate when the activity is resumed. Unfortunately, even automata with a single stopwatch have undecidable emptiness, a fact which was first shown by Henzinger et al. in [18]. The last part of our thesis is devoted to an adaptation of the proof from [18].

[2] Although this restriction was intented to guarantee decidability, it was discovered that MTL can be decided over finitary event-based semantics [24].

Original contributions

In this section we would like to emphasize the original contributions that this master thesis contains.

As a contribution to the work presented in [14], we extend the translation for MITL over bounded semantics, a context where we are able to explain the underlying relation between past and future modalities as presented in Proposition 2.2.5, obtaining a simple correspondence between past and future temporal testers. The relation among past and future temporal operators ($\mathcal{U}$ vs $\mathcal{S}$, $\bigcirc$ vs $\ominus$ and so on) has to do with a general notion of *reversability* from automata theory, introduced in Section 1.3 from Chapter 1.

Many results from [14] that were simply stated or had sketchy proofs, are proven with detail in Chapter 3, including:

- Interpretability of MITL over signals in Theorem 3.1.1
- Theorem 3.2.2 about the complexity cost of rewriting into standard form
- Correctness of the parallel composition in Proposition 3.4.1
- Correctness of basic temporal testers in Theorems 3.5.2, 3.5.6, 3.5.9

From slight corrections of the paper [14], we obtain completely new theorems, such as:

- Rewriting of temporal formulas in Lemmas 3.2.5, 3.2.7, 3.2.6, 3.2.8, obtaining new compact expressions in Lemmas 3.2.9 and 3.2.10.
- New definition of sequential composition in Definition 3.4.3 and a proof of its correctness in Proposition 3.4.2, essential for the construction of temporal testers, which is the main contribution of the paper.
- Reformulation of a characterization of all basic MITL operators $\mathcal{U}, \mathcal{S}, \Diamond_{(0,a)}, \blacklozenge_{(0,a)}$ in Theorems 3.5.1,3.5.4, 3.5.8,3.5.11

In Chapter 4, we provide a model-checker for MITL against timed automata inspired in Alur and Dill's construction from [1], which we adapt for signal-based timed automata. The model-checking algorithm and a time estimation of its complexity are discussed, something that was again simply outlined in [14].

Finally, in Chapter 5, we revisit a paper by Henzinger et al. [18] about decision problems for a general class of automata which is very abstract and difficult to read. We prove in a clear and straightforward way what in the paper is a particular case of a sketchy proof and do this by adapting the proof to the specific setting of Stopwatch Automata.

Chapter 1

Preliminaries

When facing a model-checking problem, one is provided a *system* (program, real situation) and a *specification*, in order to check whether the specification is met by the system. Before looking for efficient algorithms that give an answer to this question, one has to establish a mathematical framework where a model of the problem and the elements involved can be defined. In this thesis we use the standard automata-theoretic approach, that stands out for offering clarity and simpler decision processes. Based on concepts from automata theory and logic, its main strategy consists on reducing questions about systems and specifications, to questions about automata.

From a logical perspective, each state of a system (resp. a real case) can be associated to the set of propositions that hold in that state. For example, the following sets of propositions

$$P_1 = \{\text{green}, \text{yellow}, \text{red}\}$$
$$P_2 = \{\text{start}, \text{close}, \text{heat}, \text{stop}\}$$

allow to describe the state of a traffic light and a microwave. Therefore, each behaviour of a system over a set of propositions P induces a word over the alphabet $\mathcal{P}(P)$, and the set of all possible behaviours of the system induces a language over the same alphabet.

A system specification (resp. a law) is typically formalised as a sentence in some logic over P. Furthermore, every system specification can be viewed as a set of allowed computations, that is, a language over $\mathcal{P}(P)$.

Therefore we can conceive systems and specifications as formal languages, and sometimes represent them by finite automata. Within this framework, the stardard strategy is to reduce questions about systems and specifications, to questions about automata, such as

- is the language of an automaton empty?
- is the language of an automaton contained in the language of another automaton?

We devote this chapter to the basic concepts that allow to use all the tools from automata theory. First, we explain what we mean by formal languages and how some of them can be represented by finite automata. Then, we give a definition of finite automata and divide them into groups according to different features. To continue, we present timed automata as extensions of finite automata equipped with a finite set of real-valued clocks. Lastly, a natural way to reverse words is shown and the Reverse Theorem 1.3.1 for finite automata is exposed.

1.1 Words and formal languages

An *alphabet* Σ is a finite set of symbols (also called *letters*), and a *finite word* over Σ is a finite sequence of symbols $w = a_0a_1 \dots a_{n-1}$ from Σ. Given a word $w = a_0a_1 \dots a_{n-1}$ we say that w has length $|w| = n$. We denote by Σ^n the set of words over Σ of length n and by $\Sigma^* = \bigcup_{n\in\mathbb{N}} \Sigma^n$ the set of all finite words over Σ. The empty word is denoted by ε and $\Sigma^+ = \Sigma^* \backslash \{\varepsilon\}$. A *formal language* L is any subset of Σ^*.

Example. Let $\Sigma = \{a, b\}$. Then $\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab \dots\}$. The set

$$\{a, ab, bab\}$$

is a language over Σ. Because it contains a finite set of words, we call it a finite language. The set

$$L = \{ab^n : n \geq 0\}$$

is also a language over Σ, an infinite language. Words such as a or $abbbb$ belong to L, while aba doesn't.

In the context of formal verification, each letter of the alphabet will be used to describe the state of a system at a given time, and the sequence of letters shows the evolution of the system as the time is incremented.

Every finite word $w = a_0a_1 \dots a_{n-1}$ can be viewed as a map $w : [n] \to \Sigma$ where $[n] := \{0, \dots, n-1\}$ and $w(i) = a_i$ for $i \in [n]$. We sometimes call $[n]$-words those finite words of length n. We also define an *ω-word* as an infinite countable sequence of symbols from Σ. We denote by Σ^ω the set of all infinite words and a subset L of Σ^ω is called an ω-language. Similarly, we can view ω-words as maps $w : \omega \to \Sigma$.

In the following section we study how to represent languages and ω-languages by finite automata and ω-automata.

1.2 Finite automata

An automaton is a mathematical model of a machine that perfoms computations on an input by moving through a series of configurations. Mathematically, automata are very similar to a graph, consisting of a set of states and transitions. However, they are provided with some additional elements that make them a

useful tool in a wide range of logical and computational problems. We classify automata according to their *purpose*, into

- acceptors (either accept the input or not),
- transducers (generate output from given input),
- and combinations of them,

according to their *transition relation*, into

- deterministic (one transition for each possible input),
- non-deterministic (one, none, or more than one transition for each possible input),

and according to the class of words they may take as input, into

- finite automata (reading finite words),
- ω-automata (reading ω-words),
- timed automata (reading variants of $\mathbb{R}_0^+$-words).

1.2.1 Acceptors

Here we present a kind of finite automata whose most significant feature is being able to read finite words and then accepting or rejecting them.

Definition 1.2.1. A *deterministic finite automaton DFA* is a tuple $\mathbb{A} = (S, s_0, \Sigma, \Delta, F)$ where

- S is a finite set of *states*,
- s_0 is the *initial* state,
- Σ is a nonempty *alphabet*,
- $\Delta : S \times \Sigma \to S$ is the *transition* function,
- $F \subseteq S$ is a set of *final* states.

A DFA takes as input a finite word $w \in \Sigma^*$. From the initial state, it moves thoroughout w by reading a letter at each step, and moving to different states by means of transition. A transition from state s to state s' after reading letter a is denoted by $s \xrightarrow{a} s'$. Formally, a run of the automaton is defined as follows.

Definition 1.2.2. A *run* of a DFA $\mathbb{A}$ over a word $a_0a_1 \dots a_{n-1}$ is an alternating sequence of states and letters $s_0a_0s_1 \cdots a_{n-1}s_n$ such that $\Delta(s_i, a_i) = s_{i+1}$ for all $0 \leq i \leq n-1$. The automaton $\mathbb{A}$ *accepts* a word $a_0a_1 \dots a_{n-1}$ if the (unique) run $s_0a_0s_1 \cdots a_{n-1}s_n$ ends in a final state, this is, $s_n \in F$.

Every DFA is an acceptor and therefore induces a set of words: the set of words accepted by the automaton. For every DFA $\mathbb{A}$, the set of words accepted by $\mathbb{A}$ is called the *language* of $\mathbb{A}$ and it is denoted by $L(\mathbb{A})$. The class of languages that are accepted by DFAs are the so-called *regular languages*.

Observe that DFAs are deterministic automata, meaning that for every $(s, a) \in S \times \Sigma$, there is a unique s' such that $(s, a, s') \in \Delta$. Automata that don't satisfy this property are called non-deterministic.

Definition 1.2.3. A *non-deterministic finite automata NFA* is a tuple $\mathbb{A} = (S, S_0, \Sigma, \Delta, F)$ where $S_0 \subseteq S$ is a set of initial states, $\Delta \subseteq S \times \Sigma \times S$ is the transition relation, and the rest is defined as before.

In the case of non-deterministic automata, the same word w can lead to different runs, obtaining the following definition of acceptance.

Definition 1.2.4. A *run* of a NFA $\mathbb{A}$ over a word $a_0a_1 \cdots a_{n-1}$ is an alternating sequence of states and letters $s_0a_0s_1a_1 \cdots a_{n-1}s_n$ such that $s_0 \in S_0$ and $(s_i, a_i, s_{i+1}) \in \Delta$ for all $0 \leq i \leq n-1$. The automaton $\mathbb{A}$ *accepts* a given input word $w = a_0a_1 \dots a_{n-1}$ iff there is an execution $s_0a_0s_1 \cdots a_{n-1}s_n$ such that $s_n \in F$.

We say that two automata $\mathbb{A}, \mathbb{A}'$ are equivalent if and only if $L(\mathbb{A}) = L(\mathbb{A}')$.

Theorem 1.2.1. *For every NFA $\mathbb{A}$ with k states there is an equivalent DFA $\mathbb{B}$.*

Proof. Given an NFA $\mathbb{A} = (S, S_0, \Sigma, \Delta, F)$, let $\mathbb{A}' = (S, s_0, \Sigma, \Delta, F)$ be the equivalent automaton to $\mathbb{A}$ with a single start state. Then define $\mathbb{B} = (S', s_0', \Sigma, \Delta', F')$ where

- $S' := \mathcal{P}(S)$
- $s_0' := \{s_0\}$
- Δ' consists of elements (X, a, Y) with $X \subseteq S$, $a \in \Sigma$ and $Y = \{s' \in S : (s, a, s') \in \Delta, s \in X\}$
- $F' := \{X \subseteq S : X \cap F \neq \emptyset\}$

Then $\mathbb{B}$ is a DFA with $L(\mathbb{B}) = L(\mathbb{A})$. □

We will be interested in representing sets of ω-words as well, thereby we define a new class of automata, the so-called ω-automata. In this context, the notion of final state makes no sense, and is replaced by a different notion of acceptance.

In general, a non-deterministic finite ω-automaton is a tuple $\mathbb{A} = (S, S_0, \Sigma, \Delta, F)$ where every component has the usual meaning and F is called the *acceptance condition*. For every interpretation of F, one obtains a different kind of ω-automata. One of the most popular ω-automata are the so-called Büchi automata, whose accepting condition requires that some final state occurs infinitely often in the run. The class of languages accepted by these automata are the so-called *ω-regular languages*.

Definition 1.2.5. A *non-deterministic Büchi-automaton NBA* is a tuple $\mathbb{A} = (S, S_0, \Sigma, \Delta, F)$ where every component has the usual meaning. A *run* of the automaton is an infinite alternating sequence $s_0 a_0 s_1 a_1 \cdots$ such that $(s_i, a_i, s_{i+1}) \in \Delta$ for every $i \in \mathbb{N}$, and $s_0 \in S_0$. The automaton $\mathbb{A}$ *accepts* a given input word $w = a_0 a_1 a_2 \cdots \in \Sigma^\omega$ iff there is a run $s_0 a_0 s_1 a_1 \cdots$ such that $s_i \in F$ for infinitely many $i \in \mathbb{N}$.

We can extend this notion into a more general class of ω-automata.

Definition 1.2.6. A *generalized non-deterministic Büchi automaton GNBA* is a tuple $\mathbb{A} = (S, S_0, \Sigma, \Delta, \mathcal{F})$ where $\mathcal{F} \subseteq \mathcal{P}(S)$ is the *generalized Büchi condition.* The automaton $\mathbb{A}$ *accepts* a given input word $w = a_0 a_1 a_2 \cdots \in \Sigma^\omega$ iff there is a run $s_0 a_0 a_1 s_1 \ldots$ such that *for all* $F \in \mathcal{F}$ there are infinitely many $i \in \mathbb{N}$ with $s_i \in F$.

Theorem 1.2.2. *For every GBA $\mathbb{A}$ with k states and n sets of accepting states there is an equivalent BA $\mathbb{A}$ with $k \cdot n$ states.*

1.2.2 Timed automata

In this section we discuss different formalisms that have been proposed to model the behaviour of real-time systems. These formalisms typically arise as $\mathbb{R}_0^+$-words over the set of states of the system, to which we apply some restrictions. Also, we explain how to broaden the definition of finite automata using clocks so that they accept these words.

In particular, we can describe these models as countable sequences of the form

$$(a_0, I_0)(a_1, I_1)(a_2, I_2) \cdots$$

where w forms the word of domain $\mathbb{R}_0^+$ constantly taking value a_i throughout I_i, and $I_0 I_1 I_2 \cdots$ is an *interval sequence* [2] as explained below.

Definition 1.2.7. An *interval sequence* $\mathcal{I} = I_0 I_1 I_2 \cdots$ is a countable (finite or infinite) sequence of adjacent intervals that partitions the time domain $\mathbb{R}_0^+$. In other words,

- $\{t : t \in I_i \text{ for some } i\} = \mathbb{R}_0^+$, and
- for all i, the right end-point of I_i and the left end-point of I_{i+1} coincide and belongs to exactly one of them.

The following sequences are examples of interval sequences

$$[0,1)[1,2)[2,3)[3,4) \cdots$$
$$[0,102)[102,102](102,+\infty)$$
$$[0,0](0,0.5](0.5,1)[1,1.5] \cdots$$

Definition 1.2.8. We say that an interval partition $\mathcal{I} = \{I_i\}_{i \in \mathbb{N}}$ and a word $w : \mathbb{R}_0^+ \to \Sigma$ are *compatible* if w is constant over every I_i, for all $i \geq 0$.

Therefore the models we are interested in are those words w of domain $\mathbb{R}_0^+$ for which there exists some compatible interval sequence $\mathcal{I}$. In the literature, we find two main classes of models $(w, \mathcal{I})$ by applying restrictions to $\mathcal{I}$, namely, *timed words*, and *signals*.

Timed words

In [1], Dill introduces the so-called *timed words* by associating a real-valued time with each symbol in a word. Also, they define the corresponding class of *timed automata* accepting timed words, which can be employed to develop a theory of timed languages.

Definition 1.2.9. A *timed word* is pair $(w, \mathcal{I})$ where w is a word of domain $\mathbb{R}_0^+$ and $\mathcal{I} = I_0 I_1 I_2 \cdots$ is an *infinite* interval sequence such that I_i is left-closed and right-open, for all $i \geq 0$.

Given a timed word w we define $\mathcal{I}_w$ as the sequence $[0, d_1)[d_1, d_2)[d_2, d_3) \cdots$ where every d_i is a discontinuity of w with respect to the usual topology in $\mathbb{R}_0^+$ and the discrete topology in Σ. It is clear that $\mathcal{I}_w$ is compatible with w for every timed word w.

Definition 1.2.10. A *timed language* L_t over Σ is a set of timed words over Σ.

A simpler representation of timed words is shown in [1] using *time sequences*, infinite increasing sequences $\tau = \tau_1 \tau_2 \tau_3 \ldots$ of time values $\tau_i \in \mathbb{R}$ with $\tau_i > 0$, satisfying that $\tau_i < \tau_{i+1}$ for all $i \geq 1$, and that for every $t \in \mathbb{R}_0^+$, there is some $i \geq 1$ such that $\tau_i > t$. Let us observe that for every time sequence $\tau = \tau_1 \tau_2 \tau_3 \ldots$, $\mathcal{I} = [0, \tau_1)[\tau_1, \tau_2)[\tau_2, \tau_3) \cdots$ is an interval sequence from Definition 1.2.9, and viceversa. Thus, every timed word over Σ can be represented as an infinite countable sequence

$$(a_0, 0)(a_1, \tau_1)(a_2, \tau_2)(a_3, \tau_3) \cdots$$

where $a_i \in \Sigma$ and $\tau = \tau_1 \tau_2 \tau_3 \cdots$ is a time sequence.

To define timed automata recognizing timed words we shall introduce a few concepts. Let $C = \{x_1, \ldots, x_n\}$ denote a finite set of *clock variables*, intended to range over $\mathbb{R}_0^+$. These variables measure the time between transitions in the automaton and evolve all at the same rate. Furthermore, they can be reset along some transitions, and used as guards along other transitions or invariants to be preserved while letting time elapse in locations of the automaton.

Definition 1.2.11. For a set C of clock variables, a *clock constraint* $\phi(C)$ is a set with elements of the form $x < k, x \leq k, x = k, x \geq k$ and $x > k$ where $k \in \mathbb{Q}_0^+, x \in C$. We denote by $\Phi(C)$ the set of all clock constraints over C.

At every time instant, each clock in C has a particular value given by a valuation $v : C \to \mathbb{R}_0^+$. We say that a clock interpretation v for C satisfies a clock constraint $\phi(C)$ iff $\phi(C)$ evaluates to true using the values given by v. For every $t \geq 0$, we denote by $v + t$ the valuation mapping every clock x to the value

$v(x) + t$. For a reset condition $R \subseteq C$, we denote $R[v]$ the valuation obtained by setting to zero every component v_i of v such that $x_i \in R$.

Now we present a description of automata recognizing timed words.

Definition 1.2.12. A *non-deterministic timed automaton TA* $\mathbb{A}$ is a tuple $(S, S_0, \Sigma, C, \Delta, Inv)$ where S is a set of states and $S_0 \subseteq S$, C is a finite set of clock variables, $Inv : S \cup S_0 \to \Phi(C)$ is a map assigning an invariance condition to every state, and $\Delta \subseteq (S \cup S_0) \times \Phi(C) \times \Sigma \times \mathcal{P}(C) \times S$ is the transition relation, whose elements are of the form (s, g, a, R, s'), where

- $s \in S \cup S_0$ and $s' \in S$ are states,
- $g \in \Phi(C)$ is a clock constraint over C,
- $a \in \Sigma$ is a letter,
- $R \subseteq C$ is a set of clock variables that are reset after the transition.

A *configuration* of a timed automaton is determined by a pair (s, v) where s is a state and v is a clock valuation. Informally, given a timed word (w, τ) the timed automaton $\mathbb{A}$ starts in one of its start states at time 0 with all its clocks initialized to 0. As time advances, the values of all clocks change, reflecting the elapsed time, as long as their values satisfy the invariance condition associated with the current state. At time τ_i, $\mathbb{A}$ moves from s to s' using some transition of the form (s, g, a_i, R, s') reading the input a_i, if the current values of clocks satisfy g. With this transition the clocks in R are reset to 0, and thus start counting time with respect to the time of occurrence of this transition.

Definition 1.2.13. A *run* of a timed automaton $(S, S_0, \Sigma, C, \Delta, Inv)$ over a timed word (w, τ) is an infinite sequence of configurations and time transitions

$$r : (s_0, v_0) \xrightarrow[\tau_1]{a_1} (s_1, v_1) \xrightarrow[\tau_2]{a_2} (s_2, v_2) \xrightarrow[\tau_3]{a_3} (s_3, v_3) \cdots$$

where

- s_i are states and s_0 is initial,
- v_0 is constantly 0,
- for all $i \geq 1$ there is a transition of the form $(s_{i-1}, g_i, a_i, R_i, s_i)$ such that
  - $(v_{i-1} + \tau_i - \tau_{i-1})$ satisfies the guard g_i,
  - $v_i = R_i[v_{i-1} + \tau_i - \tau_{i-1}]$,
  - for all $t \in (0, \tau_i - \tau_{i-1})$, $v_i + t$ satisfies $Inv(s_{i-1})$.

Signals

In this section we introduce a kind of interval sequences wich alternate closed unitary intervals and open intervals.

Definition 1.2.14. A *point-segment sequence* is an interval sequence $\mathcal{I} = I_0 I_1 I_2 \cdots$ such that

1. $I_0 = [0, 0]$,
2. $I_{2n} = [t, t]$ and $I_{2n+1} = (t', t'')$ for $n \in \mathbb{N}$, where $t' < t''$ and t'' is allowed to be $+\infty$.

Thus, every point-segment sequence $\mathcal{I}$ can be represented as

$$\{0\}(0, t_0)\{t_0\}(t_0, t_1)\ldots$$

Definition 1.2.15. A *signal* is a word of domain $\mathbb{R}_0^+$ that admits a compatible point-segment sequence $\mathcal{I}$.

Thus, given a signal w and a compatible point-segement sequence $\mathcal{I} = \{0\}(0, t_0)\{t_0\}(t_0, t_1)\{t_1\}\ldots$, there is a unique decomposition of w of the form

$$\dot{a_0} \cdot a_0^{r_0} \cdot \dot{a_1} \cdot a_1^{r_1} \cdot \dot{a_2} \cdots$$

where $\dot{a_i} = w(t_i)$, $a_i = w(t)$ for any $t \in (t_i, t_{i+1})$, and $r_i = t_{i+1} - t_i$, for every $i \geq 0$.

Given a signal w we define $\mathcal{I}_w$ as the point-segement sequence $\{0\}(0, t_1)\{t_1\}$ $(t_1, t_2)\{t_2\}\cdots$ where every t_i is a discontinuity of w. It is clear that $\mathcal{I}_w$ is compatible with w for every signal w.

Definition 1.2.16. A *signal language* L_t over Σ is a set of signals over Σ.

Definition 1.2.17. A *non-deterministic signal-based timed automaton TA* $\mathbb{A}$ is a tuple $(S, \underline{s}, \Sigma, C, \Delta, Inv)$ where S is a set of states and $\underline{s} \notin S$ is the initial state, C is a finite set of clock variables, $Inv : S \to \Phi(C)$ is a map assigning an invariance condition to every state, and $\Delta \subseteq S \cup \{\underline{s}\} \times \phi(C) \times \Sigma \times \mathcal{P}(C) \times S$ is the transition relation, whose elements are of the form (s, g, a, R, s'), where

- $s \in S \cup \{\underline{s}\}$ and $s' \in S$ are states,
- $g \in \Phi(C)$ is a clock constraint over C,
- $a \in \Sigma$ is an input letter,
- $R \subseteq C$ is a set of clock variables that are reset after the transition.

The configuration of the timed automaton is again given by a pair (s, v). Informally, a run of the timed automaton is an alternating sequence of time and discrete transitions. The transducer can either stay in a state for some time, provided that the invariance condition holds, or take a transition to a different state whose guard is satisfied.

Given a signal w, the two kinds of transitions are defined as follows:

- A *time transition* is of the form $(s, v) \xrightarrow{a^r} (s, v + r)$ where
  - there is some $t_a \geq 0$ such that $w(t) = a$ for every $t \in (t_a, t_a + r)$,
  - $v + t$ satisfies $Inv(s)$ for all $t \in (0, r)$.
- A *discrete transition* is of the form $(s, v) \xrightarrow[\delta]{\dot{a}} (s', v')$ where
  - $\delta = (s, g, R, s') \in \Delta$,
  - $v' = R[v]$,
  - there is some $t_a \geq 0$ such that $w(t_a) = \dot{a}$,
  - v satisfies the guard g.

Definition 1.2.18. A *run* of the automaton starting at configuration $(\underline{s}, v)$ for an arbitrary clock valuation v over the input signal w decomposed as $\dot{a_0} \cdot a_0^{r_0} \cdot a_1 \cdot a_1^{r_1} \cdots$ is a finite or infinite sequence of the form

$$(\underline{s}, v) \xrightarrow[\delta_0]{\dot{a}_0} (s_0, v_0) \xrightarrow{a_0^{r_0}} (s_0, v_0 + r_0) \xrightarrow[\delta_1]{\dot{a}_1} (s_1, v_1) \xrightarrow{a_1^{r_1}} (s_1, v_1 + r_1) \ldots,$$

Proposition 1.2.1. *Every timed word is a signal.*

Proof. Let w be a timed word and $\mathcal{I} = [0, \tau_1)[\tau_1, \tau_2)[\tau_2, \tau_3) \cdots$ be a witness of it. Then $[0, 0](0, \tau_1)[\tau_1, \tau_1](\tau_1, \tau_2)[\tau_2, \tau_2](\tau_2, \tau_3) \cdots$ is a point-segment sequence compatible with w. □

1.2.3 Transducers

So far we have focused on representing sets of words. In addition to that, automata can also be used to represent *relations* between sets of words, and in particular, functions. These automata are the so-called transducers. They are endowed with an output alphabet O and a relation $\lambda \subseteq S \times \Sigma \times O \times S$, whose elements are of the form (s, a, b, s'). A transition where the automaton is in state s, reads symbol a, outputs symbol b, and goes to state s' is represented by the expression $s \xrightarrow{a/b} s'$.

Definition 1.2.19. A *non-deterministic finite transducer NFT* $\mathcal{T}$ is a tuple $(S, S_0, \Sigma, O, \Delta)$ where the usual symbols have the usual meaning, O is a finite set called the output alphabet and $\Delta \subseteq S \times \Sigma \times O \times S$ is the transition relation consisting of elements $\delta = (s, a, b, s')$ where a, b are the input and output labels of the edge δ. A *run* of the transducer over the input word $w = a_0 a_1 \cdots a_{n-1} \in \Sigma^*$ is a finite sequence $s_0 a_0 b_0 s_1 \cdots a_{n-1} b_{n-1} s_n$ such that $(s_i, a_i, b_i, s_{i+1}) \in \Delta$ for every $0 \leq i \leq n - 1$.

At each step, a transducer reads an input letter a from a word $a_0 a_1 \cdots a_{n-1}$ and synchronously outputs a letter b from the output alphabet. At the end of the run, we obtain an output word $b_0 b_1 \cdots b_{n-1}$ by looking at all the edges of the run. In general, one input word can give rise to none, one, or more than one output word.

Definition 1.2.20. Given an input $w = a_0a_1 \cdots a_{n-1} \in \Sigma^*$ and a transducer $\mathcal{T}$, we say that $u = b_0b_1 \cdots b_{n-1} \in O^+$ is a $\mathcal{T}$-*output* over w iff there is some run $s_0a_0b_0s_1 \cdots a_{n-1}b_{n-1}s_n$ of $\mathcal{T}$ over w.

Definition 1.2.21. A transducer $\mathcal{T}$ is *functional* if, for every input w, there exists a unique $\mathcal{T}$-output over w. If $\mathcal{T}$ is functional, we denote by $\mathcal{T}$ the map over Σ^* defined as $\mathcal{T}(w) = u$ iff u is the unique $\mathcal{T}$-output over w.

Transducers can be augmented with acceptance conditions so that a word u is a $\mathcal{T}$-output over w if u is the output of an accepting run of $\mathcal{T}$ over w.

Definition 1.2.22. A *non-deterministic finite Büchi transducer NFT* $\mathcal{T}$ is a tuple $(S, S_0, \Sigma, O, \Delta, F)$. A *run* of the transducer over the input word $w = a_0a_1 \cdots \in \Sigma^\omega$ is a finite or infinite sequence $s_0a_0b_0s_1a_1b_1s_2 \cdots$ such that $(s_i, a_i, b_i, s_{i+1}) \in \Delta$ for every $i \geq 0$. The transducer $\mathcal{T}$ accepts an ω-word w iff there exists an infinite run of $\mathcal{T}$ over w where infinitely many states are in F.

Definition 1.2.23. Given an input $w = a_0a_1 \cdots a_{n-1} \in \Sigma^+$ and a Büchi transducer $\mathcal{T}$, we say that $u = b_0b_1 \cdots b_{n-1} \in O^+$ is a $\mathcal{T}$-*output* over w iff there is some *accepting* run of $\mathcal{T}$ over w.

1.3 Reversing automata

For two alphabets Σ_1, Σ_2, we define sequential functions as maps of the form $f : \Sigma_1^* \to \Sigma_2^*$ from finite words over alphabet Σ_1 to finite words over alphabet Σ_2.

Definition 1.3.1. We define the *reverse* w^R of a finite word by $\varepsilon^R := \varepsilon$, and $(aw)^R := w^Ra$ where a is a letter and w a finite word. Equivalently for every finite word w, $w^R(t) := w(|w|-t-1)$, for $t < |w|$. If L is a language, the reverse of L is the language $L^R = \{w^R : w \in L\}$. The reverse of a *sequential function* between finite words $f : \Sigma_1^* \to \Sigma_2^*$ is the sequential function $f^R : \Sigma_1^* \to \Sigma_2^*$ such that $f^R(w) = (f(w))^R$.

The goal of this section is to explain how to represent the reverse of a regular language in a simple way by using automata. Intuitively, if we know how to build a DFA that accepts language L, then we can easily obtain a NFA that accepts L^R by inverting the arrows and switching initial and final states. Moreover, from an input-deterministic transducer representing the sequential function f, one can similarly construct an output deterministic transducer representing the sequential function f^R.

Formally, we define the reverse of a DFA $\mathbb{A} = (S, S_0, \Sigma, \Delta, F)$ as the NFA $\mathbb{A}^R := (S, F, \Sigma, \Delta^R, S_0)$ such that $(s, a, s') \in \Delta^R$ iff $\Delta(s', a) = s$ for all $s, s' \in S$,and $a \in \Sigma$.

And for any transducer with acceptance conditions $\mathcal{T} = (S, S_0, \Sigma, O, \Delta, F)$, the reverse transducer is defined as $\mathcal{T}^R := (S, F, \Sigma, \Delta^R, S_0)$ where $(s, a, b, s') \in \Delta^R$ iff $(s', a, b, s) \in \Delta$.

Theorem 1.3.1 (Reverse theorem)**.** *If $\mathbb{A}$ is a DFA, then $\mathbb{A}$ accepts w iff $\mathbb{A}^R$ accepts w^R. Moreover, if $\mathcal{T}$ is a transducer, then u^R is a $\mathcal{T}^R$-output over w^R iff u is a $\mathcal{T}$-output over w.*

Proof. By definition. □

Chapter 2

From LTL to timed automata

In the 60s, Büchi [6], Elgot [12], McNaughton [23] and Rabin [26], among others, discovered an interesting strategy for reasoning about decision problems in logic. Their idea was to establish relations among logics and classes of automata. With the help of such correspondences, it was easier to find reductions from decision problems in logic to decision problems about automata.

In this chapter, we present a translation from LTL formulas to finite automata from [14], representing the satisfiability of the formula over time. This translation serves as a preliminary version of the main construction shown in Chapter 3 for MITL.

2.1 Linear Temporal Logic

Formulas of propositional Linear Temporal Logic can be defined recursively from a finite set of propositional variables $P = \{p_1, p_2 \dots, p_k\}$, by means of boolean connectives $\neg, \vee$ and the following future and past modalities: $\mathcal{U}$ or *until*, $\bigcirc$ or *next*, $\mathcal{S}$ or *since*, $\ominus$ or *previously*.

Definition 2.1.1. The set of *LTL formulas* can be defined recursively by

- for every $p \in P$, p is an LTL formula,
- if φ is an LTL formula, then so is $\neg\varphi$,
- if φ, ψ are LTL formulas, then so is $\varphi \vee \psi$,
- if φ, ψ are LTL formulas, then so are $\varphi \mathcal{U} \psi, \bigcirc\varphi, \varphi \mathcal{S} \psi$ and $\ominus\varphi$.

Any formula with temporal modalities should consider unary operators $\bigcirc, \ominus$ as stronger as $\neg$. Thus, $\bigcirc p \mathcal{U} q$ should be read as $(\bigcirc p)\mathcal{U} q$ and similarly for $\ominus$. When dealing with $\mathcal{U}, \mathcal{S}, \vee$ at the same time in formulas such as $\varphi_1 \mathcal{U} \varphi_2 \mathcal{S} \varphi_3 \vee \varphi_4$, parenthesis should be added to avoid confusion.

From the basic operators, other standard operators can be derived, such as the remaining propositional connectives and other temporal modalities as

- $\Diamond\varphi := \top\mathcal{U}\varphi$ or *eventually* φ
- $\Box\varphi := \neg\Diamond\neg\varphi$ or *always* φ

To specify the meaning of an LTL formula, one needs to fix a representation of time, and then a class of objects encoding the truth-values of $p_1, \ldots, p_k$ over time. In this chapter, we see the timeline as having a starting point (informally, the present) and being a countable sequence of time instants, either bounded (represented by $[n] = \{0, \ldots, n-1\}$ for some $n \in \mathbb{N}$) or unbounded (represented by $\mathbb{N}$).

A timed interpretation of $P = \{p_1, \ldots, p_k\}$ is given by a word $w : D \to \{0,1\}^k$, viewed as a function from the time domain D, where D is $[n]$ or $\mathbb{N}$. At each time instant, w gives an evaluation of each propositional variable $p_1, \ldots, p_k$ in P. Thus, models of LTL formulas are objects of the form (w, t) where w is a D-word over $\{0,1\}^k$ and $t \in D$. During this chapter it will be of interest to represent, in a simple way, words that only talk about a single propositional variable p in P, as $w_p : D \to \{0,1\}$ or two propositional variables p, q in P, as $w_{p,q} : D \to \{0,1\}^2$, and so on. Moreover, given w, we will refer to the truth value of p at time t as $w_p(t)$.

From now on, we will treat simultaneously the cases of bounded and unbounded semantics and simply write w to refer to ω-words or finite words indistinctly. Nevertheless, comments will be made when there is an essential difference among the two.

The satisfaction relation $\vDash$ among pairs (w, t) giving a timed interpretation of P and LTL formulas can thus be defined as

$$
\begin{array}{lll}
(w,t) \vDash p & \text{iff} & w_p(t) = 1 \text{ for } p \in P, \\
(w,t) \vDash \neg\varphi & \text{iff} & (w,t) \nvDash \varphi, \\
(w,t) \vDash \varphi_1 \vee \varphi_2 & \text{iff} & (w,t) \vDash \varphi_1 \text{ or } (w,t) \vDash \varphi_2, \\
(w,t) \vDash \bigcirc\varphi & \text{iff} & (w,t+1) \vDash \varphi, \\
(w,t) \vDash \varphi_1 \mathcal{U} \varphi_2 & \text{iff} & \exists t' \geq t \text{ s.t. } (w,t') \vDash \varphi_2 \text{ and } \forall t'' \in [t,t') \text{ and } (w,t'') \vDash \varphi_1, \\
(w,t) \vDash \ominus\varphi & \text{iff} & t \neq 0 \text{ and } (w,t-1) \vDash \varphi, \\
(w,t) \vDash \varphi_1 \mathcal{S} \varphi_2 & \text{iff} & \exists t' \leq t \text{ s.t. } (w,t') \vDash \varphi_2 \text{ and } \forall t'' \in (t',t] \text{ and } (w,t'') \vDash \varphi_1.
\end{array}
$$

Note that we follow the convention that $w(0) \vDash \neg \ominus \varphi$ for all φ. For w bounded of length n, the satisfaction relation $\vDash$ remains the same, except for $\bigcirc\varphi$, where we have $(w, n-1) \vDash \neg \bigcirc \varphi$ for all φ. We use interval notation in the definitions for simplicity, but it should be clear that every t ranges over $\mathbb{N}$ or $[n]$.

Also note that we choose the non-strict interpretation of the until and since operator, which other authors replace by the strict ones

$$
\begin{array}{l}
(w,t) \vDash \varphi_1 \mathcal{U}^{str} \varphi_2 \quad \text{iff} \quad \exists t' > t, (w,t') \vDash \varphi_2 \text{ and } \forall t'' \in (t,t'), (w,t'') \vDash \varphi_1, \text{ and} \\
(w,t) \vDash \varphi_1 \mathcal{S}^{str} \varphi_2 \quad \text{iff} \quad \exists t' < t, (w,t') \vDash \varphi_2 \text{ and } \forall t'' \in (t',t), (w,t'') \vDash \varphi_1.
\end{array}
$$

However, we believe that our choice results in simplicity of the corresponding automata.

Definition 2.1.2. We will say that a bounded or unbounded word w satisfies the formula φ iff $(w, 0) \vDash \varphi$.

For our purposes, we are interested in viewing satisfaction as a relation between pairs φ, w and sets $\{t \in \mathbb{N} : (w, t) \vDash \varphi\}$ of time instants where φ holds in w.

Definition 2.1.3. For every formula φ, we define the *characteristic function* χ_φ of φ as the map $\chi_\varphi : (\{0,1\}^k)^D \to \{0,1\}^D$ that takes as argument a word $w : D \to \{0,1\}^k$ and outputs a word $\chi_\varphi(w) = u : D \to \{0,1\}$ such that

$$u(t) = 1 \quad \text{iff} \quad (w, t) \vDash \varphi,$$

where D is $\mathbb{N}$ or $[n]$.

The main goal of this section is to define, for every formula φ of LTL, a transducer $\mathcal{T}_\varphi$ that takes as input a word w and outputs the word $\chi_\varphi(w)$. At each time step, the automata reads the value of $w(t)$ and outputs the value $\chi_\varphi(w)(t)$ synchronously, performing a guess if needed, thereby yielding nondeterminism.

Since the truth-value of a formula $\psi_1 * \psi_2$ for any binary operator $*$ only depends on the truth-values of ψ_1 and ψ_2 and not on their forms, then it is sufficient to construct transducers for formulas $p_1 * p_2$ where $p_1, p_2 \in P$, and the same argument applies to unary operators. Following this perspective, we will define the promised construction in various steps. First, we will build the transducers for the basic formulas p, $\neg p$, $p_1 \vee p_2$, $\ominus p$, $\bigcirc p$, $p_1 \mathcal{S} p_2$ and $p_1 \mathcal{U} p_2$ for the case of bounded words, where p, p_1, p_2 are any propositional variables. Then, we will present the same transducers but for the case of unbounded words. Finally, given any LTL formula φ, we obtain the transducer for χ_φ, by composing the basic transducers according to the parse tree of the formula φ.

2.2 Temporal testers for LTL basic formulas

The transducers that we want to build have to read words in the alphabet $\{0,1\}^k$ and output words in the alphabet $\{0,1\}$. Therefore, the input and output alphabets of the automata should be $\Sigma = \{0,1\}^k$ and $O = \{0,1\}$. Because we deal with unary and binary operators, we will be interested in the truth-values of at most two propositional variables, and therefore restrict ourselves to $\Sigma = \{0,1\}^k$ for $k = 1, 2$.

For an arbitrary transducer $\mathcal{T} = (S, S_0, \Sigma, O, \Delta)$ and an arbitrary element $\delta = (s, a, b, s') \in \Delta$, we call a and b the input and output labels of δ, respectively, and sometimes denote them by a/b. Our transducers will have edges with input labels in $\{0,1\}^k$ and output labels in $\{0,1\}$. For states s, s' and an output label $b \in \{0,1\}$, we group all the outgoing edges of label x/b into a single one with label $\{x' : (s, x', b, s') \in \Delta\}/b$. To further enhance readability, we write a formula over P that can represent in a simple way the set of valuations

$\{x' : (s, x', b, s') \in \Delta\}$. Moreover, to harmonize the notation we write q or $\neg q$ for output 1 or 0.

As an illustration, consider a transducer with input alphabet $\Sigma = \{0,1\}^2$ and the following elements in Δ:

$$(s, (0,1), 1, s')$$
$$(s, (1,0), 1, s')$$
$$(s, (1,1), 1, s')$$
$$(s, (0,0), 0, s')$$

These four edges can be compressed into the following two edges

$$(s, \{(0,1), (1,0), (1,1)\}, 1, s')$$
$$(s, \{(0,0)\}, 0, s')$$

who can be represented using LTL formulas over $P = \{p_1, p_2\}$ and $Q = \{q\}$ as

$$(s, p_1 \vee p_2, q, s')$$
$$(s, \neg p_1 \wedge \neg p_2, \neg q, s')$$

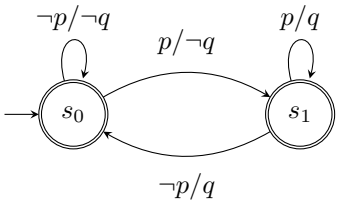


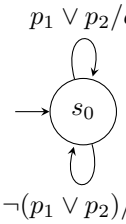
Figure 2.2.1: Temporal tester $\mathcal{T}_{\chi_{\ominus p}}$

Boolean and past basic formulas (p, $\neg p$, $p_1 \vee p_2$, $\ominus p$, $p_1 \mathcal{S} p_2$) are easy to identify with input-deterministic transducers since the value of its characteristic function is entirely determined by past and present letters (observations) recognized by the automaton.

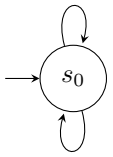
The case of propositional formulas is almost trivial. Every transducer is formed by a single state that has as many self-loops as evaluations of the variables that appear in the formula, and acts as a truth table. Formally, for every basic propositional formula φ, the transducer $\mathcal{T}_\varphi$ is the tuple

$$(\{s_0\}, \{s_0\}, \Sigma, \{0,1\}, \Delta, \emptyset)$$

where Σ is $\{0,1\}^2$ or $\{0,1\}$ and elements of Δ are of the form $(s_0, a, 1, s_0)$ if a satisfies the formula and $(s_0, a, 0, s_0)$ if a doesn't satisfy φ as in 2.2.2. We now deal with the construction of temporal testers for past basic formulas over bounded words. The key is that these transducers need to encode the information observed in the previous steps to produce an output at the current



(a) Abbreviated notation



(b) Extended notation

Figure 2.2.2: Transducer for $p_1 \vee p_2$

step. To this end, we associate a certain informal meaning to each state of the automaton. Let us illustrate this strategy by looking at the case of the temporal tester $\mathcal{T}_{\ominus p}$.

$\mathcal{T}_{\ominus p}$ is defined as the tuple $(S, S_0, P, Q, \Delta, F)$ where $S = \{s_0, s_1\}$, $S_0 = \{s_0\}$, $P = \{p\}$, $Q = \{q\}$, $F = S$, and Δ consists of elements $(s_0, \neg p, \neg q, s_0)$, $(s_0, p, \neg q, s_1), (s_1, p, q, s_1), (s_1, \neg p, q, s_0)$ as depicted in Figure 2.2.1.

This automaton can be interpreted in a simple way by explaining the meaning of the states. State s_0 means "I saw $\neg p$ in the previous time instant" and s_1 the opposite, "I saw p in the previous time instant". This translates into the fact that every incoming arrow in s_0 has read input $\neg p$ and every incoming arrow in s_1 has read input p. When observing $\neg p$, the automata then moves to (or stays in) state s_0 and when it reads p, it stays or moves to s_1. According to the meaning of $\ominus p$, after observing $\neg p$ (being in state s_0), the output should be $\neg q$, and this reflects in the automaton, since each outgoing arrow from s_0 has $\neg q$ as output. In a similar way, after observing p i.e. visiting state s_1, the output is always q.

Furthermore, the condition $w \vDash \neg \ominus \varphi$ is reflected in the automaton by putting s_0 as the unique initial state.

Proposition 2.2.1 (Correctness of $\mathcal{T}_{\ominus p}$)**.** *For every bounded w and every $t < |w|$, we have $(w, t) \vDash \ominus p$ iff $w \in L(\mathcal{T}_{\ominus p})$ and $\mathcal{T}_{\ominus p}(w)(t) = 1$.*

To understand the construction of the temporal tester for $p_1 \mathcal{S} p_2$ we study how the satisfiability of the formula $p_1 \mathcal{S} p_2$ is at an instant t according to the truth values of p_1 and p_2.

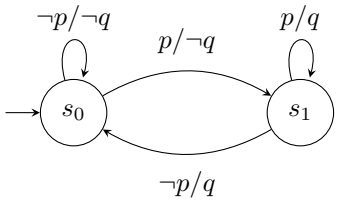


Figure 2.2.3: $\mathcal{T}^{unb}_{\ominus} p$ for unbounded words

Lemma 2.2.1. *For every w and every $p_1, p_2 \in P$, the following holds.*

(i) If $(w,t) \vDash \neg p_1 \wedge \neg p_2$ then $\chi_{p_1 \mathcal{S} p_2}(w)(t) = 0$.

(ii) If $(w,t) \vDash p_2$ then $\chi_{p_1 \mathcal{S} p_2}(w)(t) = 1$.

(iii) If $(w,t) \vDash p_1 \wedge \neg p_2$ then either $\chi_{p_1 \mathcal{S} p_2}(w)(t) = 1$ if p_1 has been holding from the last time p_2 held, or $\chi_{p_1 \mathcal{S} p_2}(w)(t) = 0$ if no p_2 has been observed before p_1 started holding.

Proof. By definition. □

We now proceed to define the temporal tester $\mathcal{T}_{p_1 \mathcal{S} p_2}$ for bounded words.

Let $\mathcal{T}_{p_1 \mathcal{S} p_2} = (S, S_0, P, Q, \Delta, F)$ where $S = \{s_0, s_1\}$, $S_0 = \{s_0\}$, $P = \{p_1, p_2\}$, $Q = \{q\}$, $F = S$, and Δ consists of elements $(s_0, p_2, q, s_1), (s_0, \neg p_2, \neg q, s_0), (s_1, p_1 \vee p_2, q, s_1), (s_1, \neg p_1 \wedge \neg p_2, \neg q, s_0)$ as illustrated in Figure 2.2.4.

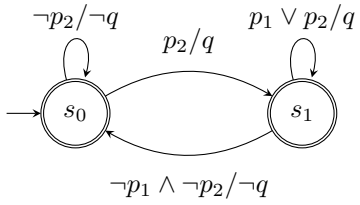
Conditions (i) and (ii) in Lemma 2.2.1 are reflected by transitions $(s_1, \neg p_1 \wedge \neg p_2, \neg q, s_0)$ and (s_0, p_2, q, s_1) respectively. In a state where $p_1 \wedge \neg p_2$ is observed, the output for a given w at instant t depends on the previous observations $\{w(t') : t' \leq t\}$, which are encoded by states s_0 and s_1. Thus, state s_1 is interpreted as "$p_1 \mathcal{S} p_2$ held at previous step" or, in terms of Lemma 3.2.4, p_1 has been continuously holding from the last time p_2 was true. State s_0 means exactly the opposite, "$p_1 \mathcal{S} p_2$ didn't hold at previous step", reflecting that no p_2 has been observed before a period where p_1 started to hold continuously.

Proposition 2.2.2 (Correctness of $\mathcal{T}_{p_1 \mathcal{S} p_2}$)**.** *For every bounded w and every $t < |w|$, we have $(w,t) \vDash p_1 \mathcal{S} p_2$ iff $w \in L(\mathcal{T}_{p_1 \mathcal{S} p_2})$ and $\mathcal{T}_{p_1 \mathcal{S} p_2}(w)(t) = 1$.*

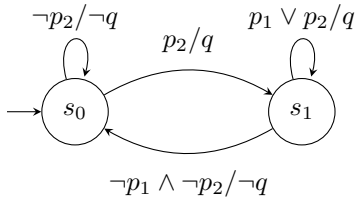
By looking at the structure of the previous automata we can infer that similar versions can be used for the case of unbounded words; we show them in Figures 2.2.3 and 2.2.4.

Proposition 2.2.3 (Correctness of $\mathcal{T}^{unb}_{\ominus p}$)**.** *For every unbounded w and every $t \in \mathbb{N}$, we have $(w,t) \vDash \ominus p$ iff $w \in L(\mathcal{T}^{unb}_{\ominus p})$ and $\mathcal{T}^{unb}_{\ominus p}(w)(t) = 1$.*

Proposition 2.2.4 (Correctness of $\mathcal{T}^{unb}_{p_1 \mathcal{S} p_2}$)**.** *For every unbounded w and every $t < |w|$, we have $(w,t) \vDash p_1 \mathcal{S} p_2$ iff $w \in L(\mathcal{T}^{unb}_{p_1 \mathcal{S} p_2})$ and $\mathcal{T}^{unb}_{p_1 \mathcal{S} p_2}(w)(t) = 1$.*



(a) $\mathcal{T}_{p_1 \mathcal{S} p_2}$ for bounded words



(b) $\mathcal{T}^{unb}_{p_1 \mathcal{S} p_2}$ for unbounded words

Figure 2.2.4: Bounded vs unbounded semantics

Transducers for future basic formulas $\bigcirc p, p_1 \mathcal{U} p_2$ over bounded semantics can be build by reversing the previous automata in the sense of Theorem 1.3.1, as the next proposition shows.

For given w, φ, we denote by $\chi^R_\varphi(w)$ the word $(\chi_\varphi(w))^R$.

Proposition 2.2.5. *For every bounded word w, and every formula φ, we have*

$$\chi_{\bigcirc\varphi}(w) = \chi^R_{\ominus\varphi}(w^R)$$

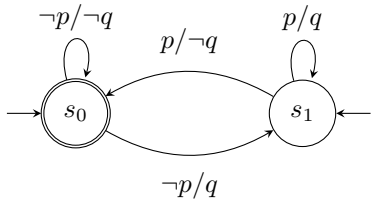
and for every φ_1, φ_2, we have

$$\chi_{\varphi_1 \mathcal{U} \varphi_2}(w) = \chi^R_{\varphi_1 \mathcal{S} \varphi_2}(w^R).$$

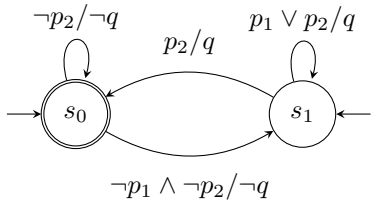
Proof. Let w be a bounded word and φ be an arbitrary formula. Then for every $t < |w|-1$, we have $\chi_{\bigcirc\varphi}(w)(t) = 1$ iff $(w, t+1) \vDash \varphi$ iff $(w^R, |w|-(t+1)-1) \vDash \varphi$ iff $(w^R, |w| - t - 2) \vDash \varphi$ iff $(w^R, |w| - t - 1) \vDash \ominus\varphi$ iff $\chi_{\ominus\varphi}(w^R)(|w| - t - 1) = 1$ iff $\chi^R_{\ominus\varphi}(w^R)(t) = 1$. For the case $t = |w| - 1$, $\chi_{\bigcirc\varphi}(w)(|w| - 1) = 0$ by definition. Hence, we want to show that $\chi^R_{\ominus\varphi}(w^R)(|w| - 1) = 0$. But observe that $\chi^R_{\ominus\varphi}(w^R)(|w|-1) = 0$ iff $\chi_{\ominus\varphi}(w^R)(0) = 0$ which holds by definition. Hence, for all $t < |w|$, $\chi_{\bigcirc\varphi}(w)(t) = \chi^R_{\ominus\varphi}(w)(t)$.

Now let φ_1, φ_2 be arbitrary formulas. By definition $\chi_{\varphi_1 \mathcal{U} \varphi_2}(w)(t) = 1$ iff $\exists t' \geq t$ s.t. $(w, t') \vDash p_2$ and $\forall t''$ s.t. $t \leq t'' < t', (w, t'') \vDash p_1$. This is equivalent to $\exists t'$ s.t. $|w| - t' - 1 \leq |w| - t - 1$ and $(w^R, |w| - t' - 1) \vDash \varphi_2$ and $\forall t''$ where $|w| - t' - 1 < |w| - t'' - 1 \leq |w| - t - 1$, we have $(w^R, |w| - t'' - 1) \vDash \varphi_1$. Therefore $\chi_{\varphi_1 \mathcal{S} \varphi_2}(w^R)(|w| - t - 1) = 1$, which can be rewritten as $\chi^R_{\varphi_1 \mathcal{S} \varphi_2}(w^R)(t) = 1$. □

Therefore we can define $\mathcal{T}_{\bigcirc p} := \mathcal{T}^R_{\ominus p}$ and $\mathcal{T}_{p_1 \mathcal{U} p_2} := \mathcal{T}^R_{p_1 \mathcal{S} p_2}$. These temporal testers are illustrated in Figure 2.2.5.



(a) Temporal tester for $\bigcirc p$ over bounded words



(b) Temporal tester for $p_1 \mathcal{U} p_2$ over bounded words

Figure 2.2.5: Temporal testers for future operators over bounded words

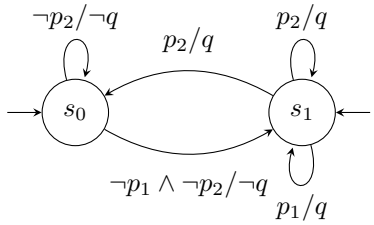


Figure 2.2.6: Temporal tester for $p_1 \mathcal{U} p_2$ over unbounded words

Lastly, we deal with the case of $\mathcal{T}_{p_1 \mathcal{U} p_2}$ for unbounded words. This temporal tester depicted in Figure 2.2.6 is a Büchi automaton. It uses an edge Büchi condition [15] that includes all edges of the tester except for the self-loop labeled p_1 in state s_1. Such a Büchi condition excludes those output words ending in an infinite suffix of q's that result from words ending in an infinite suffix of $p_1 \wedge \neg p_2$'s.

Chapter 3

From MITL to timed automata

In Chapter 2 we have seen a simple translation from every formula of LTL to a finite transducer representing its satisfiability in a precise way. In this chapter, we switch to continuous time and discuss a much more interesting translation for the logic MITL interpreted over real-time semantics into the class of Timed Automata.

Metric Interval Temporal Logic (MITL) [2], which is both a syntactic restriction of MTL [20], and a real-time extension of LTL, is extensively used as a formalism to specify quantitative real-time properties. Here, MITL (over a set of propositions P) is interpreted over a kind of real-time words called *boolean signals*. Boolean signals are functions from $\mathbb{R}_0^+$ associating a valuation of P to every time instant $t \in \mathbb{R}_0^+$. Furthermore, an additional condition is imposed on their discontinuities, excluding those with an infinite number of discontinuities in a bounded interval of the domain.

After explaining a rewriting of every formula into a *standard form* which only uses operators $\mathcal{U}_{(0,\infty)}, \mathcal{S}_{(0,\infty)}, \Diamond_{(0,a)}, \blacklozenge_{(0,a)}$, we present the two main ways of composing timed transducers. Then for a given MITL formula φ, we define an automaton $\mathcal{T}_\varphi$ taking as input an arbitrary boolean signal w and producing another boolean signal representing the satisfiability of φ throughout w. The automaton $\mathcal{T}_\varphi$ is defined in a compositional fashion, from four basic temporal testers that are combined according to the parse tree of φ.

The strength of this unique translation rests in its clarity and modularity, providing an effective technique that is easily adaptable to other situations.

3.1 Metric Interval Temporal Logic

The logic MITL is a formalism to specify quantitative (linear) real-time properties. To do so, two main temporal modalities are employed, $\mathcal{U}_I$ and $\mathcal{S}_I$, where I is an arbitrary interval of $\mathbb{R}_0^+$ with rational end-points. Their intepretation

is simple: a formula $p\mathcal{U}_{(a,b)}q$ is true at time t if q is true at some point within the time frame $(t+a, t+b)$ and p holds from t *until* q becomes true; cases $[a,b), [a,b], (a,b]$ have analogous interpretations and $\mathcal{S}$ works symmetrically. The main particularity of MITL consists in avoiding singular intervals in the subscripts. Therefore, in MITL we are not able to express the exact amount of time between two events but only an approximation, while in return we get more efficient decision procedures.

Formulas of MITL can be defined recursively from a finite set of propositional variables $P = \{p_1, p_2 \ldots, p_k\}$, by means of boolean operators $\neg, \vee$ and the following future and past modalities

- $\mathcal{U}_I$ or *timed-constrained until*,
- $\mathcal{S}_I$ or *timed-constrained since*.

If φ, ψ are formulas of MITL and I is an interval over $\mathbb{R}_0^+$ with rational endpoints, then $\varphi \mathcal{U}_I \psi$ and $\varphi \mathcal{S}_I \psi$ are formulas of MITL as well. Observe that LTL operators $\bigcirc, \ominus$ no longer make sense in the presence of dense time, due to the lack of a unique successor of a real number. However, as in the case of LTL, other time-constrained operators can be derived from the basic ones

- $\Diamond_I \varphi := \top \mathcal{U}_I \varphi$ or *time-constrained eventually*
- $\blacklozenge_I \varphi := \top \mathcal{S}_I \varphi$ or *time-constrained once*
- $\Box \varphi := \neg \Diamond \neg \varphi$ or *time-constrained always*
- $\blacksquare \varphi := \neg \blacklozenge \neg \varphi$ or *time-constrained historically*

As time domain, now we choose the set of nonnegative real numbers, denoted by $\mathbb{R}_0^+$, and signals as semantic universes. We remind to the reader the definition of a signal.

Definition 3.1.1. A *signal* over Σ is a function $w : \mathbb{R}_0^+ \to \Sigma$ that admits a compatible point-segment partition $\mathcal{I}$.

Definition 3.1.2. A *bounded signal* is a function $w : [0, r) \to \Sigma$ for some $r > 0$ and such that w has finitely many discontinuities over $[0, r)$.

Observe that w is a bounded signal iff it is defined over a bounded interval of $[0, r)$ and there exists a function $\bar{w}$ extending w such that $\bar{w}$ is a signal. Further, bounded signals can also be decomposed as a concatenation of point and open signals according to a compatible point-segment partitions of its domain.

In our presentation, the objects that verify or falsify MITL formulas are pairs (w, t) for w a signal (or bounded signal in I) over $\{0,1\}^k$ and $t \in \mathbb{R}_0^+$ (resp. $t \in I$). We refer to these signals as *k-boolean signals*. Sometimes we will consider signals that talk about a smaller group of propositional variables $P' \subseteq P$, and refer to them generally as *boolean signals*. We will join and separate signals in the following way:

Definition 3.1.3. The *projection* to the variable $p_i \in P$ of a signal w is the 1-boolean signal $w|_{p_i}$ such that $w|_{p_i}(t) = w(t)(i)$ for all t. The *projection* to $\{p_i, p_j\} \subseteq P$ for $(p_i \neq p_j)$ of a signal w is the 2-boolean signal $w|_{p_i,p_j}$ such that $w|_{p_i,p_j}(t)(0) = w(t)(i)$ and $w|_{p_i,p_j}(t)(1) = w(t)(j)$ for all t. Lastly, for a finite set $P' = \{p_{i_1}, \cdots, p_{i_n}\} \subseteq P$, the projection of w to P' is $w_{|P'}$ defined as $w_{|P'}(t)(j) = w(t)(i_j)$ for all t.

The *pairing* $w||w'$ of two 1-boolean signals w, w' as $(w||w')(t) = (w(t), w'(t))$. If P_1, P_2 are disjoint subsets of P, then the pairing of a n_1-signal w_1 over P_1 and a n_2-signal w_2 over P_2 is defined as

$$(w_1||w_2)(t) = (w_1(t)(0), \ldots, w_1(t)(n_1 - 1), w_2(t)(0), \ldots, w_2(t)(n_2 - 1)).$$

We interpret any MITL formula φ with respect to boolean signals as

$$\begin{aligned}
&(w,t) \vDash p && \text{iff} \quad w_{|p}(t) = 1,\\
&(w,t) \vDash \neg\varphi && \text{iff} \quad (w,t) \nvDash \varphi,\\
&(w,t) \vDash \varphi_1 \vee \varphi_2 && \text{iff} \quad (w,t) \vDash \varphi_1 \text{ or } (w,t) \vDash \varphi_2,\\
&(w,t) \vDash \varphi_1 \mathcal{U}_I \varphi_2 && \text{iff} \quad \exists t' \in t \oplus I, (w,t') \vDash \varphi_2 \text{ and } \forall t'' \in (t,t'), (w,t'') \vDash \varphi_1,\\
&(w,t) \vDash \varphi_1 \mathcal{S}_I \varphi_2 && \text{iff} \quad \exists t' \in t \ominus I, (w,t') \vDash \varphi_2 \text{ and } \forall t'' \in (t',t), (w,t'') \vDash \varphi_1.
\end{aligned}$$

and with respect to bounded boolean signals of domain D, for $t \in D$, as

$$\begin{aligned}
&(w,t) \vDash p && \text{iff} \quad w_{|p}(t) = 1,\\
&(w,t) \vDash \neg\varphi && \text{iff} \quad (w,t) \nvDash \varphi,\\
&(w,t) \vDash \varphi_1 \vee \varphi_2 && \text{iff} \quad (w,t) \vDash \varphi_1 \text{ or } (w,t) \vDash \varphi_2,\\
&(w,t) \vDash \varphi_1 \mathcal{U}_I \varphi_2 && \text{iff} \quad \exists t' \in (t \oplus I) \cap D, (w,t') \vDash \varphi_2 \text{ and } \forall t'' \in (t,t'), (w,t'') \vDash \varphi_1,\\
&(w,t) \vDash \varphi_1 \mathcal{S}_I \varphi_2 && \text{iff} \quad \exists t' \in t \ominus I, (w,t') \vDash \varphi_2 \text{ and } \forall t'' \in (t',t), (w,t'') \vDash \varphi_1.
\end{aligned}$$

where $t \oplus I := \{t + i : i \in I\}$ and $t \ominus I := \{t - i : i \in I\}$. We say that a (bounded) signal w *satisfies* the formula φ iff $(w, 0) \vDash \varphi$.

Definition 3.1.4. For every formula φ, we define the *characteristic function* χ_φ of φ as the map that takes as argument a (resp. bounded) boolean signal w and outputs a (resp. bounded) 1-boolean signal $\chi_\varphi(w)$ such that

$$\chi_\varphi(w)(t) = 1 \quad \text{iff} \quad (w,t) \vDash \varphi.$$

The remaining of this section is devoted to show that signals are suitable for interpreting MITL. The following lemma follows clearly from the definition.

Lemma 3.1.1. *For every interval $I \subseteq \mathbb{R}_0^+$, a boolean signal is continuous in I iff it is constant in I.*

First, we observe that boolean signals are closed under projection and pairing.

Lemma 3.1.2. *If w is a signal over P, then $w|_{P'}$ is also a signal for every $P' \subseteq P$. Moreover, if w_1, w_2 are two boolean signals over two disjoint subsets P_1, P_2 of P respectively, then $w_1||w_2$ is a signal.*

Proof. At a given bounded interval, $w|_{P'}$ has at most the same discontinuities as w, hence finitely many. Then $\mathcal{I}_{w|_{P'}}$, the point segment partition induced by the discontinuities of $w|_{P'}$, is compatible with $w|_{P'}$, whence $w|_{P'}$ is a signal. Similarly, the discontinuities of $w_1||w_2$ at a given bounded interval are the union of the discontinuities of w_1, w_2 at that interval, and thus finite. Then, $\mathcal{I}_{w_1||w_2}$ is compatible with $w_1||w_2$ and then it is a signal. □

Lemma 3.1.3. *Given a bounded interval I and a signal w, there exist $i \in \{1, 2\}$ such that if $\chi_{\varphi_1 \mathcal{U}_I \varphi_2}$ has a discontinuity in t_0 then χ_{φ_i} has a discontinuity in $t_0 \oplus I \cup t_0 \ominus I$.*

Proof. Let $t_0 > 0$ be a time instant such that $\chi_{\varphi_1 \mathcal{U}_I \varphi_2}$ has a discontinuity in t_0. There are four cases

- $\lim_{t \to t_0^-} \chi_{\varphi_1 \mathcal{U}_I \varphi_2}(t) = 0$ and $\chi_{\varphi_1 \mathcal{U}_I \varphi_2}(t_0) = 1$.
- $\lim_{t \to t_0^-} \chi_{\varphi_1 \mathcal{U}_I \varphi_2}(t) = 1$ and $\chi_{\varphi_1 \mathcal{U}_I \varphi_2}(t_0) = 0$.
- $\lim_{t \to t_0^+} \chi_{\varphi_1 \mathcal{U}_I \varphi_2}(t) = 0$ and $\chi_{\varphi_1 \mathcal{U}_I \varphi_2}(t_0) = 1$.
- $\lim_{t \to t_0^+} \chi_{\varphi_1 \mathcal{U}_I \varphi_2}(t) = 1$ and $\chi_{\varphi_1 \mathcal{U}_I \varphi_2}(t_0) = 0$.

We only prove the first case, others are analogous. Suppose $\lim_{t \to t_0^-} \chi_{\varphi_1 \mathcal{U}_I \varphi_2}(t) = 0$ and $\chi_{\varphi_1 \mathcal{U}_I \varphi_2}(t_0) = 1$. This means there exists some $t_2 \in t_0 \oplus I$ such that $(w, t_2) \vDash \varphi_2$ and for all $t_1 \in (t_0, t_2)$, we have $(w, t_1) \vDash \varphi_1$. If I is right-closed, then either $t_2 = sup(I)$ or $t_2 < sup(I)$. If $t_2 = sup(I)$ and we assume it is the first such t_2 in $t_0 \oplus I$, then $\lim_{t \to (t_0 + sup(I))^-} \chi_{\varphi_2}(t) = 0$ but $\chi_{\varphi_2}(t_0 + sup(I)) = 1$. Hence, χ_{φ_2} has a discontinuity at $t_0 \oplus I$. Otherwise, $t_2 < sup(I)$, and then there is some positive $\varepsilon < sup(I) - t_2$ s.t. for all $t \in (t_0 - \varepsilon, t_0)$, $t_2 \in t \oplus I$. But since $\lim_{t \to t_0^-} \chi_{\varphi_1 \mathcal{U}_I \varphi_2}(t) = 0$, then there has to be some time point $t_1 \in (t, t_0)$ where φ_1 doesn't hold. At the same time φ_1, holds along (t_0, t_2), and then we can assure χ_{φ_1} has a discontinuity at some $t_1 \in (t, t_0)$. Therefore, χ_{φ_1} has a discontinuity at $t_0 \ominus I$.

If I is right-open, then $t_2 < sup(I)$ and by the same argument as above χ_{φ_1} has a discontinuity at $t_0 \ominus I$. □

Lemma 3.1.4. *Given an interval I and a signal w, there exist $i \in \{1, 2\}$ such that if $\chi_{\varphi_1 \mathcal{S}_I \varphi_2}$ has a discontinuity in t_0 then χ_{φ_i} has a discontinuity in $t_0 \oplus I \cup t_0 \ominus I$.*

Proof. Symmetric to Lemma 3.1.3. □

Lemma 3.1.5. *Let φ_1, φ_2 be formulas and I be an unbounded interval. Then for every signal w, the function $\chi_{\varphi_1 \mathcal{U}_I \varphi_2}(w)$ is a signal.*

Proof. For a contradiction, suppose $\chi_{\varphi_1 \mathcal{U}_I \varphi_2}(w)$ has infinitely many discontinuities in a bounded interval J. Then there exists an increasing converging sequence $(t_n)_{n \in \mathbb{N}}$ with limit denoted by a such that $\varphi_1 \mathcal{U}_I \varphi_2$ is true at (w, t_0), false at (w, t_1), true at (w, t_2) and so on.

Since $\chi_{\varphi_1}(w), \chi_{\varphi_2}(w)$ have a finite number of discontinuities in J, then there is some $m \in \mathbb{N}$ such that $\chi_{\varphi_1}(w)$ is constant over (t_m, a). But then it has to be constantly 1 because for all even indexes n there is some $t' > t_n$ such that φ_1 holds true at (t_n, t').

Now let $n \geq m$ be odd. Then $\varphi_1 \mathcal{U}_I \varphi_2$ is false at (w, t_n). This can be for two reasons. One possibility is that there is no $t' \in t_n \oplus I$ where φ_2 holds. But this is a contradiction because $\varphi_1 \mathcal{U}_I \varphi_2$ holds at t_{n+1} and $t_n \oplus I \subseteq t_{n+1} \oplus I$. Then it must be because there is some $t''_n \in (t_n, t_{n+1})$ where φ_1 doesn't hold i.e. $\chi_{\varphi_1}(w)(t) = 0$. Contradiction. □

Lemma 3.1.6. *Let φ_1, φ_2 be formulas and I be an unbounded interval. Then for every signal w, the function $\chi_{\varphi_1 \mathcal{S}_I \varphi_2}(w)$ is a signal.*

Proof. Analogous to 3.1.5. □

The next theorem shows that MITL is interpretable over (bounded) boolean signals.

Theorem 3.1.1. *For every MITL formula φ and every (bounded) boolean signal w over P, $\chi_\varphi(w)$ is a (bounded) 1-boolean signal.*

Proof. Given a signal w, we proceed by induction on the complexity of φ. If $\varphi = p$, then $\chi_p(w)$ is the projection $w|_p$. If $\varphi = \neg\psi$, then $\chi_{\neg\psi}(w) = 1 - \chi_\psi(w)$, whence $\chi_{\neg\psi}(w)$ has a discontinuity in t iff $\chi_\psi(w)$ has a discontinuity in t. Then, by induction hypothesis, $\chi_{\neg\psi}(w)$ is a 1-boolean signal. If $\varphi = \varphi_1 \vee \varphi_2$, then $\chi_\varphi(w) = max\{\chi_{\varphi_1}(w), \chi_{\varphi_2}(w)\}$ and it's easy to see that if $\chi_\varphi(w)$ has a discontinuity in t then some $\chi_{\varphi_i}(w)$ has a discontinuity in t for $i = 1, 2$. Then, again by induction hypothesis, $\chi_{\varphi_1 \vee \varphi_2}(w)$ is a 1-boolean signal. If $\varphi = \varphi_1 \mathcal{U}_I \varphi_2$, apply Lemma 3.1.3 for I bounded, and Lemma 3.1.5 for I unbounded. We infer that $\chi_{\varphi_1 \vee \varphi_2}(w)$ has finitely many discontinuities at every bounded interval and then it is a signal. Similar for $\varphi = \varphi_1 \mathcal{S}_I \varphi_2$. □

3.2 Standard form

The main goal of this section is to show that the language of MITL can be restricted to the propositional connectives and to temporal operators that talk about immediate future and immediate past with a bounding parameter $a > 0$. In particular, we are interested in the following *standard* operators

$$\mathcal{U}_{(0,\infty)}, \mathcal{S}_{(0,\infty)}, \Diamond_{(0,a)}, \blacklozenge_{(0,a)}$$

where $a \in \mathbb{Q}^+$. Moreover, we shall measure what is the complexity cost when a formula is rewritten into another formula only containing standard operators. To this end, we introduce two parameters that measure the complexity of a MITL formula, the *size*, and the *resolution*. The first counts the number of its subformulas, while the second has to do with the relation among length and distance to the origin of the intervals appearing at subscripts in the formula.

What we want to prove exactly is stated in the following theorem.

Theorem 3.2.1. *Any MITL formula φ is equivalent to some formula φ' that only contains the temporal operators $\mathcal{U}, \mathcal{S}, \Diamond_{(0,a)}, \blacklozenge_{(0,a)}$, where a is a positive rational number. Moreover, $s(\varphi') \in O[s(\varphi) \cdot r(\varphi)]$ and $r(\varphi') \leq 1$ where s is the size and r is the resolution.*

Suppose φ is a MITL formula containing propositional connectives and operators $\mathcal{U}_I, \mathcal{S}_I$ for $I \subseteq \mathbb{R}_0^+$ an arbitrary interval with rational endpoints. To find φ' as in the theorem, we divide the elimination of the non-standard operators into four main steps, and use equivalences in MITL that behave nicely with respect to our complexity parameters. To give the reader an overview of the whole strategy, we summarize the steps that shall be taken:

1. Rewrite bounded $\mathcal{U}$ (resp. $\mathcal{S}$) in terms of unbounded $\mathcal{U}$ (resp. $\mathcal{S}$) and bounded $\Diamond$ (resp. $\blacklozenge$).
2. Rewrite unbounded $\mathcal{U}$ (resp. $\mathcal{S}$) in terms of $\mathcal{U}_{(0,\infty)}$ (resp. $\mathcal{S}_{(0,\infty)}$ and bounded $\Diamond$ (resp. $\blacklozenge$).
3. Rewrite bounded $\Diamond$ (resp. $\blacklozenge$) in terms of $\Diamond_I$ (resp. $\blacklozenge_I$) where $\inf I = 0$.
4. Rewrite right-closed or open-closed $\Diamond_{(0,a]}, \Diamond_{[0,a)}, \Diamond_{[0,a]}$ (resp. $\blacklozenge$) in terms of $\Diamond_{(0,a)}$ (resp. $\blacklozenge_{(0,a)}$) and $\mathcal{U}_{(0,\infty)}$ (resp. $\mathcal{S}_{(0,\infty)}$).

Definition 3.2.1. A formula φ is in *standard form* iff it only contains propositional connectives and standard operators $\mathcal{U}_{(0,\infty)}, \mathcal{S}_{(0,\infty)}, \Diamond_{(0,a)}, \blacklozenge_{(0,a)}$ where a is a positive rational number.

For two arbitrary formulas φ_1, φ_2, we say that $\varphi_1 \equiv \varphi_2$ iff for every signal w and $t \in \mathbb{R}_0^+$, we have that $(w,t) \vDash \varphi_1 \Leftrightarrow (w,t) \vDash \varphi_2$.

Lemma 3.2.1. *For all formulas φ_1, φ_2,*

- $(w,t) \vDash \varphi_1 \mathcal{U}_{(0,\infty)} \varphi_2$ *iff* $\exists t' > t$ *s.t.* $(w,t') \vDash \varphi_2$ *and for all* $t'' \in (t,t')$ *we have* $(w,t'') \vDash \varphi_1$,
- $(w,t) \vDash \varphi_1 \mathcal{S}_{(0,\infty)} \varphi_2$ *iff* $\exists t' < t$ *s.t.* $(w,t') \vDash \varphi_2$ *and for all* $t'' \in (t',t)$ *we have* $(w,t'') \vDash \varphi_1$.

Proof. For the first, it suffices to observe that, for any $t, t' \in \mathbb{R}_0^+$, $t' > t$ iff $t' \in t \oplus (0,\infty)$. The second is proven symetrically. □

The equivalences shown above make sense of the notation $\varphi_1 \mathcal{U} \varphi_2 := \varphi_1 \mathcal{U}_{(0,\infty)} \varphi_2$ and $\varphi_1 \mathcal{S} \varphi_2 := \varphi_1 \mathcal{S}_{(0,\infty)} \varphi_2$, where $\mathcal{U}$ is the symbol for LTL strict *until* interpreted in MITL. The non-strict *until* can also be recovered as we show below.

Lemma 3.2.2. *For all formulas φ_1, φ_2,*

- $\varphi_2 \vee (\varphi_1 \wedge (\varphi_1 \,\mathcal{U}\, \varphi_2))$ *iff* $\exists t' \geq t$ *s.t.* $(w,t') \vDash \varphi_2$ *and for all* $t'' \in [t,t')$ *we have* $(w,t'') \vDash \varphi_1$,
- $\varphi_2 \vee (\varphi_1 \wedge (\varphi_1 \,\mathcal{S}\, \varphi_2))$ *iff* $\exists t' \geq t$ *s.t.* $(w,t') \vDash \varphi_2$ *and for all* $t'' \in (t',t]$ *we have* $(w,t'') \vDash \varphi_1$.

Proof. For the first equivalence, pick some w, t such that $(w,t) \vDash \varphi_1 \underline{\mathcal{U}} \varphi_2$. This means that there is some $t' \geq t$ such that $(w,t') \vDash \varphi_2$ and for all $t'' \in [t,t')$, $(w,t'') \vDash \varphi_1$. Then either $t' = t$ and $(w,t) \vDash \varphi_2$ or $t' > t$ and $(w,t) \vDash \varphi_1$ together with $(w,t'') \vDash \varphi_2$ for all $t'' \in (t,t')$ and $(w,t') \vDash \varphi_2$, but this is equivalent to $(w,t) \vDash \varphi_2 \vee (\varphi_1 \wedge (\varphi_1 \mathcal{U} \varphi_2))$. Conversely, if $(w,t) \vDash \neg(\varphi_1 \underline{\mathcal{U}} \varphi_2)$, then $(w,t) \vDash \neg\varphi_2$ and if $(w,t) \vDash \varphi_1$, then $(w,t) \vDash \neg(\varphi_1 \mathcal{U} \varphi_2)$. The second is proven symetrically. □

Thus, we use the notation $\varphi_1 \underline{\mathcal{U}} \varphi_2 := \varphi_2 \vee (\varphi_1 \wedge (\varphi_1 \mathcal{U} \varphi_2))$ and $\varphi_1 \underline{\mathcal{S}} \varphi_2 := \varphi_2 \vee (\varphi_1 \wedge (\varphi_1 \mathcal{S} \varphi_2))$, where $\underline{\mathcal{U}}$ is the symbol for the LTL non-strict *until* interpreted in MITL.

The previous observations show that, in a certain way, MITL is an extension of LTL. From now on, we write the standard operators as

$$\mathcal{U}, \mathcal{S}, \Diamond_{(0,a)}, \blacklozenge_{(0,a)}$$

where $a \in \mathbb{Q}^+$.

Next, we see how to define a notion of succession that is suitable in the dense-time context.

Definition 3.2.2. Define the operators $\bigcirc_{\mathbb{R}} \varphi := \varphi \mathcal{U} \varphi$ and $\ominus_{\mathbb{R}} \varphi := \varphi \mathcal{S} \varphi$. In other words,

- $(w,t) \vDash \bigcirc_{\mathbb{R}} \varphi$ iff $\exists t' > t$ s.t. $(w,t') \vDash \varphi$ and for all $t'' \in (t,t')$ we have $(w,t'') \vDash \varphi$,
- $(w,t) \vDash \ominus_{\mathbb{R}} \varphi$ iff $\exists t' < t$ s.t. $(w,t') \vDash \varphi$ and for all $t'' \in (t',t)$ we have $(w,t'') \vDash \varphi$.

Remark. Observe that in discrete-time semantics, $\bigcirc_{\mathbb{R}} \varphi \equiv \bigcirc \varphi$ and $\ominus_{\mathbb{R}} \varphi \equiv \ominus \varphi$ for each formula φ. If $(w,t) \vDash \bigcirc_{\mathbb{R}} \varphi$, then there is some $t' > t$ such that $(w,t') \vDash \varphi$ and for all $t < t'' < t' (t'' \in \mathbb{N})$ we have that $(w,t'') \vDash \varphi$. Then $(w,t+1) \vDash \varphi$ which is $(w,t) \vDash \bigcirc \varphi$. Conversely, if $(w,t) \vDash \bigcirc \varphi$, then there is $t' > t$ $(t' = t+1)$ such that $(w,t') \vDash \varphi$ and for all t'' between t and t' (none), then $(w,t'') \vDash \varphi$. Hence $(w,t) \vDash \bigcirc_{\mathbb{R}} \varphi$. Symetrically for $\ominus_{\mathbb{R}}$.

For the sake of readibility, throughout this Chapter we simply write $\bigcirc, \ominus$. We propose two ways of measuring the degree of complexity of a MITL formula.

Definition 3.2.3. We define the set of subformulas $Sb(\varphi)$ of a formula φ recursively as

$$\begin{aligned} Sb(p) &= \{p\} \\ Sb(\neg\varphi) &= Sb(\varphi) \cup \{\neg\varphi\} \\ Sb(\varphi \vee \psi) &= Sb(\varphi) \cup Sb(\psi) \cup \{\varphi \vee \psi\} \\ Sb(\varphi \mathcal{U}_I \psi) &= Sb(\varphi) \cup Sb(\psi) \cup \{\varphi \mathcal{U}_I \psi\} \\ Sb(\varphi \mathcal{S}_I \psi) &= Sb(\varphi) \cup Sb(\psi) \cup \{\varphi \mathcal{S}_I \psi\} \end{aligned}$$

We define the *size* $s(\varphi)$ of a formula recursively as the cardinality of $Sb(\varphi)$.

Further, we define the *resolution* $r(\varphi)$ of a formula recursively, as

$$\begin{aligned} r(p) &= 0 \\ r(\varphi \vee \psi) &= \max\{r(\varphi), r(\psi)\} \\ r(\neg\varphi) &= r(\varphi) \\ r(\varphi \mathcal{U}_I \psi) = r(\varphi \mathcal{S}_I \psi) &= \max\{r(\varphi), r(\psi), \bar{r}(I)\} \end{aligned}$$

where $\bar{r}(I) = 2\lceil \frac{\inf I}{\sup I - \inf I} \rceil + 1$ if $\sup I < \infty$ and $\bar{r}(I) = 1$ otherwise.

Even though the role of $\bar{r}$ in our analysis will become clear in Lemmas 3.2.9 and 3.2.10, we compute the size and resolution of the following formulas as an illustration of the definitions above.

Example. 1. If $\varphi = p \wedge p \wedge p \wedge p$, then $s(\varphi) = 4$ and $r(\varphi) = 0$.

2. If $\varphi = p \wedge p\mathcal{U}_{(100,101)}q$, then $s(\varphi) = 4$ and $r(\varphi) = max\{r(p), r(p\mathcal{U}_{(100,101)}q)\} = max\{r(p), r(q), \bar{r}((100,101))\} = \bar{r}((100,101)) = 2\lceil \frac{100}{101-100} \rceil + 1 = 201$.
3. If $\varphi = \Diamond_{(50,51]}p$, then $s(\varphi) = 2$ and $r(\varphi) = \bar{r}((50,51]) = 2 \cdot 50 + 1 = 101$.
4. If $\varphi = \Diamond_{[100,150)}p$, then $s(\varphi) = 2$ and $r(\varphi) = \bar{r}([100,150)) = 2 \cdot 2 + 1 = 5$.
5. If $\varphi = \blacklozenge_{(0,5)}p \wedge p\mathcal{U}_{(0,100)}q$, then $s(\varphi) = 5$ and $r(\varphi) = 1$.

In the remaining of this section, we justify that the eliminations announced in steps 1-4 can be performed and how they affect our complexity parameters. In steps 1 and 2, we shall use the following lemmas. We obtain a way to express the timed *until* by a combination of untimed *until* and bounded *eventually* in the following lemma.

Lemma 3.2.3. *For any rational numbers a, b, c such that $0 \leq a < b < \infty$ and $0 < c < \infty$, and formulas φ_1, φ_2 we have*

$$\begin{aligned} &\varphi_1 \mathcal{U}_{(a,b)} \varphi_2 \equiv \varphi_1 \mathcal{U}_{(a,\infty)} \varphi_2 \wedge \Diamond_{(a,b)} \varphi_2 \\ &\varphi_1 \mathcal{U}_{(a,b]} \varphi_2 \equiv \varphi_1 \mathcal{U}_{(a,\infty)} \varphi_2 \wedge \Diamond_{(a,b]} \varphi_2 \qquad \varphi_1 \mathcal{U}_{(c,\infty)} \varphi_2 \equiv \Box_{(0,c]}(\varphi_1 \wedge \varphi_1 \mathcal{U} \varphi_2) \\ &\varphi_1 \mathcal{U}_{[a,b)} \varphi_2 \equiv \varphi_1 \mathcal{U}_{[a,\infty)} \varphi_2 \wedge \Diamond_{[a,b)} \varphi_2 \qquad \varphi_1 \mathcal{U}_{[c,\infty)} \varphi_2 \equiv \Box_{(0,c)} \varphi_1 \wedge \Box_{(0,c]}(\varphi_1 \underline{\mathcal{U}} \varphi_2) \\ &\varphi_1 \mathcal{U}_{[a,b]} \varphi_2 \equiv \varphi_1 \mathcal{U}_{(a,\infty)} \varphi_2 \wedge \Diamond_{[a,b]} \varphi_2 \end{aligned}$$

Proof. We prove the first and fifth equivalences as the rest can be proved in a similar way. To show the first equivalence we let w be a signal and $t \in \mathbb{R}_0^+$. We will show that $(w,t) \vDash \varphi_1 \mathcal{U}_{(a,b)} \varphi_2$ iff $(w,t) \vDash \varphi_1 \mathcal{U}_{(a,\infty)} \varphi_2 \wedge \Diamond_{(a,b)} \varphi_2$. The left-to-right implication holds since the existence of a point $t' \in (t+a, t+b)$ such that $(w,t') \vDash \varphi_2$ and for all $t'' \in (t,t')$, $(w,t'') \vDash \varphi_1$, implies that $t' \in (t+a, \infty)$ and $(w,t) \vDash \varphi_1 \mathcal{U}_{(a,\infty)} \varphi_2 \wedge \Diamond_{(a,b)} \varphi_2$. To show the right-to-left implication, we assume that $(w,t) \vDash \varphi_1 \mathcal{U}_{(a,\infty)} \varphi_2 \wedge \Diamond_{(a,b)} \varphi_2$. This means that there exists $t_1 \in (t+a, \infty)$ such that $(w,t_1) \vDash \varphi_2$ and for all φ_1 holds along the interval (t,t_1). Moreover, there is some $t_2 \in (t+a, t+b)$ such that $(w,t_2) \vDash \varphi_2$. We distinguish two cases: if $t_1 \leq t_2$ then $t_1 \in (t+a, t+b)$ and t_1 is a witness for $\varphi_1 \mathcal{U}_{(a,b)} \varphi_2$; if $t_1 > t_2$, then φ_1 holds along (t,t_2) and t_2 is a witness of $\varphi_1 \mathcal{U}_{(a,b)} \varphi_2$.

Similarly, in the case of the fifth equivalence, the left-to-right direction holds because if $(w,t) \vDash \varphi_1 \mathcal{U}_{(c,\infty)} \varphi_2$ with witness $t_1 \in (t+c,\infty)$, then for every $t_2 \in (t, t+c]$ we have $t < t_2 \leq t+c < t_1$, whence $(w,t_2) \vDash \varphi_2$ and $(w,t_2) \vDash \varphi_1 \mathcal{U} \varphi_2$. Therefore $(w,t) \vDash \Box_{(0,c]}(\varphi_1 \wedge \varphi_1 \mathcal{U} \varphi_2)$. For the right-to-left implication we assume $(w,t) \vDash \Box_{(0,c]}(\varphi_1 \wedge \varphi_1 \mathcal{U} \varphi_2)$. This means that $\varphi_1 \wedge \varphi_1 \mathcal{U} \varphi_2$ holds along $(t, t+c]$. In particular, $(w, t+c) \vDash \varphi_1 \mathcal{U} \varphi_2$, whence there is $t_1 \in (t+c,\infty)$ where φ_2 holds and φ_1 holds along $(t+c, t_1)$. Then φ_1 holds along (t, t_1) and therefore t_1 is a witness for $(w,t) \vDash \varphi_1 \mathcal{U}_{(c,\infty)} \varphi_2$. □

A way to express the timed *since* by a combination of untimed *since* and bounded *historically* is shown in the following lemma.

Lemma 3.2.4. *For any rational numbers a, b, c such that $0 \leq a < b < \infty$ and $0 < c < \infty$, and formulas φ_1, φ_2, we have*

$$\varphi_1 \mathcal{S}_{(a,b)} \varphi_2 \equiv \varphi_1 \mathcal{S}_{(a,\infty)} \varphi_2 \wedge \blacklozenge_{(a,b)} \varphi_2$$
$$\varphi_1 \mathcal{S}_{(a,b]} \varphi_2 \equiv \varphi_1 \mathcal{S}_{(a,\infty)} \varphi_2 \wedge \blacklozenge_{(a,b]} \varphi_2 \qquad \varphi_1 \mathcal{S}_{(c,\infty)} \varphi_2 \equiv \blacksquare_{(0,c]}(\varphi_1 \wedge \varphi_1 \mathcal{S} \varphi_2)$$
$$\varphi_1 \mathcal{S}_{[a,b)} \varphi_2 \equiv \varphi_1 \mathcal{S}_{[a,\infty)} \varphi_2 \wedge \blacklozenge_{[a,b)} \varphi_2 \qquad \varphi_1 \mathcal{S}_{[c,\infty)} \varphi_2 \equiv \blacksquare_{(0,c)} \varphi_1 \wedge \blacksquare_{(0,c]}(\varphi_1 \underline{\mathcal{S}} \varphi_2)$$
$$\varphi_1 \mathcal{S}_{[a,b]} \varphi_2 \equiv \varphi_1 \mathcal{S}_{(a,\infty)} \varphi_2 \wedge \blacklozenge_{[a,b]} \varphi_2$$

Proof. Symmetric to Lemma 3.2.3. □

In terms of complexity, the increase in the size under these equivalences is of an additive constant. Consider, for example, the equivalence $\varphi_1 \mathcal{U}_{(a,b)} \varphi_2 \equiv \varphi_1 \mathcal{U}_{(a,\infty)} \varphi_2 \wedge \Diamond_{(a,b)} \varphi_2$. We observe that $s(\varphi_1 \mathcal{U}_{(a,\infty)} \varphi_2 \wedge \Diamond_{(a,b)} \varphi_2) = s(\varphi_1 \mathcal{U}_{(a,b)} \varphi_2) + 2$. Now take the equivalence $\varphi_1 \mathcal{S}_{(c,\infty)} \varphi_2 \equiv \blacksquare_{(0,c]}(\varphi_1 \wedge \varphi_1 \mathcal{S} \varphi_2)$. Then $s(\blacksquare_{(0,c]}(\varphi_1 \wedge \varphi_1 \mathcal{S} \varphi_2)) = s(\varphi_1 \mathcal{S}_{(c,\infty)} \varphi_2) + 2$. Other cases are clearly similar. Observe that in both lemmas the resolution of the formula in the right-hand side of the equivalences is not affected by any of these rewritings.

The following lemmas cover the eliminations from step 3, showing that we are able to move bounded intervals in the subscripts of $\Diamond_I, \blacklozenge_I$, towards the origin, an amount equal to its length or to its distance to the origin.

Lemma 3.2.5. *For any rational numbers a, b such that $0 < a < b$ and $a \geq b - a$ and every formula φ, if we put $c = b - a$, then*

$$\Diamond_{(a+c,b+c)} \varphi \equiv \Diamond_{(0,c)} \Box_{(0,c)} \Diamond_{(a,b)} \varphi \qquad \Diamond_{[a+c,b+c)} \varphi \equiv \Diamond_{[0,c)} \Box_{(0,c]} \Diamond_{[a,b)} \varphi$$
$$\Diamond_{(a+c,b+c]} \varphi \equiv \Diamond_{(0,c]} \Box_{[0,c)} \Diamond_{(a,b]} \varphi \qquad \Diamond_{[a+c,b+c]} \varphi \equiv \Diamond_{[0,c]} \Box_{[0,c]} \Diamond_{[a,b]} \varphi$$

Proof. All proofs are similar, so we only show the first equivalence.

First, assume that $(w,t) \vDash \Diamond_{(a+c,b+c)} \varphi$. This means there is some $t' \in (t+a+c, t+b+c)$ such that $(w,t') \vDash \varphi$. Then, for every $t'' \in (t'-b, t'-a)$, we have that $t' \in (t''+a, t''+b)$, whence $(w,t'') \vDash \Diamond_{(a,b)} \varphi$. Since $|(t'-b, t'-a)| = c$, then $(w, t'-b) \vDash \Box_{(0,c)} \Diamond_{(a,b)} \varphi$. But $t' - b \in (t, t+c)$. Hence, we conclude that $(w,t) \vDash \Diamond_{(0,c)} \Box_{(0,c)} \Diamond_{(a,b)} \varphi$.

The converse is proven by contraposition. Assume that φ is false over $(t + a + c, t + b + c)$. Then $\Diamond_{(a,b)} \varphi$ is false at time $t + c$, and $\Box_{(0,c)} \Diamond_{(a,b)} \varphi$ is false over $(t, t+c)$, so that the right-hand side of the equivalence is false at time t. □

Lemma 3.2.6. *For any rational numbers a, b such that $0 < a < b$ and $a > b - a$ and every formula φ, if we put $c = b - a$, then*

$$\blacklozenge_{(a+c,b+c)}\varphi \equiv \blacklozenge_{(0,c)}\blacksquare_{(0,c)}\blacklozenge_{(a,b)}\varphi \qquad \blacklozenge_{[a+c,b+c)}\varphi \equiv \blacklozenge_{[0,c)}\blacksquare_{(0,c]}\blacklozenge_{[a,b)}\varphi$$
$$\blacklozenge_{(a+c,b+c]}\varphi \equiv \blacklozenge_{(0,c]}\blacksquare_{[0,c)}\blacklozenge_{(a,b]}\varphi \qquad \blacklozenge_{[a+c,b+c]}\varphi \equiv \blacklozenge_{[0,c]}\blacksquare_{[0,c]}\blacklozenge_{[a,b]}\varphi$$

Proof. Symmetric to Lemma 3.2.5. □

Lemma 3.2.7. *For any rational numbers a, b such that $0 < a < b$ and $a \leq b - a$ and every formula φ, we have*

$$\Diamond_{(a,b)}\varphi \equiv \Diamond_{(0,b-a)}\Box_{(0,a)}\Diamond_{(0,a)}\varphi \qquad \Diamond_{[a,b)}\varphi \equiv \Diamond_{[0,b-a)}\Box_{(0,a]}\Diamond_{[0,a)}\varphi$$
$$\Diamond_{(a,b]}\varphi \equiv \Diamond_{(0,b-a]}\Box_{[0,a)}\Diamond_{(0,a]}\varphi \qquad \Diamond_{[a,b]}\varphi \equiv \Diamond_{[0,b-a]}\Box_{[0,a]}\Diamond_{[0,a]}\varphi$$

Proof. We only show the first equivalence.

First, assume that $(w, t) \vDash \Diamond_{(a,b)}\varphi$. This means there is some $t' \in (t+a, t+b)$ such that $(w, t') \vDash \varphi$. Then, for every $t'' \in (t' - a, t')$, we have that $t' \in (t'', t'' + a)$, whence $(w, t'') \vDash \Diamond_{(0,a)}\varphi$. Since $|(t' - a, t')| = a$, then $(w, t' - a) \vDash \Box_{(0,a)}\Diamond_{(0,a)}\varphi$. But $t' - a \in (t, t + b - a)$. Hence, we conclude that $(w, t) \vDash \Diamond_{(0,b-a)}\Box_{(0,a)}\Diamond_{(0,a)}\varphi$.

The converse is proven by contraposition. Assume that φ is false over $(t + a, t + b)$. Then $\Diamond_{(0,b-a)}\varphi$ is false at time $t + a$, and $\Box_{(0,a)}\Diamond_{(0,b-a)}\varphi$ is false over $(t, t + a)$, so that the right-hand side of the equivalence is false at time t. □

Lemma 3.2.8. *For any rational numbers a, b such that $0 < a < b$ and $a \leq b - a$ and every formula φ, we have*

$$\blacklozenge_{(a,b)}\varphi \equiv \blacklozenge_{(0,b-a)}\blacksquare_{(0,a)}\blacklozenge_{(0,a)}\varphi \qquad \blacklozenge_{[a,b)}\varphi \equiv \blacklozenge_{[0,b-a)}\blacksquare_{(0,a]}\blacklozenge_{[0,a)}\varphi$$
$$\blacklozenge_{(a,b]}\varphi \equiv \blacklozenge_{(0,b-a]}\blacksquare_{[0,a)}\blacklozenge_{(0,a]}\varphi \qquad \blacklozenge_{[a,b]}\varphi \equiv \blacklozenge_{[0,b-a]}\blacksquare_{[0,a]}\blacklozenge_{[0,a]}\varphi$$

Proof. Symmetric to Lemma 3.2.7. □

Let us explain how these lemmas work with an example. Consider a formula of the form $\Diamond_{(5,6)}\varphi$ where we want to eliminate the occurrence of $\Diamond_{(5,6)}$. This operator will be replaced by a sequence of $\Diamond_I\Box_J$ where $\inf I = \inf J = 0$ to obtain an equivalent formula. We perform the division $\dfrac{5}{6-5}$ and obtain the quotient 5 and residuum 0. Then, we apply 5 iterations of Lemma 3.2.5 to obtain $\Diamond_{(5,6)}\varphi \equiv \Diamond_{(0,1)}\Box_{(0,1)}\Diamond_{(4,5)}\varphi \equiv \Diamond_{(0,1)}\Box_{(0,1)}\Diamond_{(0,1)}\Box_{(0,1)}\Diamond_{(3,4)}\varphi \equiv \cdots \equiv (\Diamond_{(0,1)}\Box_{(0,1)})^5\varphi$. Now consider a different formula $\Diamond_{(5,8)}\varphi$. We perform the division $\dfrac{5}{8-5}$ and obtain the quotient 1 and positive residuum. In this case, we apply one iteration of Lemma 3.2.5 to obtain $\Diamond_{(5,8)}\varphi \equiv \Diamond_{(0,3)}\Box_{(0,3)}\Diamond_{(2,5)}\varphi$ and one of Lemma 3.2.7 to obtain $\Diamond_{(2,5)}\varphi \equiv \Diamond_{(0,3)}\Box_{(0,2)}\Diamond_{(0,2)}\varphi$. Joining the two expresions, we obtain $\Diamond_{(5,8)}\varphi \equiv \Diamond_{(0,3)}\Box_{(0,3)}\Diamond_{(0,3)}\Box_{(0,2)}\Diamond_{(0,2)}\varphi$. Thus, the previous lemmas can be joined to obtain compact expressions.

Lemma 3.2.9. *For every formula φ and rational numbers $0 < a < b < \infty$, we have $\Diamond_I\varphi \equiv \varphi'$ where φ' is*

1. $\left(\Diamond_{(0,c)}\Box_{(0,c)}\right)^{\lfloor \frac{a}{c} \rfloor}\left(\Diamond_{(0,c)}\Box_{(0,a)}\right)^{\lceil \frac{r}{c} \rceil}\Diamond_{(0,a)}\varphi$ *if* $I = (a,b)$,
2. $\left(\Diamond_{[0,c)}\Box_{(0,c]}\right)^{\lfloor \frac{a}{c} \rfloor}\left(\Diamond_{[0,c)}\Box_{(0,a]}\right)^{\lceil \frac{r}{c} \rceil}\Diamond_{[0,a)}\varphi$ *if* $I = [a,b)$,
3. $\left(\Diamond_{(0,c]}\Box_{[0,c)}\right)^{\lfloor \frac{a}{c} \rfloor}\left(\Diamond_{(0,c]}\Box_{[0,a)}\right)^{\lceil \frac{r}{c} \rceil}\Diamond_{(0,a]}\varphi$ *if* $I = (a,b]$,
4. $\left(\Diamond_{[0,c]}\Box_{[0,c]}\right)^{\lfloor \frac{a}{c} \rfloor}\left(\Diamond_{[0,c]}\Box_{[0,a]}\right)^{\lceil \frac{r}{c} \rceil}\Diamond_{(0,a)}\varphi$ *if* $I = [a,b]$.

where, $c = b - a$, *and* $r = a - \lfloor \frac{a}{c} \rfloor \cdot a$. *Moreover,* $s(\varphi') = s(\varphi) + \bar{r}(I)$ *and* $r(\varphi') = r(\varphi)$.

Proof. Observe that $a = c \cdot \lfloor \frac{a}{c} \rfloor + r$, this is, $\lfloor \frac{a}{c} \rfloor$ is the quotient of the integer division $\frac{a}{b-a}$ and $\lceil \frac{r}{c} \rceil$ is 1 if is the residuum is positive or 0 if the division $\frac{a}{b-a}$ is exact. To obtain the expressions above, we apply Lemma 3.2.5 $\lfloor \frac{a}{c} \rfloor$ times and Lemma 3.2.7 only one time if $r > 0$ i.e. $\lceil \frac{r}{c} \rceil$ times. Moreover, the size of φ' is $2 \cdot \lfloor \frac{a}{c} \rfloor + 2 \cdot \lceil \frac{r}{c} \rceil + 1 + s(\varphi) = 2 \cdot \lceil \frac{a}{c} \rceil + 1 + s(\varphi) = \bar{r}(I) + s(\varphi)$ and the resolution of φ' is equal to the resolution of φ because the new operators that appear in φ' have subscripts whose infimum is zero. □

Lemma 3.2.10. *For every formula* φ *and rational numbers* $0 < a < b < \infty$, *we have* $\blacklozenge_I\varphi \equiv \varphi'$ *where* φ' *is*

1. $\left(\blacklozenge_{(0,c)}\blacksquare_{(0,c)}\right)^{\lfloor \frac{a}{c} \rfloor}\left(\blacklozenge_{(0,c)}\blacksquare_{(0,a)}\right)^{\lceil \frac{r}{c} \rceil}\blacklozenge_{(0,a)}\varphi$ *if* $I = (a,b)$,
2. $\left(\blacklozenge_{[0,c)}\blacksquare_{(0,c]}\right)^{\lfloor \frac{a}{c} \rfloor}\left(\blacklozenge_{[0,c)}\blacksquare_{(0,a]}\right)^{\lceil \frac{r}{c} \rceil}\blacklozenge_{[0,a)}\varphi$ *if* $I = [a,b)$,
3. $\left(\blacklozenge_{(0,c]}\blacksquare_{[0,c)}\right)^{\lfloor \frac{a}{c} \rfloor}\left(\blacklozenge_{(0,c]}\blacksquare_{[0,a)}\right)^{\lceil \frac{r}{c} \rceil}\blacklozenge_{(0,a]}\varphi$ *if* $I = (a,b]$,
4. $\left(\blacklozenge_{[0,c]}\blacksquare_{[0,c]}\right)^{\lfloor \frac{a}{c} \rfloor}\left(\blacklozenge_{[0,c]}\blacksquare_{[0,a]}\right)^{\lceil \frac{r}{c} \rceil}\blacklozenge_{(0,a)}\varphi$ *if* $I = [a,b]$.

where, $c = b - a$, *and* $r = a - \lfloor \frac{a}{c} \rfloor \cdot a$. *Moreover,* $s(\varphi') = s(\varphi) + \bar{r}(I)$ *and* $r(\varphi') = r(\varphi)$.

Proof. Symmetric using Lemmas 3.2.6 and 3.2.8. □

Lastly, we arrive to the lemma that allows to perform step 4, eliminating subscripts I of the form $(0,a], [0,a), [0,a]$ from $\Diamond_I$ using the following equivalences.

Lemma 3.2.11. *For every formula* φ,

$$\Diamond_{(0,a]}\varphi \equiv \Diamond_{(0,a)}\varphi \vee (\bigcirc\Diamond_{(0,a)}\varphi \wedge \neg\varphi\mathcal{U}\varphi)$$
$$\Diamond_{[0,a)}\varphi \equiv \varphi \vee \Diamond_{(0,a)}\varphi$$
$$\Diamond_{[0,a]}\varphi \equiv \varphi \vee \Diamond_{(0,a]}\varphi$$
$$\blacklozenge_{(0,a]}\varphi \equiv \blacklozenge_{(0,a)}\varphi \vee (\bigcirc\blacklozenge_{(0,a)}\varphi \wedge \neg\varphi\mathcal{U}\varphi)$$
$$\blacklozenge_{[0,a)}\varphi \equiv \varphi \vee \blacklozenge_{(0,a)}\varphi$$
$$\blacklozenge_{[0,a]}\varphi \equiv \varphi \vee \blacklozenge_{(0,a]}\varphi$$

Proof. We show the first equivalence: the second and third are easy, and the rest are proven symetrically. First, assume that $(w,t) \vDash \Diamond_{(0,a]}\varphi$. Then φ holds at some $t' \in (t, t+a]$. If $t' \in (t, t+a)$, then $(w,t) \vDash \Diamond_{(0,a)}\varphi$, and if $t' = t+a$, then $(w,t) \vDash (\bigcirc\Diamond_{(0,a)}\varphi \wedge \neg\varphi\mathcal{U}\varphi)$ as for every $t'' \in (t, t+a)$, $(w,t'') \vDash \Diamond_{(0,a)}\varphi$ and $(w,t) \vDash \neg\varphi\mathcal{U}\varphi$ as φ is true at $t+a$ and false between t and $t+a$. For the converse implication, assume $(w,t) \vDash \Diamond_{(0,a)}\varphi \vee (\bigcirc\Diamond_{(0,a)}\varphi \wedge \neg\varphi\mathcal{U}\varphi)$. If $(w,t) \vDash \Diamond_{(0,a)}\varphi$ then clearly $(w,t) \vDash \Diamond_{(0,a]}\varphi$. Otherwise $(\bigcirc\Diamond_{(0,a)}\varphi \wedge \neg\varphi\mathcal{U}\varphi)$ holds at t. Then we know there is some first $t' > t$ where φ holds and φ is false between t and t'. We want to show that there is some $t' \in (t, t+a]$ where φ holds so we assume $t' > t+a$ to get a contradiction. But on the other hand $\Diamond_{(0,a)}\varphi$ holds immediately after t, whence there is some $t'' > t$ such that $t''+a < t'$ and then φ is true at some point between t'' and $t''+a$, contradicting that φ is false along (t,t'). □

Observe that the size of after the rewritings above increases by an additive constant. For example, the size of $\Diamond_{(0,a)}\varphi \vee (\bigcirc\Diamond_{(0,a)}\varphi \wedge \neg\varphi\mathcal{U}\varphi)$ equals to the size of $\Diamond_{(0,a]}\varphi$+5, and the size of $\varphi \vee \Diamond_{(0,a)}\varphi$ is the size of $\Diamond_{[0,a)}\varphi$+1. On the other hand, the resolution after the rewriting remains the same.

Theorem 3.2.2. *Any MITL formula φ is equivalent to some formula φ' that only contains the temporal operators $\mathcal{U}, \mathcal{S}, \Diamond_{(0,a)}, \blacklozenge_{(0,a)}$, where a is a positive rational number. Moreover, $s(\varphi') \in O[s(\varphi) \cdot r(\varphi)]$ and $r(\varphi') \leq 1$.*

Proof. Let φ be an arbitrary formula. Let us denote by k the maximum additive constant of size increase in all previous lemmas. Then apply the following procedure:

1. For every occurrence of bounded $\mathcal{U}$ (resp. $\mathcal{S}$) in φ, apply one iteration of Lemmas 3.2.3 and 3.2.4 to write a new formula φ_1 using unbounded $\mathcal{U}$ (resp. $\mathcal{S}$) and bounded $\Diamond$ (resp. $\blacklozenge$). The number of occurrences of these operators is bounded by $s(\varphi)$. Moreover, after every application, the size of the formula grows by a constant additive factor and the resolution remains the same. Hence, $s(\varphi_1) = k \cdot s(\varphi)$ for some natural number k and $r(\varphi_1) = r(\varphi)$.

2. For every occurrence of unbounded $\mathcal{U}$ (resp. $\mathcal{S}$) in φ_1, apply one iteration of Lemmas 3.2.3 and 3.2.4 to write a new formula φ_2 using $\mathcal{U}_{(0,\infty)}$ (resp. $\mathcal{S}_{(0,\infty)}$ and bounded $\Diamond$ (resp. $\blacklozenge$). The number of occurrences of these operators is bounded by $s(\varphi_1)$. Moreover, after every application, the size of the formula grows by a constant additive factor and the resolution remains the same. Hence, $s(\varphi_2) = k \cdot s(\varphi_1)$ for some natural number k and $r(\varphi_2) = r(\varphi_1)$.

3. For every occurrence of bounded $\Diamond$ (resp. $\blacklozenge$) in φ_2, apply one iteration of Lemmas 3.2.5 and 3.2.7 to obtain a new formula φ_3 using $\Diamond_I$ (resp. $\blacklozenge_I$) where $\inf I = 0$. The number of occurrences of these operators is bounded by $s(\varphi)$. Moreover, after every application, the size of the formula grows by an additive factor bounded by $r(\varphi)$ and the resolution is reduced to 1

because once all bounded $\Diamond, \blacklozenge$ are eliminated, all operators have subscripts with infimum zero or unbounded. Hence, $s(\varphi_3) = r(\varphi) \cdot s(\varphi_2)$ for some natural number k and $r(\varphi_3) \leq 1$.

4. For every occurrence of $\Diamond_{(0,a]}, \Diamond_{[0,a)}, \Diamond_{[0,a]}$ (resp. $\blacklozenge$) in φ_3 use at most two iterations of Lemma 3.2.11 to obtain a formula φ_4 using $\Diamond_{(0,a)}$ (resp. $\blacklozenge_{(0,a)}$) and $\mathcal{U}_{(0,\infty)}$ (resp. $\mathcal{S}_{(0,\infty)}$). The number of occurrences of these operators is bounded by $s(\varphi_3)$. Moreover, after every application, the size of the formula grows by a constant additive factor and the resolution remains the same. Hence, $s(\varphi_4) = k \cdot s(\varphi_3)$ for some natural number k and $r(\varphi_4) = r(\varphi_3)$.

Then put $\varphi' = \varphi_4$. The formula φ' satisfies $\varphi \equiv \varphi'$ and only contains standard operators. Furthermore, the size of ψ' is bounded by $k^3 \cdot s(\varphi) \cdot r(\varphi)$ where k is the maximum additive constant of size increase among Lemmas 3.2.3, 3.2.4 and 3.2.11. Hence, $s(\varphi') \in O(s(\varphi) \cdot r(\varphi))$ Finally, $r(\varphi') \leq 1$. □

3.3 Timed signal-based transducers

In this section we define a variant of timed transducers that accept and output signals. To describe the computations of such automata with respect to a signal, we will be interested in decomposing signals into a concatenation of *point segments* $\dot{w} : \{0\} \to \Sigma$ and *open segments* of bounded length $w^r : (0, r) \to \Sigma$ for some $r > 0$ or infinite length $w^\infty : (0, +\infty) \to \Sigma$. We define several kinds of concatenation

- The *concatenation* of two functions $\dot{w}, w^r$ is defined as $\dot{w} \cdot w^r : [0, r) \to \Sigma$ such that $(\dot{w} \cdot w^r)(0) = \dot{w}(0)$ and $(\dot{w} \cdot w^r)(t) = w^r(t)$ for $t \in (0, r)$.

- The *concatenation* of two functions $\dot{w}, w^\infty$ is defined as $\dot{w} \cdot w^\infty : [0, +\infty) \to \Sigma$ such that $(\dot{w} \cdot w^\infty)(0) = \dot{w}(0)$ and $(\dot{w} \cdot w^\infty)(t) = w^\infty(t)$ for $t \in (0, +\infty)$.

- The *concatenation* of two functions $w_1 : [0, r_1) \to \Sigma$ and $w_2 : [0, r_2) \to \Sigma$ for $r_1, r_2 > 0$ is defined as $w_1 \cdot w_2 : [0, r_1 + r_2) \to \Sigma$ such that $(w_1 \cdot w_2)(t) = w_1(t)$ if $t \in [0, r_1)$ and $(w_1 \cdot w_2)(t) = w_2(t - r_1)$ if $t \in [r_1, r_1 + r_2)$.

- The *concatenation* of two functions $w_1 : [0, r) \to \Sigma$ and $w_2 : [0, +\infty) \to \Sigma$ for $r > 0$ is defined as $w_1 \cdot w_2 : [0, +\infty) \to \Sigma$ such that $(w_1 \cdot w_2)(t) = w_1(t)$ if $t \in [0, r)$ and $(w_1 \cdot w_2)(t) = w_2(t - r)$ if $t \in [r, +\infty)$.

Let w be a signal w and $\mathcal{I} = \{0\}(0, t_0)\{t_0\}(t_0, t_1)\{t_1\} \cdots$ a point-segment partition witnessing this. Then if $\mathcal{I}$ is infinite, for every $i \geq 0$ there exist point and open segments $\dot{w}_i, w_i^{t_i - t_{i-1}}$ satisfying $\dot{w}_i(0) = w(t_i)$ and $w_i^{t_i - t_{i-1}}(t) = w(t + t_0 + \cdots + t_{i-1})$ for all $t \in (0, t_i - t_{i-1})$. If $\mathcal{I}$ is finite, let $(t_M, +\infty)$ be the last open segment. Then for every $0 \leq i < M$ there exist point and open segments $\dot{w}_i, w_i^{t_i - t_{i-1}}$ as above, and for $i = M$ there exist $\dot{w}_M, w_M^\infty$ such that

$\dot{w}_M(0) = w(t_M)$ and $w_M^\infty(t) = w(t + t_0 + t_1 + \cdots + t_M)$ for all $t \in (0, +\infty)$. We define

$$\begin{aligned} w_0 &= (\dot{w}_0 \cdot w_0^{t_0}) \\ w_1 &= (\dot{w}_0 \cdot w_0^{t_0}) \cdot (\dot{w}_1 \cdot w_1^{t_1 - t_0}) \\ &\vdots \\ w_i &= (\dot{w}_0 \cdot w_0^{t_0}) \cdot (\dot{w}_1 \cdot w_1^{t_1 - t_0}) \cdots (\dot{w}_i \cdot w_i^{t_i - t_{i-1}}) \\ &\vdots \end{aligned}$$

and then $w = \bigcup_{i \geq 0} w_i$. We will usually describe the decomposition of a signal w with respect to a point-segment partition $\mathcal{I} = \{0\}(0, t_0)\{t_0\}(t_0, t_1)\{t_1\} \cdots$ as

$$w = (\dot{w}_0 \cdot w_0^{t_0}) \cdot (\dot{w}_1 \cdot w_1^{t_1 - t_0}) \cdot (\dot{w}_2 \cdot w_2^{t_2 - t_1}) \cdots,$$

or similarly, for $r_0 = t_0$ and $r_i = t_i - t_{i-1}$, as

$$w = (\dot{w}_0 \cdot w_0^{r_0}) \cdot (\dot{w}_1 \cdot w_1^{r_1}) \cdot (\dot{w}_2 \cdot w_2^{r_2}) \cdots.$$

Definition 3.3.1. A *timed transducer* is a tuple $\mathcal{T} = (S, \underline{s}, C, P, Q, \Delta, \lambda, \gamma, Inv, \mathcal{F})$ where

- S is a finite set of *states* and $\underline{s} \notin S$ is the *initial state*,
- P is a finite set of *input variables*,
- Q is a finite set of *output variables*,
- C is a finite set of *clock variables*,
- Inv is the *invariance* map $Inv : S \to \Phi(C)$ assigning some clock constraint to every state,
- Δ is the *transition relation* consisting of elements of the form $\delta = (s, g, R, s')$ where $s \in S \cup \{\underline{s}\}$ and $s' \in S$, the guard $g \in \Phi(C)$, and the reset instruction $R \subseteq C$,
- λ is a map $\lambda : S \cup \Delta \to BC(P)$ called the *input labeling*,
- γ is a map $\gamma : S \cup \Delta \to BC(Q)$ called the *output labeling*,
- $\mathcal{F} \subseteq \mathcal{P}(S \cup \Delta)$ is a *generalized Büchi acceptance condition.*

The *configuration* of a timed transducer is determined by a pair (s, v) where s is a state and v is a clock valuation. Informally, a run of the timed transducer is an alternating sequence of time and discrete transitions. The transducer can either stay in a state for some time, provided that the invariance condition holds, or take a transition to a different state whose guard is satisfied. Moreover, during a time step of duration r in a state s, the transducer reads an open segment

w^r of the input w of length r, whose values are required to satisfy $\lambda(s)$, and outputs an open fragment u^r of the output u according to $\gamma(s)$. While taking a transition δ, a point segment $\dot{w}$ from w is read, whose value must satisfy $\lambda(\delta)$, and then a point segment $\dot{u}$ of the output u is written according to $\gamma(s)$.

Definition 3.3.2. For a timed transducer $\mathcal{T}$ and a given signal w, two kinds of relations among configurations of $\mathcal{T}$ are defined

- A *time transition* is a tuple $((s,v), w^r, u^r, (s, v+r))$, denoted by

$$(s,v) \xrightarrow{w^r/u^r} (s, v+r)$$

  where w^r, u^r are open segments of length r of w such that $(w^r, t) \vDash \lambda(s)$, $(u^r, t) \vDash \gamma(s)$, and $v+t$ satisfies $Inv(s)$ for all $t \in (0, r)$.

- A *discrete transition* is a tuple $((s,v), \dot{w}, \dot{u}, (s', R[v]))$, denoted by

$$(s,v) \xrightarrow{\dot{w}/\dot{u}} (s', R[v]),$$

  for some transition $\delta = (s, g, R, s') \in \Delta$ such that $(\dot{w}, 0) \vDash \lambda(\delta)$, and $(\dot{u}, 0) \vDash \gamma(\delta)$, and v satisfies g.

Thus, a *run* of the automaton starting at configuration $(\underline{s}, v)$ for an arbitrary clock valuation v over the signal w is a finite or infinite alternating sequence of time and discrete transitions

$$(\underline{s}, v) \xrightarrow{\dot{w}_0/\dot{u}_0} (s_0, v_0) \xrightarrow{w_0^{r_0}/u_0^{r_0}} (s_0, v_0 + r_0) \xrightarrow{\dot{w}_1/\dot{u}_1} (s_1, v_1) \to \cdots,$$

such that w can be decomposed as $\dot{w}_0 \cdot w_0^{r_0} \cdot \dot{w}_1 \cdot w_1^{r_1} \cdots$. Then w is called the *input signal*, and $u = \dot{u}_0 \cdot u_0^{r_0} \cdot \dot{u}_1 \cdot u_1^{r_1} \cdots$ is the *output signal*. We say that a run r over an input signal w *induces* an output signal u. In particular, a *finite run* over w is of the form

$$(\underline{s}, v) \xrightarrow{\dot{w}_0/\dot{u}_0} (s_0, v_0) \xrightarrow{w_0^{r_0}/u_0^{r_0}} (s_0, v_0 + r_0) \to \cdots \xrightarrow{\dot{w}_i/\dot{u}_i} (s, v)^{w_i^\infty/u_i^\infty},$$

where $w = \dot{w}_0 \cdot w_0^{r_0} \cdots w_{i-1}^{r_{i-1}} \cdot \dot{w}_i \cdot w_i^\infty$, inducing the output signal $u = \dot{u}_0 \cdot u_0^{r_0} \cdots u_{i-1}^{r_{i-1}} \cdot \dot{u}_i \cdot u_i^\infty$ such that $w_i^\infty \vDash \lambda(s)$, $u_i^\infty \vDash \gamma(s)$, and $v+t$ satisfies $Inv(s)$ for all $t > 0$.

To introduce the notion of accepting run, we explain how external time operates in the context of a run of a timed transducer.

Definition 3.3.3. Given an infinite run of a timed transducer $\mathcal{T}$ over a signal w of the form

$$(\underline{s}, v) \xrightarrow{\dot{w}_0/\dot{u}_0} (s_0, v_0) \xrightarrow{w_0^{r_0}/u_0^{r_0}} (s_0, v_0 + r_0) \xrightarrow{\dot{w}_1/\dot{u}_1} (s_1, v_1) \to \cdots,$$

and a point of time $t \geq 0$, we say that $\mathcal{T}$ is at state $\underline{s}$ if $t = 0$, and at state s_j at time $t > 0$ iff $\sum_{k=0}^{j} r_k \leq t < \sum_{k=0}^{j+1} r_k$. Further, we say that $\mathcal{T}$ is at transition δ_0 if $t = 0$ and at transition δ_j iff $t = \sum_{k=0}^{j} r_k$.

Similarly, given a finite run of a timed transducer $\mathcal{T}$ over a signal w of the form

$$(\underline{s}, v) \xrightarrow{\dot{w}_0/\dot{u}_0} (s_0, v_0) \xrightarrow{w_0^{r_0}/u_0^{r_0}} (s_0, v_0 + r_0) \to \cdots \xrightarrow{\dot{w}_i/\dot{u}_i} (s_i, v_i)^{w_i^\infty/u_i^\infty},$$

we say that the automaton $\mathcal{T}$ is at state s_i at time t iff $\sum_{k=0}^{i-1} r_k \leq t$.

Definition 3.3.4. A a run of $\mathcal{T}$ over the input signal w is *accepting* if it starts at the initial configuration $(\underline{s}, 0)$ and satisfies the *generalized Büchi condition*: for all $F \in \mathcal{F}$ and $t \geq 0$ there exists $t' > t$ and a state or transition $\alpha \in F$ such that $\mathcal{T}$ is at α at time t'. Hence, u is a $\mathcal{T}$*-output* over w iff there exists an accepting run over w of inducing u.

For a timed transducer $\mathcal{T}$, we say $\mathcal{T}$ is *Q-functional* iff for every signal w there exists a $\mathcal{T}$*-output* over w which is unique with respect to the assignments of the variables in Q. In this case, we denote by $\mathcal{T}$ the map associated to it, and we say that $u = \mathcal{T}(w)$ is *the* output of the transducer over the signal w.

3.4 Composing timed signal-based transducers

We present the main way of composing timed transducers.

Definition 3.4.1. Let $\mathcal{T}_1, \mathcal{T}_2$ be two timed transducers whose components are $(S_i, \underline{s}_i, P_i, Q_i, C_i, Inv_i, \Delta_i, \lambda_i, \gamma_i, \mathcal{F}_i)$ for $i = 1, 2$. Assume that $C_1 \cap C_2 = P_1 \cap P_2 = Q_1 \cap Q_2 = \emptyset$. The *parallel composition* of $\mathcal{T}_1$ and $\mathcal{T}_2$, denoted by $\mathcal{T}_1 || \mathcal{T}_2$, is the timed transducer $(S, \underline{s}, P, Q, X, Inv, \Delta, \lambda, \gamma, \mathcal{F})$ such that

- $S = S_1 \times S_2$
- $\underline{s} = (\underline{s}_1, \underline{s}_2)$
- $P = P_1 \cup P_2$
- $Q = Q_1 \cup Q_2$
- $C = C_1 \cup C_2$
- $Inv((s_1, s_2)) = Inv_1(s_1) \wedge Inv_2(s_2)$
- The transition relation Δ consists of three kinds of transitions:
  - *Simultaneous transitions* $\delta = ((s_1, s_2), g, R, (s_1', s_2'))$, where $\delta_1 = (s_1, g_1, R_1, s_1') \in \Delta_1$, $\delta_2 = (s_2, g_2, R_2, s_2') \in \Delta_2$, $g = g_1 \wedge g_2$ and $R = R_1 \cup R_2$. We say that δ is *built* from δ_1 and δ_2.
  - *Left-sided transitions* $\delta = ((s_1, s_2), g \wedge Inv_2(s_2), R, (s_1', s_2))$ where $\delta_1 = (s_1, g, R, s_1') \in \Delta_1$. We say that δ is *built* from δ_1.
  - *Right-sided transitions* $\delta = ((s_1, s_2), Inv_1(s_1) \wedge g, R, (s_1, s_2'))$ where $\delta = (s_2, g, R, s_2') \in \Delta_2$. We say that δ is *built* from δ_2.

- The input labeling $\lambda : S \cup \Delta \to BC(P)$ is defined similarly as λ,
  - for a state (s_1, s_2), put $\lambda((s_1, s_2)) = \lambda_1(s_1) \wedge \lambda_2(s_2)$
  - for a simultaneous transition $\delta = ((s_1, s_2), g, R, (s_1', s_2'))$, put
    $$\lambda(\delta) = \lambda_1((s_1, g_1, R_1, s_1')) \wedge \lambda_2((s_2, g_2, R_2, s_2'))$$
  - for a left-sided transition $\delta = ((s_1, s_2), g \wedge Inv_2(s_2), R, (s_1', s_2))$, put
    $$\lambda(\delta) = \lambda_1((s_1, g, R, s_1')) \wedge \lambda_2(s_2)$$
  - for a right-sided transition $\delta = ((s_1, s_2), Inv_1(s_1) \wedge g, R, (s_1, s_2'))$ put
    $$\lambda(\delta) = \lambda_1(s_1) \wedge \lambda_2(s_2, g, R, s_2')$$
- The output labeling $\gamma : S \cup \Delta \to BC(Q)$ is defined as follows
  - for a state (s_1, s_2), put $\gamma((s_1, s_2)) = \gamma_1(s_1) \wedge \gamma_2(s_2)$
  - for a simultaneous transition $\delta = ((s_1, s_2), g, R, (s_1', s_2'))$, put
    $$\gamma(\delta) = \gamma_1((s_1, g_1, R_1, s_1')) \wedge \gamma_2((s_2, g_2, R_2, s_2'))$$
  - for a left-sided transition $\delta = ((s_1, s_2), g \wedge Inv_2(s_2), R, (s_1', s_2))$, put
    $$\gamma(\delta) = \gamma_1((s_1, g, R, s_1')) \wedge \gamma_2(s_2)$$
  - for a right-sided transition $\delta = ((s_1, s_2), Inv_1(s_1) \wedge g, R, (s_1, s_2'))$ put
    $$\gamma(\delta) = \gamma_1(s_1) \wedge \gamma_2(s_2, g, R, s_2')$$
- The generalized Büchi condition $\mathcal{F}$ contains the following sets:
  - For every $F_2 \in \mathcal{F}_2$, it contains the set $S_1 \times (F_2 \cap S_2) \cup \{\delta \in \Delta : \delta \text{ is simultaneous or right-sided built from a transition } \delta_2 \in F_2\}$.
  - For every $F_1 \in \mathcal{F}_1$, it contains the set $(F_1 \cap S_1) \times S_2 \cup \{\delta \in \Delta : \delta \text{ is simultaneous or left-sided built from a transition } \delta_1 \in F_1\}$.

In the following lemmas we describe the relation among the runs of $\mathcal{T}_1, \mathcal{T}_2$ and the runs of $\mathcal{T}_1 || \mathcal{T}_2$.

Definition 3.4.2. A *partial run* r_i with duration t over a signal w is a finite sequence of alternating time and discrete steps

$$(\underline{s}, v) \xrightarrow{\dot{w}_0/\dot{u}_0} (s_0, v_0) \xrightarrow{w_0^{r_0}/u_0^{r_0}} (s_0, v_0 + r_0) \xrightarrow{\dot{w}_1/\dot{u}_1} \cdots \xrightarrow{\dot{w}_i/\dot{u}_i} (s_i, v_i),$$

whose last step is a discrete transition, such that $t = r_0 + r_1 + \cdots + r_{i-1}$ and $w_{|[0,t]} = \dot{w}_0 \cdot w_0^{r_0} \cdots \dot{w}_i$.

Lemma 3.4.1. *Let $\mathcal{T}_1, \mathcal{T}_2$ be two timed transducers with input and output variables P_1, P_2 and Q_1, Q_2 respectively. Then for every signal w over $P_1 \cup P_2$, if r_1 is a run of $\mathcal{T}_1$ over $w_{|P_1}$ inducing u_1 and r_2 is a run of $\mathcal{T}_2$ over $w_{|P_2}$ inducing u_2, then there exists a run r of $\mathcal{T}_1||\mathcal{T}_2$ over w inducing $u_1||u_2$. If both r_1, r_2 are accepting, then r is accepting.*

Proof. Let $\mathcal{T}_1, \mathcal{T}_2$ be arbitrary and let w be a signal over $P_1 \cup P_2$. Let r_1, r_2 be runs of $\mathcal{T}_1, \mathcal{T}_2$ over w. We suppose that both runs are infinite; other cases are similar and left to the reader. We construct the run r of $\mathcal{T}_1||\mathcal{T}_2$ inductively. Since $C_1 \cap C_2 = \emptyset$, for any valuations v_1, v_2 over C_1, C_2 respectively, the evaluation $v_1 \cup v_2$ over C is well-defined.

Suppose the run r_1 starts at configuration $(\underline{s}_1, \underline{v}_1)$ and then takes a transition of the form $(\underline{s}_1, \underline{v}_1) \xrightarrow{\dot{w}_0/\dot{u}_0^1} (s_0^1, v_0^1)$ and the run r_2 starts at configuration $(\underline{s}_2, \underline{v}_2)$ and then takes a transition of the form $(\underline{s}_2, \underline{v}_2) \xrightarrow{\dot{w}_0/\dot{u}_0^2} (s_0^2, v_0^2)$. Thus, define the partial run r_0 (of duration 0) as $((\underline{s}_1, \underline{s}_2), \underline{v}_1 \cup \underline{v}_2) \xrightarrow{\dot{w}_0/\dot{u}_0^1||\dot{u}_0^2} ((s_0^1, s_0^2), v_0^1 \cup v_0^2)$.

Now assume that we have constructed the partial run r_i

$$((\underline{s}_1, \underline{s}_2), \underline{v}_1 \cup \underline{v}_2) \xrightarrow{\dot{w}_0/\dot{u}_0^1||\dot{u}_0^2} ((s_0^1, s_0^2), v_0^1 \cup v_0^2) \to \cdots \xrightarrow{\dot{w}_i/\dot{u}_i^1||\dot{u}_i^2} ((s_i^1, s_i^2), v_i^1 \cup v_i^2)$$

of duration $t = r_0 + r_1 + \cdots + r_{i-1}$ containing at least i discrete steps, where i is the number of discrete transitions until time t performed in r_1 or r_2. Then time point t determines two indexes j_1, j_2 (steps of r_1, r_2) such that $j_1 = \max_i\{\sum_{j=0}^{i-1} r_j^1 \leq t\}$ and $j_2 = \max_i\{\sum_{j=0}^{i-1} r_j^2 \leq t\}$. We shall maintain the inductive assumption that at least one of the transducers is at a discrete transition at time t. Note this is true for the partial run of duration 0 constructed above. Let us suppose without loss of generality that it is $\mathcal{T}_1$ and that $\mathcal{T}_2$ is in the middle of a time step. Then $d_1 := r_{j_1}^1$ is the duration of the time step taken after $\delta_{j_1}^1$ occurs in r_1 at time t, and let $d_2 := (r_0^2 + r_1^2 + \cdots + r_{j_2}^2) - t$ be the remaining duration of the time step from t until the next transition $\delta_{j_2+1}^2$ in r_2. Then we compare d_1, d_2 and distinguish three cases:

- *Case $d_1 < d_2$*: the partial run r_{i+1} of duration $t + d_1$ is obtained by adding to r_i a time step of duration d_1 followed by a left-sided transition. During the time step, the automaton reads the open segment $(w_{j_1}^{d_1})_{|P_1}||(\widetilde{w}_{j_2}^{d_1})_{|P_2}$, where $\widetilde{w}_{j_2}^{d_1}$ has domain $(0, d_1)$ and evaluates as $\widetilde{w}_{j_2}^{d_1}(x) = w_{j_2}^{d_2}(x + t - (r_0^2 + r_1^2 + \cdots + r_{j_2-1}^2))$ for all $x \in (0, d_1)$. Further, it outputs the segment $u_{j_1}^{d_1}||\widetilde{u}_{j_2}^{d_1}$ where $\widetilde{u}_{j_2}^{d_1}$ has domain $(0, d_1)$ and it is defined as $\widetilde{u}_{j_2}^{d_1}(x) = u_{j_2}^{d_2}(x + t - (r_0^2 + r_1^2 + \cdots + r_{j_2-1}^2))$ for all $x \in (0, d_1)$. The left-sided transition δ_{i+1} is built from $\delta_{j_1}^1$ while the automaton reads $(\dot{w}_{j_1+1}^1)_{|P_1}||\dot{w}_{d_1}^2$ and outputs $\dot{u}_{j_1+1}^1||\dot{u}_{d_1}^2$ where $\dot{w}_{d_1}^2, \dot{u}_{d_1}^2$ are the point segments defined at 0 with values $w_2^{d_2}(d_1 + t - (r_0^2 + r_1^2 + \cdots + r_{j_2-1}^2))$ and $u_2^{d_2}(d_1 + x + t - (r_0^2 + r_1^2 + \cdots + r_{j_2-1}^2))$.
- *Case $d_1 = d_2$*: we define the partial run r_{i+1} from r_i by adding a time transition of duration d_1 followed by a simultaneous transition built from

$\delta^1_{j_1+1}, \delta^2_{j_2+2}$. During the time transition, $\mathcal{T}_1||\mathcal{T}_2$ reads the open segment $(w^{d_1}_{j_1})_{|P_1}||(\widetilde{w}^{d_2}_{j_2})_{|P_2}$ and outputs $u^{d_1}_1||\widetilde{u}^{d_2}_{j_2}$, where $\widetilde{w}^{d_2}_{j_2}(x) = w^{d_2}_{j_2}(x+t-(r^2_0+r^2_1+\cdots+r^2_{j_2-1}))$ for all $x \in (0,d_2)$ and $\widetilde{u}^{d_2}_{j_2}(x) = u^{d_2}_{j_2}(x+t-(r^2_0+r^2_1+\cdots+r^2_{j_2-1}))$ for all $x \in (0,d_2)$. During the simultaneous transition the automaton reads $(\dot{w}^1_{j_1+1})_{|P_1}||\dot{w}^2_{j_2+1}$ and outputs $\dot{u}^1_{j_1+1}||\dot{u}^2_{j_2+1}$.

- *Case* $d_1 > d_2$: we apply symmetrically the process shown for $d_1 < d_2$.

Observe that all constructed transitions are allowed by $\mathcal{T}_1||\mathcal{T}_2$ by definition. Moreover, the partial run r_{i+1} has duration $t + \min\{d_1, d_2\}$ and has at least $i+1$ discrete steps. We define the run r as $\bigcup_i r_i$ and it is clear to see that it is accepting if both r_1, r_2 are accepting. See that if r_i visits state s_i at time t for $i = 1, 2$, then r visits state (s_1, s_2) at time t. Similar for transitions. Then let t be a point of time. We want to see that for every $F \in \mathcal{F}$, there is some $t' > t$ s.t. r is visiting some element of F at time t'. For every $F \in \mathcal{F}$ there is some $F_i \in \mathcal{F}_i$ from which it is constructed. Suppose $F = (F_1 \cap S_1) \times S_2 \cup$
$\{\delta \in \Delta : \delta$ is simultaneous or left-sided built from a transition $\delta_1 \in F_1\}$; the other case is symmetric. Then since r_1 is accepting there is some s_1 or δ_1 in $F_1 \in \mathcal{F}_1$ visited by r_1 at some time $t' > t$. If it is a state s_1, then r is visiting (s_1, s_2) at time $t' > t$ for some $s_2 \in S_2$, but then $(s_1, s_2) \in F$. Otherwise it is a transition δ_1, and then the left-sided transition δ built from δ_1 belongs to F. □

Lemma 3.4.2. *Let $\mathcal{T}_1, \mathcal{T}_2$ be two timed transducers with input and output variables P_1, P_2 and Q_1, Q_2 respectively. Then for every signal w over $P_1 \cup P_2$, if r is a run of $\mathcal{T}_1||\mathcal{T}_2$ over w inducing u, then there exist runs r_1, r_2 of $\mathcal{T}_1, \mathcal{T}_2$ over $w_{|P_1}, w_{|P_2}$ inducing $u_{|Q_1}, u_{|Q_2}$ respectively. Moreover, if r is accepting then r_1, r_2 are accepting.*

Proof. Let r be an infinite arbitrary run of $\mathcal{T}_1||\mathcal{T}_2$ over w inducing u. For every valuation v over $C_1 \cup C_2$ we define v_1, v_2 over C_1, C_2 as $v_1 = v_{|C_1}$ and $v_2 = v_{|C_2}$. We define a *pseudorun* as a sequence of time and discrete transitions such that it satisfies all the properties of a run except that several time steps can be taken subsequently without any discrete transition in between. We construct the pseudoruns r_1, r_2 as a union of partial pseudoruns $\bigcup_i r^1_i$ and $\bigcup_i r^2_i$ build recursively on i, where i indexes the i-th time step of r. For all index $i \geq 0$, the partial pseudoruns r^1_i, r^2_i have duration $t = \sum_{j=0}^{i} r_j$ where r_j is the length of the j-th time step in r, having at most i discrete steps.

Suppose r starts at configuration $((\underline{s}_1, \underline{s}_2), \underline{v})$. Then the partial pseudoruns r^1_0, r^2_0 start at configurations $(\underline{s}_1, \underline{v}_1), (\underline{s}_2, \underline{v}_2)$ respectively.

For all $i \geq 0$, suppose we have constructed the partial pseudo runs r^1_i, r^2_i of duration t. Then one of the following discrete transitions appears in r at time t:

- *a simultaneous transition* $\delta_{i+1} = ((s^1_i, s^2_i), g, R, (s^1_{i+1}, s^2_{i+1}))$ by reading $\dot{w}_{i+1}$ and outputing $\dot{u}_{i+1}$ where $g = g_1 \wedge g_2$ and $R = R_1 \cup R_2$, followed by *a time step of duration* d_{i+1} by reading $w^{d_{i+1}}_{i+1}$ and outputing $u^{d_{i+1}}_{i+1}$.

Then add to r_i^1, r_i^2 discrete transitions $\delta_{i+1}^1 = ((s_i^1, g_1, R_1, s_{i+1}^1))$ and $\delta_{i+1}^2 = ((s_i^2, g_2, R_2, s_{i+1}^2))$ by reading $(\dot{w}_{i+1})_{|P_1}, (\dot{w}_{i+1})_{|P_2}$ and outputing $(\dot{u}_{i+1})_{|Q_1}, (\dot{u}_{i+1})_{|Q_2}$ respectively, followed by a time step of duration d_{i+1} by reading $(w_{i+1}^{d_{i+1}})_{|P_1}, (w_{i+1}^{d_{i+1}})_{|P_2}$, and outputing $(u_{i+1}^{d_{i+1}})_{|Q_1}, (u_{i+1}^{d_{i+1}})_{|Q_2}$ respectively.

- *a left-sided transition* $\delta_{i+1} = ((s_i^1, s_i^2), g \wedge Inv_2(s_i^2), R, (s_{i+1}^1, s_i^2))$ by reading $\dot{w}_{i+1}$ and outputing $\dot{u}_{i+1}$, followed by *a time step of duration* d_{i+1} by reading $w_{i+1}^{d_{i+1}}$ and outputing $u_{i+1}^{d_{i+1}}$.

  Then put the discrete transition (s_i^1, g, R, s_{i+1}^1) in r_i^1, reading $(\dot{w}_{i+1})_{|P_1}$, and outputing $(\dot{u}_{i+1})_{|Q_1}$. After this, put a time step of duration d_{i+1} both in r_i^2, and in r_i^1 after the discrete transition constructed above. During this time step, $\mathcal{T}_1$ reads the open segment $(w_{i+1}^{r_{i+1}})_{|P_1}$ and outputs $(u_{i+1}^{r_{i+1}})_{|Q_1}$, while $\mathcal{T}_2$ reads the open segment $(w_{i+1}^{r_{i+1}})_{|P_2}$ and outputs $(u_{i+1}^{r_{i+1}})_{|Q_2}$.

- for *a right-sided transition* proceed symmetrically.

Observe that all constructed transitions are allowed by $\mathcal{T}_1$ and $\mathcal{T}_2$ by definition. Moreover, the partial runs r_{i+1}^1, r_{i+1}^2 have duration $t + d_{i+1}$ and at most $i + 1$ discrete steps. It only remains to prove that if r is accepting then so are $r_1 = \bigcup_i r_i^1$ and $r_2 = \bigcup_i r_i^2$.

In fact, if $\mathcal{T}_1||\mathcal{T}_2$ is at state $s = (s_1, s_2)$ at time t', then $\mathcal{T}_1$ is at state s_1 and $\mathcal{T}_1$ is at state s_2 at time t'. If $\mathcal{T}_1||\mathcal{T}_2$ is at transition δ built from δ_1, δ_2, then $\mathcal{T}_1$ is at transition δ_1 and $\mathcal{T}_2$ is at transition δ_2 at time t'.

To see that r_1 is accepting, let $t \geq 0$ and $F_1 \in \mathcal{F}_1$. Then there is some $F \in \mathcal{F}$ constructed from F_1 and some element $x \in F_1$ and some $t' > t$ s.t. $\mathcal{T}_1||\mathcal{T}_2$ is at x at time t'. By definition of $\mathcal{F}$, then x is a state $s = (s_1, s_2)$ where $s_1 \in F_1$ or a left-sided transition δ built from some $\delta_1 \in F_1$. But if $\mathcal{T}_1||\mathcal{T}_2$ is at state $s = (s_1, s_2)$ at time t', then $\mathcal{T}_1$ is at state s_1 at time t'. If $\mathcal{T}_1||\mathcal{T}_2$ is at transition δ built from δ_1, then $\mathcal{T}_1$ is at transition δ_1 at time t'. Hence r_1 is accepting. By a symmetric argument, so is r_2. □

As a consequence, for all parallel-composable timed transducers $\mathcal{T}_1$ and $\mathcal{T}_2$ with input variables P_1, P_2, if w is a signal over $P_1 \cup P_2$, then $w \in L(\mathcal{T}_1||\mathcal{T}_2)$ iff $w \in L(\mathcal{T}_1) \cap L(\mathcal{T}_2)$.

Proposition 3.4.1. *For all timed functional transducers $\mathcal{T}_1$ and $\mathcal{T}_2$ with input variables P_1 and P_2 respectively, if $\mathcal{T}_1$ and $\mathcal{T}_2$ are parallel-composable, then $\mathcal{T}_1||\mathcal{T}_2$ is functional, and for any signal w over $P_1 \cup P_2$ we have*

$$\mathcal{T}_1||\mathcal{T}_2(w) = \mathcal{T}_1(w|_{P_1})||\mathcal{T}_2(w|_{P_2}).$$

Proof. Let $\mathcal{T}_1, \mathcal{T}_2$ be two functional timed transducers with input and output variables P_1, P_2 and Q_1, Q_2 respectively. Also let w be a signal over $P_1 \cup P_2$. Denote by $u_i = \mathcal{T}_i(w_{|P_i})$ for $i = 1, 2$. This means that there exist runs r_1, r_2 of $\mathcal{T}_1, \mathcal{T}_2$ over $w_{|P_1}, w_{|P_2}$ inducing u_1, u_2. Then by Lemma 3.4.1 there exists a run r of $\mathcal{T}_1||\mathcal{T}_2$ over w inducing $u_1||u_2$ which is exactly $\mathcal{T}_1(w_{|P_1})||\mathcal{T}_2(w_{|P_2})$.

Now suppose there exists another $\mathcal{T}_1||\mathcal{T}_2$-output over w, and denote it u'. But then, from the run r inducing it, by Lemma 3.4.2 we can find two runs r'_1, r'_2 of $\mathcal{T}_1, \mathcal{T}_2$ over $w_{|P_1}, w_{|P_2}$ inducing $u'_{|Q_1}, u'_{|Q_2}$. But since $\mathcal{T}_1, \mathcal{T}_2$ are functional, then $u'_{|Q_1} = u_1$ and $u_2 = u'_{|Q_2}$. Hence $u' = u$. □

Definition 3.4.3. We say that two timed transducers $\mathcal{T}_1$ and $\mathcal{T}_2$ with input and output variables P_1, P_2 and Q_1, Q_2 respectively are *sequential-composable* if $C_1 \cap C_2 = P_1 \cap P_2 = Q_1 \cap Q_2 = \emptyset$ and $Q_1 = P_2$. In this case, for $i = 1, 2$ and $\mathcal{T}_i = (S_i, \underline{s}_i, P_i, Q_i, C_i, Inv_i, \Delta_i, \lambda_i, \gamma_i, \mathcal{F}_i)$, we can define their sequential composition.

The *sequential composition* of $\mathcal{T}_1$ and $\mathcal{T}_2$, denoted by $\mathcal{T}_1; \mathcal{T}_2$, is the timed transducer $(S, \underline{s}, P, Q, X, Inv, \Delta, \lambda, \gamma, \mathcal{F})$ such that

- $S = S_1 \times S_2$
- $\underline{s} = (\underline{s}_1, \underline{s}_2)$
- $P = P_1$
- $Q = Q_2$
- $C = C_1 \cup C_2$
- $Inv((s_1, s_2)) = Inv_1(s_1) \wedge Inv_2(s_2)$
- The transition relation Δ consists of three kinds of transitions:
  - *Simultaneous transitions* $\delta = ((s_1, s_2), g, R, (s'_1, s'_2))$, where $\delta_1 = (s_1, g_1, R_1, s'_1) \in \Delta_1$, $\delta_2 = (s_2, g_2, R_2, s'_2) \in \Delta_2$, $g = g_1 \wedge g_2$ and $R = R_1 \cup R_2$. We say that δ is *built* from δ_1 and δ_2.
  - *Left-sided transitions* $\delta = ((s_1, s_2), g \wedge Inv_2(s_2), R, (s'_1, s_2))$ where $\delta_1 = (s_1, g, R, s'_1) \in \Delta_1$. We say that δ is *built* from δ_1.
  - *Right-sided transitions* $\delta = ((s_1, s_2), Inv_1(s_1) \wedge g, R, (s_1, s'_2))$ where $\delta = (s_2, g, R, s'_2) \in \Delta_2$. We say that δ is *built* from δ_2.
- The *input labeling* $\lambda : S \cup \Delta \to BC(P)$ is defined as follows
  - for a state (s_1, s_2), put $\lambda((s_1, s_2)) = \lambda_1(s_1)$ if $\gamma_1(s_1) \wedge \lambda_2(s_2)$ is satisfiable; otherwise put $\lambda((s_1, s_2)) = \bot$
  - for a simultaneous transition δ built from δ_1, δ_2, put $\lambda(\delta) = \lambda_1(\delta_1)$ if $\gamma_1(\delta_1) \wedge \lambda_2(\delta_2)$ is satisfiable; otherwise put $\lambda(\delta) = \bot$.
  - for a left-sided transition $\delta = ((s_1, s_2), g \wedge Inv_2(s_2), R, (s'_1, s_2))$, put $\lambda(\delta) = \lambda_1((s_1, g, R, s'_1))$ if $\gamma_1((s_1, g, R, s'_1)) \wedge \lambda_2(s_2))$ is satisfiable; otherwise put $\lambda(\delta) = \bot$.
  - for a right-sided transition $\delta = ((s_1, s_2), Inv_1(s_1) \wedge g, R, (s_1, s'_2))$ put $\lambda(\delta) = \lambda_1(s_1)$ if $\gamma_1(s_1) \wedge \lambda_2((s_2, g, R, s'_2))$ is satisfiable; otherwise put $\lambda(\delta) = \bot$.

- The *output labeling* $\gamma : S \cup \Delta \to BC(Q)$ is defined as follows
  - for a state (s_1, s_2), put $\gamma((s_1, s_2)) = \gamma_2(s_2)$
  - for a simultaneous transition δ built from δ_1, δ_2, put $\gamma(\delta) = \gamma_2(\delta_2)$
  - for a left-sided transition $\delta = ((s_1, s_2), g \wedge Inv_2(s_2), R, (s_1', s_2))$, put $\gamma(\delta) = \gamma_2(s_2)$
  - for a right-sided transition $\delta = ((s_1, s_2), Inv_1(s_1) \wedge g, R, (s_1, s_2'))$ put $\gamma(\delta) = \gamma_2(s_2, g, R, s_2')$
- The generalized Büchi condition $\mathcal{F}$ contains the following sets:
  - For every $F_2 \in \mathcal{F}_2$, it contains the set $S_1 \times (F_2 \cap S_2) \cup \{\delta \in \Delta : \delta$ is simultaneous or right-sided built from a transition $\delta_2 \in F_2\}$.
  - For every $F_1 \in \mathcal{F}_1$, it contains the set $(F_1 \cap S_1) \times S_2 \cup \{\delta \in \Delta : \delta$ is simultaneous or left-sided built from a transition $\delta_1 \in F_1\}$.

In the following lemmas we show what is the relation among the runs of $\mathcal{T}_1, \mathcal{T}_2$ and the runs of $\mathcal{T}_1; \mathcal{T}_2$.

Lemma 3.4.3. *Let $\mathcal{T}_1, \mathcal{T}_2$ be two sequential-composable timed transducers with input and output variables P_1, P_2 and P_2, Q_2 respectively. Then for every signal w over P_1, if r_1 is a run of $\mathcal{T}_1$ over w inducing u' and r_2 is a run of $\mathcal{T}_2$ over u' inducing u, then there exists a run r of $\mathcal{T}_1; \mathcal{T}_2$ over w inducing u. Moreover, if r_1, r_2 are accepting, then r is accepting.*

Proof. Let w be an arbitrary signal over P_1. We construct the run r of $\mathcal{T}_1; \mathcal{T}_2$ over w inductively.

Suppose the run r_1 starts at configuration $(\underline{s}_1, \underline{v}_1)$ and then takes a transition of the form $(\underline{s}_1, \underline{v}_1) \xrightarrow[\delta_0^1]{\dot{w}_0/\dot{u}_0'} (s_0^1, v_0^1)$ while r_2 starts at configuration $(\underline{s}_2, \underline{v}_2)$ and then takes a transition of the form $(\underline{s}_2, \underline{v}_2) \xrightarrow[\delta_0^2]{\dot{u}_0'/\dot{u}_0} (s_0^2, v_0^2)$. By definition, this is equivalent to $\dot{w}_0 \vDash \lambda_1(\delta_0^1)$, $\dot{u}_0' \vDash \gamma_1(\delta_0^1) \wedge \lambda_2(\delta_0^2)$ and $\dot{u}_0 \vDash \gamma_2(\delta_0^2)$. Then $\gamma_1(\delta_0^1) \wedge \lambda_2(\delta_0^2)$ is satisfiable and then the simultaneous transition δ_0 built from δ_0^1, δ_0^2 satisfies $\lambda(\delta_0) = \lambda_1(\delta_0^1)$. And $\gamma(\delta_0) = \gamma_2(\delta_2)$.

Thus, define the partial run r_0 (of duration 0) as $((\underline{s}_1, \underline{s}_2), \underline{v}_1 \cup \underline{v}_2) \xrightarrow{\dot{w}_0/\dot{u}_0} ((s_0^1, s_0^2), v_0^1 \cup v_0^2)$.

Now assume that we have constructed the partial run r_i

$$((\underline{s}_1, \underline{s}_2), \underline{v}_1 \cup \underline{v}_2) \xrightarrow{\dot{w}_0/\dot{u}_0} ((s_0^1, s_0^2), v_0^1 \cup v_0^2) \to \cdots \xrightarrow{\dot{w}_i/\dot{u}_i} ((s_i^1, s_i^2), v_i^1 \cup v_i^2)$$

of duration $t = r_0 + r_1 + \cdots + r_{i-1}$ containing at least i discrete steps, where i is the number of discrete transitions until time t performed in r_1 or r_2. Then time point t determines two indexes j_1, j_2 (steps of r_1, r_2) such that

$j_1 = \max_i\{\sum_{j=0}^{i-1} r_j^1 \leq t\}$ and $j_2 = \max_i\{\sum_{j=0}^{i-1} r_j^2 \leq t\}$. We shall maintain the inductive assumption that at least one of the transducers is at a discrete transition at time t. Note this is true for the partial run of duration 0 constructed above. Let us suppose without loss of generality that $\mathcal{T}_1$ is at a discrete transition (from s_i^1) and that $\mathcal{T}_2$ is in the middle of a time step (at s_i^2). Then $d_1 := r_{j_1}^1$ is the duration of the time step taken after $\delta_{j_1}^1$ occurs in r_1 at time t, and let $d_2 := (r_0^2 + r_1^2 + \cdots + r_{j_2}^2) - t$ be the remaining duration of the time step from t until the next transition $\delta_{j_2+1}^2$ in r_2. Then we compare d_1, d_2 and distinguish three cases:

- *Case* $d_1 < d_2$: the partial run r_{i+1} of duration $t+d_1$ is obtained by adding to r_i a time step of duration d_1 followed by a left-sided transition. During the time step, the automaton reads the open segment $w_{j_1}^{d_1}$. Further, it outputs the segment $\widetilde{u}_{j_2}^{d_1}$ where $\widetilde{u}_{j_2}^{d_1}$ has domain $(0, d_1)$ and it is defined as $\widetilde{u}_{j_2}^{d_1}(x) = u_{j_2}^{d_2}(x + t - (r_0^2 + r_1^2 + \cdots + r_{j_2-1}^2))$ for all $x \in (0, d_1)$. The left-sided transition δ_{i+1} is built from $\delta_{j_1}^1$ while the automaton reads $\dot{w}_{j_1+1}^1$ and outputs $\dot{u}_{d_1}^2$ where $\dot{u}_{d_1}^2$ is the point segment defined at 0 with value $u_2^{d_2}(d_1 + x + t - (r_0^2 + r_1^2 + \cdots + r_{j_2-1}^2))$.

- *Case* $d_1 = d_2$: we define the partial run r_{i+1} from r_i by adding a time transition of duration d_1 followed by a simultaneous transition built from $\delta_{j_1+1}^1, \delta_{j_2+2}^2$. During the time transition, $\mathcal{T}_1; \mathcal{T}_2$ reads the open segment $w_{j_1}^{d_1}$ and outputs $\widetilde{u}_{j_2}^{d_2}$, where $\widetilde{u}_{j_2}^{d_2}(x) = u_{j_2}^{d_2}(x+t-(r_0^2+r_1^2+\cdots+r_{j_2-1}^2))$ for all $x \in (0, d_2)$. During the simultaneous transition the automaton reads $\dot{w}_{j_1+1}^1$ and outputs $\dot{u}_{j_2+1}^2$.

- *Case* $d_1 > d_2$: we apply symmetrically the process shown for $d_1 < d_2$.

Observe that all constructed transitions are allowed by $\mathcal{T}_1; \mathcal{T}_2$ by definition. Moreover, the partial run r_{i+1} has duration $t + \min\{d_1, d_2\}$ and has at least $i + 1$ discrete steps. We define the run r as $\bigcup_i r_i$ and it is clear to see that it is accepting if both r_1, r_2 are accepting by the same argument as in the proof of Lemma 3.4.1. □

Lemma 3.4.4. *Let $\mathcal{T}_1, \mathcal{T}_2$ be two sequential-composable timed transducers with input and output variables P_1, P_2 and P_2, Q_2 respectively. Then for every signal w over P_1, if r is a run of $\mathcal{T}_1; \mathcal{T}_2$ over w inducing u, then there exist runs r_1, r_2 of $\mathcal{T}_1, \mathcal{T}_2$ such that r_1 is over w and r_2 induces u. Moreover, if r is accepting, then r_1, r_2 are accepting.*

Proof. Let r be an infinite arbitrary run of $\mathcal{T}_1; \mathcal{T}_2$ over w inducing u. For every valuation v over $C_1 \cup C_2$ we define v_1, v_2 over C_1, C_2 as $v_1 = v_{|C_1}$ and $v_2 = v_{|C_2}$. We construct the pseudoruns r_1, r_2 as a union of partial pseudoruns $\bigcup_i r_i^1$ and $\bigcup_i r_i^2$ build recursively on i, where i indexes the i-th time step of r. For all index $i \geq 0$, the partial pseudoruns r_i^1, r_i^2 have duration $t = \sum_{j=0}^{i} r_j$ where r_j is the length of the j-th time step in r, having at most i discrete steps.

Suppose r starts at configuration $((\underline{s}_1, \underline{s}_2), \underline{v})$. Then the partial pseudoruns r_0^1, r_0^2 start at configurations $(\underline{s}_1, \underline{v}_1), (\underline{s}_2, \underline{v}_2)$ respectively.

For all $i \geq 0$, suppose we have constructed the partial pseudoruns r_i^1, r_i^2 of duration t. Then one of the following discrete transitions appears in r at time t:

- *a simultaneous transition* $\delta_{i+1} = ((s_i^1, s_i^2), g, R, (s_{i+1}^1, s_{i+1}^2))$ by reading $\dot{w}_{i+1}$ and outputing $\dot{u}_{i+1}$ where $g = g_1 \wedge g_2$ and $R = R_1 \cup R_2$, followed by *a time step of duration* d_{i+1} by reading $w_{i+1}^{d_{i+1}}$ and outputing $u_{i+1}^{d_{i+1}}$, at state (s_{i+1}^1, s_{i+1}^2).

  Then add to r_i^1, r_i^2 discrete transitions $\delta_{i+1}^1 = ((s_i^1, g_1, R_1, s_{i+1}^1))$ and $\delta_{i+1}^2 = ((s_i^2, g_2, R_2, s_{i+1}^2))$ by reading $\dot{w}_{i+1}, \dot{u}'_{i+1}$ and outputing $\dot{u}'_{i+1}, \dot{u}_{i+1}$ respectively. The point segment $\dot{u}'_{i+1}$ is such that $\dot{u}'_{i+1} \vDash \gamma_1(\delta_{i+1}^1) \wedge \lambda_2(\delta_{i+1}^2)$ and its existance is by definition of λ.

  After the discrete transitions, add a time step of duration d_{i+1} by reading $w_i^{d_{i+1}}, {u'_i}^{d_{i+1}}$ and outputing ${u'_i}^{d_{i+1}}, u_i^{d_{i+1}}$ respectively. The open segment ${u'_{i+1}}^{d_{i+1}}$ is such that ${u'_i}^{d_i} \vDash \gamma_1(s_{i+1}^1) \wedge \lambda_2(s_{i+1}^2)$ and its existance is by definition of λ.

- *a left-sided transition* $\delta_{i+1} = ((s_i^1, s_i^2), g \wedge Inv_2(s_i^2), R, (s_{i+1}^1, s_i^2))$ by reading $\dot{w}_{i+1}$ and outputing $\dot{u}_{i+1}$, followed by *a time step of duration* d_{i+1} by reading $w_{i+1}^{d_{i+1}}$ and outputing $u_{i+1}^{d_{i+1}}$, at state (s_{i+1}^1, s_{i+1}^2).

  Then put discrete transition (s_i^1, g, R, s_{i+1}^1) in r_i^1 by reading $\dot{w}_{i+1}$ and outputing $\dot{u}'_{i+1}$ where $\dot{u}'_{i+1} \vDash \gamma_1((s_i^1, g, R, s_{i+1}^1)) \wedge \lambda_2(s_i^2)$. Such $\dot{u}'_{i+1}$ exists by definition of λ.

  After put a time step both in r_i^2 and in r_i^1 after the transition constructed above, of duration d_{i+1}, by reading $w_{i+1}^{d_{i+1}}, {u'_{i+1}}^{d_{i+1}}$ and outputing ${u'_{i+1}}^{d_{i+1}}, u_{i+1}^{d_{i+1}}$ respectively. The open segment ${u'_{i+1}}^{d_{i+1}}$ is such that ${u'_i}^{d_i} \vDash \gamma_1(s_{i+1}^1) \wedge \lambda_2(s_{i+1}^2)$ and its existance is by definition of λ.

- for *a right-sided transition* proceed symmetrically.

Observe that all constructed transitions are allowed by $\mathcal{T}_1$ and $\mathcal{T}_2$ by definition. Moreover, the partial pseudoruns r_{i+1}^1, r_{i+1}^2 have duration $t + d_{i+1}$ and at most $i + 1$ discrete steps. From the pseudoruns r_1, r_2 it is easy to obtains the final runs r_1, r_2 with alternating time and discrete steps. It only remains to prove that if r is accepting then so are $r_1 = \bigcup_i r_i^1$ and $r_2 = \bigcup_i r_i^2$.

In fact, if $\mathcal{T}_1; \mathcal{T}_2$ is at state $s = (s_1, s_2)$ at time t', then $\mathcal{T}_1$ is at state s_1 and $\mathcal{T}_1$ is at state s_2 at time t'. If $\mathcal{T}_1; \mathcal{T}_2$ is at transition δ built from δ_1, δ_2, then $\mathcal{T}_1$ is at transition δ_1 and $\mathcal{T}_2$ is at transition δ_2 at time t'.

To see that r_1 is accepting, let $t \geq 0$ and $F_1 \in \mathcal{F}_1$. Then there is some $F \in \mathcal{F}$ constructed from F_1 and some element $x \in F_1$ and some $t' > t$ s.t. $\mathcal{T}_1; \mathcal{T}_2$ is at x at time t'. By definition of $\mathcal{F}$, then x is a state $s = (s_1, s_2)$ where $s_1 \in F_1$ or a left-sided transition δ built from some $\delta_1 \in F_1$. But if $\mathcal{T}_1; \mathcal{T}_2$ is at state $s = (s_1, s_2)$ at time t', then $\mathcal{T}_1$ is at state s_1 at time t'. If $\mathcal{T}_1; \mathcal{T}_2$ is at transition

δ built from δ_1, then $\mathcal{T}_1$ is at transition δ_1 at time t'. Hence r_1 is accepting. By a symmetric argument, so is r_2. □

Proposition 3.4.2. *For all timed functional transducers $\mathcal{T}_1$ and $\mathcal{T}_2$ with input variables P_1 and P_2 respectively, if $\mathcal{T}_1$ and $\mathcal{T}_2$ are sequential-composable, then $\mathcal{T}_1;\mathcal{T}_2$ is functional, and for any signal w over P_1 we have*

$$\mathcal{T}_1;\mathcal{T}_2(w) = \mathcal{T}_2(\mathcal{T}_1(w)).$$

Proof. Let $\mathcal{T}_1, \mathcal{T}_2$ be two functional sequential-composable timed transducers with input and output variables P_1, P_2 and P_2, Q_2 respectively. Also let w be a signal over P_1. Denote by $u' = \mathcal{T}_1(w)$ and by $u = \mathcal{T}_2(u')$. This means that there exist accepting runs r_1, r_2 of $\mathcal{T}_1, \mathcal{T}_2$ over w, u' inducing u', u respectively. Then by Lemma 3.4.3 there exists an accepting run r of $\mathcal{T}_1;\mathcal{T}_2$ over w inducing u which is exactly $\mathcal{T}_2(\mathcal{T}_1(w))$. Now suppose there exists another $\mathcal{T}_1;\mathcal{T}_2$-output over w, and denote it by $\widetilde{u}$. But then, from the accepting run $\widetilde{r}$ inducing it, by Lemma 3.4.4 we can find accepting runs $\widetilde{r}_1$ of $\mathcal{T}_1$ over w inducing u', and $\widetilde{r}_2$ of $\mathcal{T}_2$ over u' inducing $\widetilde{u}$. But then $\widetilde{u} = u$. □

3.5 Temporal testers for basic formulas

3.5.1 Strict Since

To construct a temporal tester for $p_1 \mathcal{S} p_2$ for arbitrary different propositions p_1, p_2, we analize the semantics of $\mathcal{S}$ and describe some of its properties. We know that for a signal w and time instant t, $(w,t) \vDash p_1 \mathcal{S} p_2$ if and only if there is some $t' < t$ such that $(w,t') \vDash p_2$ and for all $t'' \in (t',t), (w,t'') \vDash p_1$. As a consequence of the definition, $(w,t) \vDash p_1 \mathcal{S} p_2$ if and only if $(w_{|[0,t]}, t) \vDash p_1 \mathcal{S} p_2$.

Let us begin by pointing out that the characteristic function for *since* is left-continuous.

Lemma 3.5.1. *For every signal and time w, t, if $(w,t) \vDash \varphi_1 \mathcal{S} \varphi_2$, then either $t = 0$ or there exists $t' < t$ such that for all $t'' \in (t',t)$, $(w,t'') \vDash \varphi_1 \mathcal{S} \varphi_2$. In other words, for all formulas φ_1, φ_2 and signal w, the function $\chi_{\varphi_1 \mathcal{S} \varphi_2}(w)$ is* left-continuous.

Proof. Let w be an arbitrary signal and $u = \chi_{\varphi_1 \mathcal{S} \varphi_2}(w)$. Then u can be decomposed as a concatenation of the form $\dot{u}_0 \cdot u_0^{r_0} \cdot \dot{u}_1 \cdot u_1^{r_1} \cdots$. In this sense, left-continuity is equivalent to showing for every $i \geq 1$ that $\dot{u}_i(0) = u_{i-1}^{r_{i-1}}(t)$ for all $t \in (0, r_{i-1})$. First, assume that $\dot{u}_i(0) = 1$. Then there exists $t < t_i$ such that φ_2 is satisfied at t and φ_1 holds continuously throughout the interval (t, t_i). As a consequence $p_1 \mathcal{S} p_2$ holds along (t, t_i), whence $u_{i-1}^{r_{i-1}}(t) = 1 = \dot{u}_i(0)$ for all $t \in (0, r_{i-1})$. If $\dot{u}_i(0) = 0$, then we have two possibilities, (i) either φ_2 never becomes true at any $t \in [0, t_i)$ and then $u(t) = 0$ for every $t \in [0, t_i)$, or (ii) for any $t \in [0, t_i)$ where φ_2 is true, there is some t' between t and t_i where φ_1 is false, which implies that $\varphi_1 \mathcal{S} \varphi_2$ is false over (t', t_i), and then $u_{i-1}^{r_{i-1}}(t) = 0 = \dot{u}_i(0)$ for all $t \in (0, r_{i-1})$. □

If $\dot{w}_0 \cdot w_0^{r_0} \cdot \dot{w}_1 \cdot w_1^{r_1} \cdots$ is a compatible point-segment partition of a signal w, from now on we use the notation w_i for the constant value $w_i^{r_i}(t)$ with $t \in (0, r_i)$ and also use w_i to refer to $w_i^{r_i}$ as an open segment. Similarly, by $\dot{w}_i$ we mean the point segment as well as the value $\dot{w}_i(0)$. Thus, we write $w_i \vDash \varphi$ instead of $(w_i^{r_i}, t) \vDash \varphi$ as well as $\dot{w}_i \vDash \varphi$ instead of $(\dot{w}_i, 0) \vDash \varphi$.

In the next theorem, we characterise the semantics of *since*.

Theorem 3.5.1. *For every boolean signal w over p_1, p_2 and every boolean signal u over q, we have*

$$u = \chi_{p_1 \mathcal{S} p_2}(w)$$

if and only if there exists a point-segment partition $\mathcal{I} = \{t_0\}(t_0, t_1)\{t_1\}(t_1, t_2)\cdots$ compatible with w, u such that $w = \dot{w}_0 \cdot w_0^{r_0} \cdot \dot{w}_1 \cdot w_1^{r_1} \cdots$ and $u = \dot{u}_0 \cdot u_0^{r_0} \cdot \dot{u}_1 \cdot u_1^{r_1} \cdots$ such that for all $i \geq 0$,

1. $\dot{u}_0 = 0$,
2. $u_i = \dot{u}_{i+1}$,
3. *if* $w_i \vDash \neg p_1$, *then* $u_i = 0$,
4. *if* $w_i \vDash p_1 \wedge p_2$, *then* $u_i = 1$,
5. *if* $w_i \vDash p_1 \wedge \neg p_2$, *there are three possibilities*
   (a) *if* $\dot{w}_i \vDash \neg p_1 \wedge \neg p_2$, *then* $u_i = 0$,
   (b) *if* $\dot{w}_i \vDash p_2$, *then* $u_i = 1$,
   (c) *if* $\dot{w}_i \vDash p_1 \wedge \neg p_2$, *then* $u_i = \dot{u}_i$.

Proof. ($\Rightarrow$) Let w be an arbitrary signal. If $u = \chi_{p_1 \mathcal{S} p_2}(w)$, then u is a signal and satisfies these properties by definition. Consider the point-segment partition $\mathcal{I}$ induced by the discontinuities of w and u. Then u and w can be decomposed with respect to $\mathcal{I}$. By following the semantics of $p_1 \mathcal{S} p_2$ we have $(w, 0) \nvDash p_1 \mathcal{S} p_2$ as there is no $t' < 0$. Hence $\dot{u}_0 = 0$. Now if $i \geq 1$ we can see that $u_{i-1} = \dot{u}_i$. If $\dot{w}_i \vDash p_1 \mathcal{S} p_2$ then by left-continuity there exists some $t' < t_i$ such that $\forall t'' \in (t', t_i)$, $(w, t'') \vDash p_1 \mathcal{S} p_2$. Hence, since $(t', t_i) \cap (t_{i-1}, t_i) \neq \emptyset$, then we obtain $w_{i-1} \vDash p_1 \mathcal{S} p_2$. Conversely, if $w_{i-1} \nvDash p_1 \mathcal{S} p_2$, then there is no $t' < t_i$, such that $p_1 \mathcal{S} p_2$ holds along (t', t_i), whence by left-continuity, $\dot{w}_i \nvDash p_1 \mathcal{S} p_2$.

Moreover, for all $i \geq 0$,

1. If $w_i \vDash \neg p_1$, then $\forall t' < t_i$, there exists $t'' \in (t', t_i)$ such that $(w, t'') \vDash \neg p_1$, and then $w_i \nvDash p_1 \mathcal{S} p_2$. Hence, $u_i = 0$.
2. If $w_i \vDash p_1 \wedge p_2$, then $\forall t \in (t_i, t_{i+1})$, there is some t' between t_i and t such that $(w, t') \vDash p_2$ and $\forall t'' \in (t', t)$, $(w, t'') \vDash p_1$. Hence $(w, t) \vDash p_1 \mathcal{S} p_2$, or equivalently, $u_i = 1$.
3. If $w_i \vDash p_1 \wedge \neg p_2$, then

(a) if $\dot{w}_i \vDash \neg p_1 \wedge \neg p_2$, then pick some $t \in (t_i, t_{i+1})$. For all $t' < t_i$ such that $(w, t') \vDash p_2$, there exists some t'', exactly t_i, between t' and t such that $(w, t'') \nvDash p_1$, and then $w_i \nvDash p_1 \mathcal{S} p_2$, whence $u_i = 0$.

(b) if $\dot{w}_i \vDash p_2$, then for every $t \in (t_i, t_{i+1})$, we have that there exists $t' = t_i < t$ satisfying p_2 and for every $t' < t'' < t$, p_1 holds true. Then $(w, t) \vDash p_1 \mathcal{S} p_2$, whence $u_i = 1$.

(c) if $\dot{w}_i \vDash p_1 \wedge \neg p_2$, either $(w, t) \vDash p_1 \wedge \neg p_2$ for all $t < t_i$, in which case $\dot{u}_0 = u_0 = \cdots = \dot{u}_i = u_i = 0$. Otherwise, we consider the point or open segment of w with maximum index, less than i, where $p_1 \wedge \neg p_2$ doesn't hold. If it is a point segment where p_2 holds, then $u_i = \dot{u}_i = 1$; if $\neg p_1 \wedge \neg p_2$ holds, then $u_i = \dot{u}_i = 0$. If it is an open segment where $p_1 \wedge p_2$ holds, then $u_i = \dot{u}_i = 1$, but if $\neg p_1$ holds, then $u_i = \dot{u}_i = 0$.

($\Leftarrow$) Suppose that there exists such an u. We have to prove that $u = \chi_{p_1 \mathcal{S} p_2}(w)$ i.e. that for all $t \geq 0$, $(u, t) \vDash 1$ iff $(w, t) \vDash p_1 \mathcal{S} p_2$. Let $t \geq 0$, and let us show both directions of the equivalence, one directly and the second by contraposition. We have two cases, $t \in (t_i, t_{i+1})$ or $t = t_i$ for some $i \geq 0$.
Case $t \in (t_i, t_{i+1})$.

- If $(w, t) \vDash p_1 \mathcal{S} p_2$, then by definition either
  (a) $(w, t) \vDash p_1 \wedge p_2$, and then by 4, $u_i = 1$, or
  (b) $(w, t) \vDash p_1 \wedge \neg p_2$, and either the point or open segment of w with maximum index, less than i, where $p_1 \wedge \neg p_2$ doesn't hold satisfies (i) p_2 if it is a point segment, and then by 5(b), 5(c) and 2, $u_i = 1$, or (ii) $p_1 \wedge p_2$ if it is an open segment, and then by 4, 2 and 5(c), $u_i = 1$.

- If $w(t) \nvDash p_1 \mathcal{S} p_2$, then by definition either
  (a) $(w, t) \vDash \neg p_1$, and then by 3, $u_i = 0$, or
  (b) $(w, t) \vDash p_1 \wedge \neg p_2$ and either the point or open segment of w with maximum index, less than i, where $p_1 \wedge \neg p_2$ doesn't hold satisfies (i) $\neg p_1 \wedge \neg p_2$ if it is a point segment, and then by 5(a), 5(c) and 2, $u_i = 0$, or (ii) $\neg p_1$ if it is an open segment, and then by 2,3, and 5(c), $u_i = 0$.

Case $t = t_i$.

- If $i \geq 1$ and $\dot{w}_i \vDash p_1 \mathcal{S} p_2$, then by definition either (a) $w_{i-1} \vDash p_1 \wedge p_2$, and then by 4 and 2, $\dot{u}_i = 1$, or (b) $w_{i-1} \vDash p_1 \wedge \neg p_2$, and either the point or open segment of w with maximum index, less than $i - 1$, where $p_1 \wedge \neg p_2$ doesn't hold satisfies (i) p_2 if it is a point segment, and then by 5(b), 5(c) and 2, $\dot{u}_i = 1$, or (ii) $p_1 \wedge p_2$ if it is an open segment, and then by 4, 2 and 5(c), $\dot{u}_i = 1$.

- If $\dot{w}_i \nvDash p_1 \mathcal{S} p_2$, then by definition $i = 0$ and $\dot{u}_0 = 0$ by 1, or $i \geq 1$ and either (a) $w_{i-1} \vDash \neg p_1$, and then by 2 and 3, $\dot{u}_i = 0$ or (b) $w_{i-1} \vDash p_1 \wedge \neg p_2$ and either the point or open segment of w with maximum index, less than $i - 1$, where $p_1 \wedge \neg p_2$ doesn't hold satisfies (i) $\neg p_1 \wedge \neg p_2$ if it is a point

segment, and then by 5(a), 5(c) and 2, $\dot{u}_i = 0$, or (ii) $\neg p_1$ if it is an open segment, and then by 2,3, and 5(c), $\dot{u}_i = 0$.

□

We now define the tester $\mathcal{T}_{p_1 \mathcal{S} p_2} = (S, \underline{s}, P, Q, C, Inv, \Delta, \lambda, \gamma, \mathcal{F})$ that realizes the characteristic function of the formula $p_1 \mathcal{S} p_2$, consisting of

- $S = \{s_0, s_1, s_2, s_3\}$
- $P = \{p_1, p_2\}$ and $Q = \{q\}$
- $C = \emptyset$
- $\Delta = \{\delta_1, \ldots, \delta_{20}\}$
- $\mathcal{F} = \emptyset$
- The following input and output labels, and invariants for every state

$$\begin{array}{ccc} \lambda(s_0) = p_1 \wedge p_2 & \gamma(s_0) = q & Inv(s_0) = \top \\ \lambda(s_1) = p_1 \wedge \neg p_2 & \gamma(s_1) = q & Inv(s_1) = \top \\ \lambda(s_2) = \neg p_1 & \gamma(s_2) = \neg q & Inv(s_2) = \top \\ \lambda(s_3) = p_1 \wedge \neg p_2 & \gamma(s_3) = \neg q & Inv(s_3) = \top \end{array}$$

- In the following table we can see at the same time all the edges in Δ and their respective input and output labels

i	δ_i	$\lambda(\delta_i)$	$\gamma(\delta_i)$
1	$(\underline{s}, \top, \emptyset, s_0)$	$\top$	$\neg q$
2	$(\underline{s}, \top, \emptyset, s_1)$	p_2	$\neg q$
3	$(\underline{s}, \top, \emptyset, s_2)$	$\top$	$\neg q$
4	$(\underline{s}, \top, \emptyset, s_3)$	$\neg p_2$	$\neg q$
5	$(s_0, \top, \emptyset, s_0)$	$\neg p_1 \vee \neg p_2$	q
6	$(s_0, \top, \emptyset, s_1)$	$p_1 \vee p_2$	q
7	$(s_0, \top, \emptyset, s_2)$	$\top$	q
8	$(s_0, \top, \emptyset, s_3)$	$\neg p_1 \wedge \neg p_2$	q
9	$(s_1, \top, \emptyset, s_0)$	$\top$	q
10	$(s_1, \top, \emptyset, s_1)$	p_2	q
11	$(s_1, \top, \emptyset, s_2)$	$\top$	q
12	$(s_1, \top, \emptyset, s_3)$	$\neg p_1 \wedge \neg p_2$	q
13	$(s_2, \top, \emptyset, s_0)$	$\top$	$\neg q$
14	$(s_2, \top, \emptyset, s_1)$	p_2	$\neg q$
15	$(s_2, \top, \emptyset, s_2)$	p_1	$\neg q$
16	$(s_2, \top, \emptyset, s_3)$	$\neg p_2$	$\neg q$
17	$(s_3, \top, \emptyset, s_0)$	$\top$	$\neg q$
18	$(s_3, \top, \emptyset, s_1)$	p_2	$\neg q$
19	$(s_3, \top, \emptyset, s_2)$	$\top$	$\neg q$
20	$(s_3, \top, \emptyset, s_3)$	$\neg p_1 \wedge \neg p_2$	$\neg q$

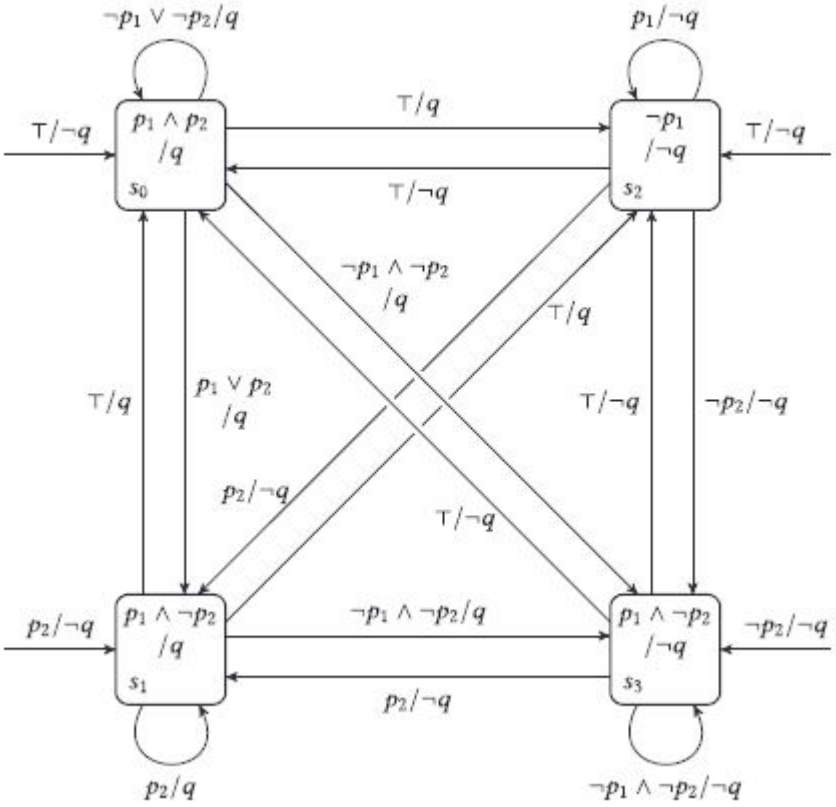


Figure 3.5.1: Temporal tester for $p_1 \mathcal{S} p_2$

The temporal tester $\mathcal{T}_{p_1 \mathcal{S} p_2}$ for operator *since* with input p_1, p_2 and output q is depicted in Figure 3.5.1. In the following theorem we prove that $\mathcal{T}_{p_1 \mathcal{S} p_2}$ computes $\chi_{p_1 \mathcal{S} p_2}$ by proving that every $\mathcal{T}_{p_1 \mathcal{S} p_2}$-output u over a signal w equals $\chi_{p_1 \mathcal{S} p_2}(w)$.

Theorem 3.5.2. *The transducer $\mathcal{T}_{p_1 \mathcal{S} p_2}$ is functional. Moreover, for every signal w over p_1, p_2, we have*

$$\chi_{p_1 \mathcal{S} p_2}(w) = \mathcal{T}_{p_1 \mathcal{S} p_2}(w)$$

Proof. Let w be an arbitrary signal over p_1, p_2. To see that there exists an accepting run of $\mathcal{T}_{p_1 \mathcal{S} p_2}$ over w we use the fact that from any configuration (s, v) of $\mathcal{T}_{p_1 \mathcal{S} p_2}$, a time or discrete transition reading some point segment of w can be taken, followed by a time step (or an "infinite time" step where the automaton reads an infinite open segment). Let us fix some partition $\mathcal{I}$ compatible with w. Without loss of generality, we assume that for all $i \geq 0$, we have that if $w_i = \dot{w}_i$, then $w_i \neq w_{i+1}$. Let $i \geq 0$ and consider $\dot{w}_i, w_i$.

- Suppose $s = \underline{s}$, then $i = 0$ and
  - If $\dot{w}_0 \vDash p_2$, then transitions $\delta_1, \delta_2, \delta_3$ are allowed. These transitions lead to states s_0, s_1, s_2, and then a time step can be performed for any w_0.
  - If $\dot{w}_0 \vDash \neg p_2$, then transitions $\delta_1, \delta_3, \delta_4$ are allowed. These transitions lead to states s_0, s_2, s_3, and then a time step can be performed for any w_0.
- Suppose $s = s_0$, then
  - If $\dot{w}_i \vDash p_1 \wedge p_2$, transitions δ_6, δ_7 are allowed. These transitions lead to states s_1, s_2, and then a time step can be performed for any value of w_i, unless for $w_i \vDash p_1 \wedge p_2$, but by hypothesis $w_i \nvDash p_1 \wedge p_2$.
  - If $\dot{w}_i \vDash p_1 \wedge \neg p_2$, then transitions $\delta_5, \delta_6, \delta_7$ are allowed. These transitions lead to states s_0, s_1, s_2, and then a time step can be performed for any w_i.
  - If $\dot{w}_i \vDash \neg p_1 \wedge p_2$, then transitions $\delta_5, \delta_6, \delta_7$ are allowed. These transitions lead to states s_0, s_1, s_2, and then a time step can be performed for any w_i.
  - If $\dot{w}_i \vDash \neg p_1 \wedge \neg p_2$, then transitions $\delta_5, \delta_7, \delta_8$ are allowed. These transitions lead to states s_5, s_7, s_8, and then a time step can be performed for any w_i.
- Cases $s = s_1, s_2, s_3$ are similar.

Furthermore, since $\mathcal{F} = \emptyset$, every run is accepting.

Now consider an $\mathcal{T}_{p_1 \mathcal{S} p_2}$-output over w. This means that there is an accepting run

$$(\underline{s}, 0) \xrightarrow[\delta_0]{\dot{w}_0/\dot{u}_0} (s_0, v_0) \xrightarrow{w_0/u_0} (s_0, v_0 + r_0) \xrightarrow[\delta_1]{\dot{w}_1/\dot{u}_1} (s_1, v_1) \xrightarrow{w_1/u_1} (s_1, v_1 + r_1) \dots,$$

the output of $\mathcal{T}_{p_1 \mathcal{S} p_2}$ over w inducing u such that $\dot{w}_0 \cdot w_0^{r_0} \cdot w_1 \cdot w_1^{r_1} \cdots$ and $\dot{u}_0 \cdot u_0^{r_0} \cdot \dot{u}_1 \cdot u_1^{r_1} \cdots$. We show that for all $i \geq 0$,

- $\dot{u}_0 = 0$.
  The first step performed by the transducer is a discrete transition coming from $\underline{s}$, i.e. δ_i for $i = 1, 2, 3, 4$. But observe that, $\gamma(\delta_i) = \neg q$ for all $i = 1, 2, 3, 4$. Hence, $\dot{u}_0 = 0$.
- $u_i = \dot{u}_{i+1}$.
  We have to show that for every s and every transition $\delta = (s, g, Z, s')$ we have $\gamma(s) = \gamma(\delta)$. For $s = s_0$ we have $\gamma(s_0) = q$ and $\gamma(\delta_j) = q$ for $j = 5, 6, 7, 8$. For s_1, we have $\gamma(s_1) = q$ and $\gamma(\delta_j) = q$ for $j = 9, 10, 11, 12$. For s_2, we have $\gamma(s_2) = \neg q$ and $\gamma(\delta_j) = \neg q$ for $j = 13, 14, 15, 16$. For s_3, we have $\gamma(s_3) = \neg q$ and $\gamma(\delta_j) = \neg q$ for $j = 17, 18, 19, 20$.

- if $w_i \vDash \neg p_1$, then $u_i = 0$.
  At step i the transducer takes a time step at state s_1 which is the only one satisfying $\lambda(s) = \neg p_1$, whence $u_i \vDash \gamma(s_1) = \neg q$. Hence $u_i = 0$.

- if $w_i \vDash p_1 \wedge p_2$, then $u_i = 1$.
  At step i the transducer takes a time step at state s_0 which is the only one satisfying $\lambda(s) = p_1 \wedge p_2$, whence $u_i \vDash \gamma(s_0) = q$. Hence $u_i = 1$.

- if $w_i \vDash p_1 \wedge \neg p_2$, there are three possibilities

  1. if $\dot{w}_i \vDash \neg p_1 \wedge \neg p_2$ then $u_i = 0$.
     For every s, if $\delta = (s, g, R, s')$ satisfies $\lambda(\delta) = \neg p_1 \wedge \neg p_2$ and $\lambda(s') \vDash p_1 \wedge \neg p_2$, then $\delta = \delta_j$ for $j = 4, 8, 12, 16, 20$, and $s' = s_3$. Hence $u_i \vDash \gamma(s_3) = \neg q$, and $u_i = 0$.
  2. if $\dot{w}_i \vDash p_2$, then $u_i = 1$.
     For every s, if $\delta = (s, g, R, s')$ satisfies $\lambda(\delta) = p_2$ and $\lambda(s') \vDash p_1 \wedge \neg p_2$, then $\delta = \delta_j$ for $j = 2, 6, 10, 14, 18$, and $s' = s_1$. Hence $u_i \vDash \gamma(s_3) = q$, and $u_i = 1$.
  3. if $\dot{w}_i \vDash p_1 \wedge \neg p_2$, then $u_i = \dot{u}_i$.
     For every s, if $\delta = (s, g, R, s')$ satisfies $\lambda(\delta) = p_1 \wedge \neg p_2$ and $\lambda(s') \vDash p_1 \wedge \neg p_2$, then $\delta = \delta_j$ for $j = 4, 6, 16$. For δ_4, we have $\gamma(\delta_4) = \neg q = \gamma(s_3)$, whence $u_i = \dot{u}_i = 0$. For δ_6, we have $\gamma(\delta_6) = q = \gamma(s_1)$, whence $u_i = \dot{u}_i = 1$. For δ_{16}, we have $\gamma(\delta_{16}) = \neg q = \gamma(s_3)$, whence $u_i = \dot{u}_i = 0$.

Then by Theorem 3.5.1, $u = \chi_{p_1 \mathcal{S} p_2}(w)$, and then u is unique. □

Theorem 3.5.1 can easily be adapted to hold true for bounded signals by bounding the index i of the t_i and considering time points of the domain of w. Moreover, we have observed that for all w, t, we have $(w, t) \vDash p_1 \mathcal{S} p_2$ iff $(w_{|[0,t]}, t) \vDash p_1 \mathcal{S} p_2$. Hence, the temporal tester $\mathcal{T}_{p_1 \mathcal{S} p_2}$ also realizes the characteristic function $\chi_{p_1 \mathcal{S} p_2}$ over bounded signals and we get the corresponding theorem which can be proved analogously.

Theorem 3.5.3. *The transducer $\mathcal{T}_{p_1 \mathcal{S} p_2}$ is functional. Moreover, for every bounded signal w over p_1, p_2, we have*

$$\chi_{p_1 \mathcal{S} p_2}(w) = \mathcal{T}_{p_1 \mathcal{S} p_2}(w).$$

3.5.2 Strict Until

To construct a temporal tester for $p_1 \mathcal{U} p_2$ for arbitrary different propositions p_1, p_2, we analize the semantics of $\mathcal{U}$ and describe some of its properties. We know that $(w, t) \vDash p_1 \mathcal{U} p_2$ if and only if there is some $t' > t$ such that $(w, t') \vDash p_2$ and for all $t'' \in (t, t'), (w, t'') \vDash p_1$. Let us begin by pointing out the following property.

Lemma 3.5.2. *For every signal w, time instant t and formulas φ_1, φ_2, if $(w,t) \vDash \varphi_1 \mathcal{U} \varphi_2$, then there exists $t' > t$ such that for all $t'' \in (t,t')$, $(w,t'') \vDash \varphi_1 \mathcal{U} \varphi_2$. In other words, for every signal w and formulas φ_1, φ_2, the characteristic function $\chi_{\varphi_1 \mathcal{U} \varphi_2}(w)$ is* right-continuous.

Proof. Let w be an arbitrary signal and $u = \chi_{\varphi_1 \mathcal{U} \varphi_2}(w)$. Then u can be decomposed as a concatenation of the form $\dot{u}_0 \cdot u_0^{r_0} \cdot \dot{u}_1 \cdot u_1^{r_1} \cdots$ where $r_i = t_i - t_{i-1}$ for a point-segment partition $\mathcal{I} = \{0\}(0,t_0)\{t_0\}(t_0,t_1)\cdots$. In this sense, right-continuity is equivalent to showing for every $i \geq 0$ that $\dot{u}_i(0) = u_i^{r_i}(t)$ for all $t \in (0, r_i)$. First, assume that $\dot{u}_i(0) = 1$. Then there exists $t > t_i$ such that φ_2 is satisfied at t and φ_1 holds continuously throughout the interval (t_i, t). As a consequence $p_1 \mathcal{U} p_2$ holds along (t_i, t), whence $u_i^{r_i}(t) = 1$ for all $t \in (0, r_i)$. If $\dot{u}_i(0) = 0$, then we have two possibilities, (i) either φ_2 never becomes true at any $t > t_i$ and then $u(t) = 0$ for every $t > t_i$, or (ii) for any $t > t_i$ where φ_2 is true, there is some t' between t_i and t where φ_1 is false, which implies that $\varphi_1 \mathcal{U} \varphi_2$ is false over (t_i, t'), and then $u_i^{r_i}(t) = 0$ for all $t \in (0, r_i)$. □

We have the corresponding result for bounded signals.

Lemma 3.5.3. *For every bounded signal w of domain D, time instant $t < \sup D$ and formulas φ_1, φ_2, if $(w,t) \vDash \varphi_1 \mathcal{U} \varphi_2$, then there exists $t' > t$ such that for all $t'' \in (t,t')$, $(w,t'') \vDash \varphi_1 \mathcal{U} \varphi_2$. In other words, for every signal w and formulas φ_1, φ_2, the characteristic function $\chi_{\varphi_1 \mathcal{U} \varphi_2}(w)$ is* right-continuous.

In the next theorem, we characterise the semantics of *until* operator over unbounded signals.

Theorem 3.5.4. *For every boolean signal w over p_1, p_2 and every 1-boolean signal u over q, we have*

$$u = \chi_{p_1 \mathcal{U} p_2}(w)$$

if and only if there exists a point-segment partition $\mathcal{I} = \{t_0\}(t_0,t_1)\{t_1\}(t_1,t_2)\cdots$ compatible with w and u such that $w = \dot{w}_0 \cdot w_0^{r_0} \cdot \dot{w}_1 \cdot w_1^{r_1} \cdots$ and $u = \dot{u}_0 \cdot u_0^{r_0} \cdot \dot{u}_1 \cdot u_1^{r_1} \cdots$ and such that for all $i \geq 0$,

1. *$\dot{u}_i = u_i$,*
2. *if $w_i \vDash \neg p_1$, then $u_i = 0$,*
3. *if $w_i \vDash p_1 \wedge p_2$, then $u_i = 1$,*
4. *if $w_i \vDash p_1 \wedge \neg p_2$, then w_i is the last segment of w and $u_i = 0$, or this is not the case, and*
   - *(a) if $\dot{w}_{i+1} \vDash \neg p_1 \wedge \neg p_2$, then $u_i = 0$,*
   - *(b) if $\dot{w}_{i+1} \vDash p_2$, then $u_i = 1$,*
   - *(c) if $\dot{w}_{i+1} \vDash p_1 \wedge \neg p_2$, then $u_i = \dot{u}_{i+1}$.*

Proof. ($\Rightarrow$) Let w be an arbitrary signal. If $u = \chi_{p_1 \mathcal{U} p_2}(w)$, then u is a signal and satisfies these properties by definition. Consider the point-segment partition $\mathcal{I}$ induced by the discontinuities of w and u. Then u and w can be decomposed with respect to $\mathcal{I}$. Moreover,

1. Let $i \geq 0$. If $\dot{w}_i \vDash p_1 \mathcal{U} p_2$ then by right-continuity of $\chi_{p_1 \mathcal{U} p_2}(w)$ there exists some $t' > t_i$ such that $\forall t'' \in (t_i, t')$, $(w, t'') \vDash p_1 \mathcal{U} p_2$. Hence, since $(t_i, t') \cap (t_i, t_{i+1}) \neq \emptyset$, then we obtain $w_i \vDash p_1 \mathcal{U} p_2$. Conversely, if $w_i \nvDash p_1 \mathcal{U} p_2$, then for all $t > t_i$ such that $(w, t) \vDash p_1 \mathcal{U} p_2$, there exists $t' \in (t_i, t)$ such that $(w, t') \nvDash p_1 \mathcal{U} p_2$ and by right-continuity $\dot{w}_i \nvDash p_1 \mathcal{U} p_2$.

2. If $w_i \vDash \neg p_1$, then $\forall t' > t_i$ where p_2 holds, there exists $t'' \in (t_i, t')$ such that $(w, t'') \nvDash \neg p_1$, and then $w_i \nvDash p_1 \mathcal{U} p_2$. Hence, $u_i = 0$.

3. If $w_i \vDash p_1 \wedge p_2$, then $\exists t \in (t_i, t_{i+1})$ such that $(w, t) \vDash p_2$ and $\forall t'' \in (t_i, t)$, $(w, t'') \vDash p_1$. Hence $(w, t_i) \vDash p_1 \mathcal{U} p_2$, or equivalently, $u_i = 1$.

4. If $w_i \vDash p_1 \wedge \neg p_2$, then either w_i is the last segment in w and for every $t > t_i$, $(w, t) \vDash \neg p_2$, whence $(w, t) \nvDash p_1 \mathcal{U} p_2$, or

   (a) if $\dot{w}_{i+1} \vDash \neg p_1 \wedge \neg p_2$, then for all $t > t_i$ such that $(w, t) \vDash p_2$, there exists some $t' = t_{i+1} \in (t_i, t)$ such that $(w, t') \nvDash p_1$, and then $w_i \nvDash p_1 \mathcal{U} p_2$, whence $u_i = 0$.

   (b) if $\dot{w}_{i+1} \vDash p_2$, then for every $t \in (t_i, t_{i+1})$, we have that there exists $t' = t_{i+1} > t$ satisfying p_2 and for every $t'' \in (t, t')$, p_1 holds true. Then $(w, t) \vDash p_1 \mathcal{U} p_2$, whence $u_i = 1$.

   (c) if $\dot{w}_{i+1} \vDash p_1 \wedge \neg p_2$, we consider the point or open segment of w with minimum index, greater than $i+1$, where $p_1 \wedge \neg p_2$ doesn't hold. If it is a point segment where p_2 holds, then $u_i = \dot{u}_{i+1} = 1$; if $\neg p_1 \wedge \neg p_2$ holds, then $u_i = \dot{u}_{i+1} = 0$. If it is an open segment where $p_1 \wedge p_2$ holds, then $u_i = \dot{u}_{i+1} = 1$, but if $\neg p_1$ holds, then $u_i = \dot{u}_{i+1} = 0$.

($\Leftarrow$) Suppose that there exists such an u. We have to prove that $u = \chi_{p_1 \mathcal{U} p_2}(w)$ i.e. that for all $t \geq 0$, $u(t) = 1$ iff $(w, t) \vDash p_1 \mathcal{U} p_2$. Let $t \geq 0$, and let us show both directions of the equivalence, one directly and the other by contraposition. We have two cases, $t \in (t_i, t_{i+1})$ or $t = t_i$ for some $i \in \mathbb{N}$.

Case $t \in (t_i, t_{i+1})$ for some $i \geq 0$.

- Suppose $(w, t) \vDash p_1 \mathcal{U} p_2$. Then by definition either
  (a) $(w, t) \vDash p_1 \wedge p_2$, and then by 3, $u_i = 1$, or
  (b) $(w, t) \vDash p_1 \wedge \neg p_2$, w_i is not the last segment of w, and the first point or open segment where $p_1 \wedge \neg p_2$ is false satisfies: (i) $p_1 \wedge p_2$ if it is an open segment or (ii) p_2 if it is a point segment.

  In case (b), if the index is $j = i + 1$, then by 4(b), $u_i = 1$. Otherwise, $j > i + 1$. If it is a point segment t_j, then $w_{j-1} \vDash p_1 \wedge \neg p_2$ and $\dot{w}_j \vDash p_2$, whence by 4(b) $u_{j-1} = 1$. But then, by 4(c) and 1, $u_i = \dot{u}_{i+1} = u_{i+1} = \cdots = u_{j-1} = 1$. If it is an open segment (t_j, t_{j+1}) for some j, then

$\dot{w}_j \vDash p_1 \wedge \neg p_2$ and $w_j \vDash p_1 \wedge p_2$, whence by 3 $u_j = 1$. But then, by 4(c) and 1, $u_i = \dot{u}_{i+1} = u_{i+1} = \cdots = u_j = 1$.

- Suppose $(w,t) \nvDash p_1 \mathcal{U} p_2$. Then by definition either
  (a) $(w,t) \vDash \neg p_1$, and then by 2, $u_i = 0$ or
  (b) $(w,t) \vDash p_1 \wedge \neg p_2$ and (i) w_i is the last segment in w, or (ii) the first point or open segment such that $p_1 \wedge \neg p_2$ is false satisfies: (i) $\neg p_1$ if it is an open segment or (ii) $\neg p_1 \wedge \neg p_2$ if it is a point segment.

  In case (b), if w_i is the last segment of w, by 4, $u_i = 0$. Otherwise, consider such a segment. If it is t_{i+1}, then by 4(a), $u_i = 0$. Otherwise, if the first segment is a point segment t_j, then $w_{j-1} \vDash p_1 \wedge \neg p_2$ and $\dot{w}_j \vDash \neg p_1 \wedge \neg p_2$, whence by 4(b) $u_{j-1} = 1$. But then, by 4(c) and 1, $u_i = u_{i+1} = \dot{u}_{i+1} = \cdots = u_{j-1} = 1$. If it is an open segment (t_j, t_{j+1}), then $\dot{w}_j \vDash p_1 \wedge \neg p_2$ and $w_j \vDash \neg p_1$, whence by 2 $u_j = 1$. But then, by 4(c) and 1, $u_i = \dot{u}_{i+1} = u_{i+1} = \cdots = u_j = 1$.

Case $t = t_i$ for some $i \geq 0$.

- If $i \geq 1$ and $\dot{w}_i \vDash p_1 \mathcal{U} p_2$, then by definition either
  (a) $w_i \vDash p_1 \wedge p_2$, and then by 3 and 1, $\dot{u}_i = 1$, or
  (b) $w_i \vDash p_1 \wedge \neg p_2$, and either the point or open segment of w with minimum index, greater than $i+1$, where $p_1 \wedge \neg p_2$ doesn't hold satisfies (i) p_2 if it is a point segment, and then by 4(b), 4(c), 1 and 2, $\dot{u}_i = 1$, or (ii) $p_1 \wedge p_2$ if it is an open segment, and then by 3, 1 and 4(c), $\dot{u}_i = 1$.

- If $\dot{w}_i \nvDash p_1 \mathcal{U} p_2$, then by definition either
  (a) $w_i \vDash \neg p_1$, and then by 1 and 3, $\dot{u}_i = 0$ or
  (b) $w_i \vDash p_1 \wedge \neg p_2$ and either the point or open segment of w with maximum index, greater than $i+1$, where $p_1 \wedge \neg p_2$ doesn't hold satisfies (i) $\neg p_1 \wedge \neg p_2$ if it is a point segment, and then by 4(a), 4(c) and 1, $\dot{u}_i = 0$, or (ii) $\neg p_1$ if it is an open segment, and then by 1,2, and 4(c), $\dot{u}_i = 0$.

□

For bounded signals, we get the following characterisation.

Theorem 3.5.5. *For every bounded boolean signal w over p_1, p_2 and every bounded 1-boolean signal u over q, we have*

$$u = \chi_{p_1 \mathcal{U} p_2}(w)$$

if and only if there exists a finite point-segment partition $\mathcal{I} = \{t_0\}(t_0,t_1)\{t_1\}\cdots$ ending at some (t_j, t_{j+1}) compatible with w and u such that $w = \dot{w}_0 \cdot w_0^{r_0} \cdot \dot{w}_1 \cdots$ and $u = \dot{u}_0 \cdot u_0^{r_0} \cdot \dot{u}_1 \cdots$ and such that for all $0 \leq i \leq j$,

1. *$\dot{u}_i = u_i$,*
2. *if $w_i \vDash \neg p_1$, then $u_i = 0$,*
3. *if $w_i \vDash p_1 \wedge p_2$, then $u_i = 1$,*

4. *if* $w_i \vDash p_1 \wedge \neg p_2$, *then either* $i = j$ *and it is the last segment where* $u_j = 0$, *or this is not the case and*

   (a) *if* $\dot{w}_{i+1} \vDash \neg p_1 \wedge \neg p_2$, *then* $u_i = 0$,

   (b) *if* $\dot{w}_{i+1} \vDash p_2$, *then* $u_i = 1$,

   (c) *if* $\dot{w}_{i+1} \vDash p_1 \wedge \neg p_2$, *then* $u_i = \dot{u}_{i+1}$.

For unbounded signals, we now define the temporal tester $\mathcal{T}_{p_1 \mathcal{U} p_2} = (S, \underline{s}, P, Q, C, Inv, \Delta, \lambda, \gamma, \mathcal{F})$ that computes the characteristic function of the formula $p_1 \mathcal{U} p_2$ for any $p_1 \neq p_2 \in P$, consisting of

- $S = \{s_0, s_1, s_2, s_3\}$
- $P = \{p_1, p_2\}$ and $Q = \{q\}$
- $C = \emptyset$
- $\Delta = \{\delta_1, \ldots, \delta_{20}\}$
- $\mathcal{F} = \{F_1\}$ where $F_1 = (S \cup \Delta) \backslash \{s_1\}$
- The states have the following input and output labels, and invariants

$$\begin{array}{ccc} \lambda(s_0) = p_1 \wedge p_2 & \gamma(s_0) = q & Inv(s_0) = \top \\ \lambda(s_1) = p_1 \wedge \neg p_2 & \gamma(s_1) = q & Inv(s_1) = \top \\ \lambda(s_2) = \neg p_1 & \gamma(s_2) = \neg q & Inv(s_2) = \top \\ \lambda(s_3) = p_1 \wedge \neg p_2 & \gamma(s_3) = \neg q & Inv(s_3) = \top \end{array}$$

- In the following table we can see at the same time all the edges in Δ and their respective input and output labels

i	δ_i	$\lambda(\delta_i)$	$\gamma(\delta_i)$
1	$(\underline{s}, \top, \emptyset, s_0)$	$\top$	q
2	$(\underline{s}, \top, \emptyset, s_1)$	$\top$	q
3	$(\underline{s}, \top, \emptyset, s_2)$	$\top$	$\neg q$
4	$(\underline{s}, \top, \emptyset, s_3)$	$\top$	$\neg q$
5	$(s_0, \top, \emptyset, s_0)$	$\neg p_1 \vee \neg p_2$	q
6	$(s_0, \top, \emptyset, s_1)$	$\top$	q
7	$(s_0, \top, \emptyset, s_2)$	$\top$	$\neg q$
8	$(s_0, \top, \emptyset, s_3)$	$\top$	$\neg q$
9	$(s_1, \top, \emptyset, s_0)$	$p_1 \vee p_2$	q
10	$(s_1, \top, \emptyset, s_1)$	p_2	q
11	$(s_1, \top, \emptyset, s_2)$	p_2	$\neg q$
12	$(s_1, \top, \emptyset, s_3)$	p_2	$\neg q$
13	$(s_2, \top, \emptyset, s_0)$	$\top$	q
14	$(s_2, \top, \emptyset, s_1)$	$\top$	q
15	$(s_2, \top, \emptyset, s_2)$	p_1	$\neg q$
16	$(s_2, \top, \emptyset, s_3)$	$\top$	$\neg q$
17	$(s_3, \top, \emptyset, s_0)$	$\neg p_1 \wedge \neg p_2$	q
18	$(s_3, \top, \emptyset, s_1)$	$\neg p_1 \wedge \neg p_2$	q
19	$(s_3, \top, \emptyset, s_2)$	$\neg p_2$	$\neg q$
20	$(s_3, \top, \emptyset, s_3)$	$\neg p_1 \wedge \neg p_2$	$\neg q$

The temporal tester $\mathcal{T}_{p_1 \mathcal{U} p_2}$ for operator *until* with input p_1, p_2 and output q is depicted in Figure 3.5.2. It can be obtained by inverting the tester for *since* operator with some additional modifications. As opposed to the tester for *since*, this tester for *until* has to predict an output at every step, which is eventually aborted or confirmed according to future observations. In the following theorem we prove that $\mathcal{T}_{p_1 \mathcal{U} p_2}$ mimicks the behaviour of $\chi_{p_1 \mathcal{U} p_2}$ by proving that every $\mathcal{T}_{p_1 \mathcal{U} p_2}$-output u over a signal w corresponds to $\chi_{p_1 \mathcal{U} p_2}(w)$.

Theorem 3.5.6. *The transducer $\mathcal{T}_{p_1 \mathcal{U} p_2}$ is functional. Moreover, for every signal w over p_1, p_2,*

$$\chi_{p_1 \mathcal{U} p_2}(w) = \mathcal{T}_{p_1 \mathcal{U} p_2}(w)$$

Proof. Let w be an arbitrary signal over p_1, p_2. First, we show that there exists a $\mathcal{T}_{p_1 \mathcal{U} p_2}$-output over w and secondly we prove that every $\mathcal{T}_{p_1 \mathcal{U} p_2}$-output over w coincides with $\chi_{p_1 \mathcal{U} p_2}(w)$. For the second part, we use the characterisation shown in Theorem 3.5.4.

A $\mathcal{T}_{p_1 \mathcal{U} p_2}$-output over w exists iff there exists an accepting run of $\mathcal{T}_{p_1 \mathcal{U} p_2}$ over w. To prove this last assertion, we show that every partial run over w can be extended. Let us fix some decomposition $\dot{w}_0 \cdot w_0^{r_0} \cdot \dot{w}_1 \cdot w_1^{r_1} \cdots$ of w. We assume without loss of generality that for all $i \geq 0$, if $w_i = \dot{w}_i$, then $w_i \neq w_{i+1}$. Suppose our partial run ends in configuration (s, v) having read w until some point segment $\dot{w}_i$. Then, we must check whether $\mathcal{T}_{p_1 \mathcal{U} p_2}$ can take a discrete step reading $\dot{w}_i$ for every $\dot{w}_i$, which can be followed by some time step. If $i = 0$, then it is easy to see that $\mathcal{T}_{p_1 \mathcal{U} p_2}$ can take transitions δ_j for $j = 1, 2, 3, 4$ as every $\dot{w}_0$ satisfies $\top$ at 0. Moreover, a time step can be taken if $\exists s, \exists i$ such

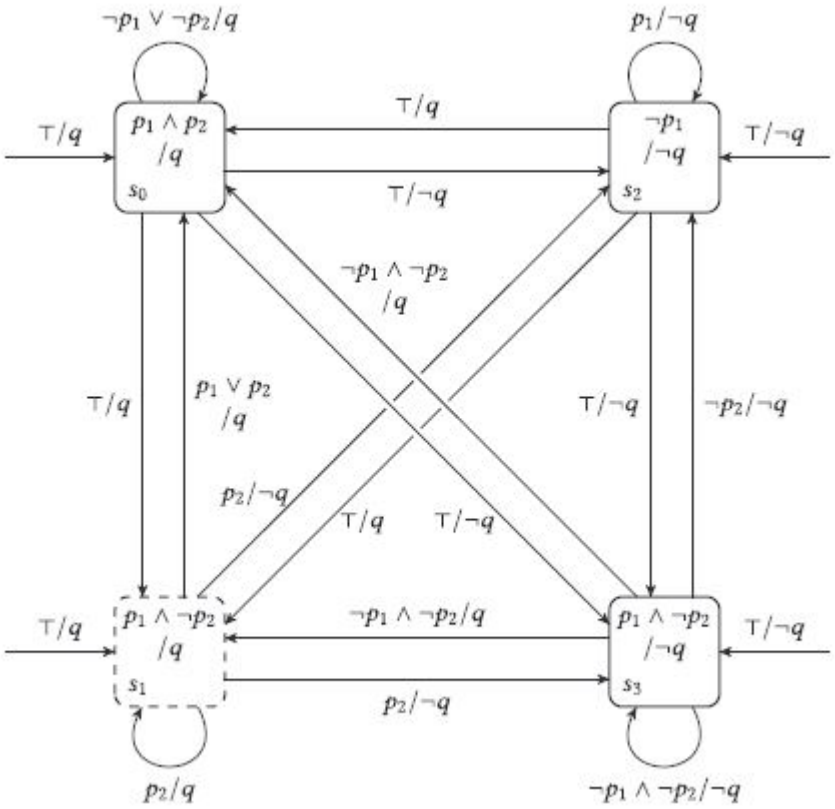


Figure 3.5.2: Temporal tester for $p_1 \mathcal{U} p_2$

that $\delta_i = (\underline{s}, g, R, s)$ and $w_0 \vDash \lambda(s)$, which is clearly true. If $i > 0$, we consider every possibility and conclude that the only cases where $\mathcal{T}_{p_1 \mathcal{U} p_2}$ cannot perform a transition are those where (i) $s = s_1$ and $\dot{w}_i \vDash \neg p_1 \wedge \neg p_2$ or (ii) $s = s_3$ and $\dot{w}_i \vDash p_2$. However, for every state s and every evaluation of p_1, p_2, there is a transition from s to s_1 iff there is a transition from s to s_3. Hence, in case (i), if our partial run r ends in s_1, we consider the partial run r' run coincing with r until step $i-1$, and then put a transition leading to s_3 instead of s_1, and then take a transition from state s_3. We perform a similar procedure for case (ii). By these observations, we can infer that there exists a run of $\mathcal{T}_{p_1 \mathcal{U} p_2}$ over w. Because every run is accepting unless it ends in state s_1 without taking any transition, and if there exists a partial run leading to s_1, there exists another partial run of the same duration and number of steps leading to s_3, there is an accepting run for every signal.

Now let u be an $\mathcal{T}_{p_1 \mathcal{U} p_2}$-output over w. Then u is the output of an accepting run over w. Moreover, there exist decompositions $\dot{w}_0 \cdot w_0^{r_0} \cdot w_1 \cdot w_1^{r_1} \cdots$ and $\dot{u}_0 \cdot u_0^{r_0} \cdot \dot{u}_1 \cdot u_1^{r_1} \cdots$ compatible with w and u respectively, and a corresponding

accepting run

$$(\underline{s},0) \xrightarrow[\delta_0]{\dot{w}_0/\dot{u}_0} (s_0,v_0) \xrightarrow{w_0^{r_0}/u_0^{r_0}} (s_0,v_0{+}r_0) \xrightarrow[\delta_1]{\dot{w}_1/\dot{u}_1} (s_1,v_1) \xrightarrow{w_1^{r_1}/u_1^{r_1}} (s_1,v_1{+}r_1)\cdots .$$

We show the following

- For every $i \geq 0$, $\dot{u}_i = u_i$. This follows because for every transition δ, if it is of the form $\delta = (s, g, R, s')$, then $\gamma(\delta) = \gamma(s')$. Indeed,

$$\begin{aligned}
\gamma(\delta_i) = \gamma(s_0) = q \quad & \text{for } i = 1, 5, 9, 13, 17 \\
\gamma(\delta_j) = \gamma(s_1) = q \quad & \text{for } j = 2, 6, 10, 14, 18 \\
\gamma(\delta_k) = \gamma(s_2) = \neg q \quad & \text{for } k = 3, 7, 11, 15, 19 \\
\gamma(\delta_l) = \gamma(s_3) = \neg q \quad & \text{for } l = 4, 8, 12, 16, 20
\end{aligned}$$

- For every $i \geq 0$, if $w_i \vDash \neg p_1$, then $u_i = 0$. In this case the automata takes a time transition. This amounts to show that for every state s such that $w_i \vDash \lambda(s)$, then $u_i = 0$. If $w_i \vDash \neg p_1$, then $s = s_2$ and then $u_i \vDash \gamma(s_2) = \neg q$, whence $u_i = 0$.

- For every $i \geq 0$, if $w_i \vDash p_1 \wedge p_2$, then $u_i = 1$. Indeed, if s is such that $\lambda(s) = p_1 \wedge p_2$, then $s = s_0$ and then $u_i \vDash \gamma(s_0) = q$. Hence $u_i = 1$.

- If $w_i \vDash p_1 \wedge \neg p_2$, then either w_i is the last segment in w and $u_i = 0$, or

  (i) if $\dot{w}_{i+1} \vDash \neg p_1 \wedge \neg p_2$, then $u_i = 0$,
  (ii) if $\dot{w}_{i+1} \vDash p_2$, then $u_i = 1$,
  (iii) if $\dot{w}_{i+1} \vDash p_1 \wedge \neg p_2$, then $u_i = \dot{u}_{i+1}$.

  First suppose w_i is the last segment and the automaton is at some state s such that $\lambda(s) = p_1 \wedge \neg p_2$. Then $s = s_1$ or $s = s_3$. But the run is accepting iff for all $t \geq 0$, there exists $t' > t$ such that the automaton is in a state or transition from $\mathcal{F}$. Since w_i is the last segment, the automaton is not allowed to take any transition from s_1 to itself. As $s_1 \notin F_1$, then $s = s_3$ and $u_i \vDash \gamma(s_3) = \neg q$. Hence $u_i = 0$.

  (i) We consider transitions $\delta = (s, g, Z, s')$ such that $\lambda(s) = p_1 \wedge \neg p_2$ and $\neg p_1 \wedge \neg p_2 \vDash \lambda(\delta)$. Then $\delta = \delta_i$ for $i = 17, 18, 19, 20$. Then $s = s_3$ and $u_i = \gamma(s_3) = \neg q$. Hence $u_i = 0$.
  (ii) We consider transitions $\delta = (s, g, Z, s')$ such that $\lambda(s) = p_1 \wedge \neg p_2$ and $p_2 \vDash \lambda(\delta)$. Then $\delta = \delta_i$ for $i = 9, 10, 11, 12$. Then $s = s_1$ and $u_i = \gamma(s_1) = q$.
  (iii) We consider transitions $\delta = (s, g, Z, s')$ such that $\lambda(s) = p_1 \wedge \neg p_2$ and $p_1 \wedge \neg p_2 \vDash \lambda(\delta)$. Then $s = s_1, s_3$ and $\delta = \delta_i$ for $i = 9, 19$. If $\delta = \delta_9$, then $u_i = \gamma(s_1) = q = \gamma(\delta_9) = \dot{u}_{i+1}$. If $\delta = \delta_{19}$, then $u_i = \gamma(s_3) = \neg q = \gamma(\delta_{19}) = \dot{u}_{i+1}$.

Therefore, by Theorem 3.5.4, $u = \chi_{p_1 \mathcal{U} p_2}$. Hence, u is the unique $\mathcal{T}_{p_1 \mathcal{U} p_2}$-output over w and we can conclude that $\mathcal{T}_{p_1 \mathcal{U} p_2}(w) = u$. □

For bounded signals, we define the temporal tester $\mathcal{T}^b_{p_1 \mathcal{U} p_2} = (S, \underline{s}, P, Q, C, Inv, \Delta, \lambda, \gamma, F)$ coinciding with $\mathcal{T}_{p_1 \mathcal{U} p_2}$ in every component and such that $F = \{s_0, s_2, s_3\}$ is the set of final states. Then we have

Theorem 3.5.7. *The transducer $\mathcal{T}^b_{p_1 \mathcal{U} p_2}$ is functional. Moreover, for every bounded signal w over p_1, p_2,*

$$\chi_{p_1 \mathcal{U} p_2}(w) = \mathcal{T}^b_{p_1 \mathcal{U} p_2}(w).$$

3.5.3 Eventually

For every signal w and time instant t, we have that $(w,t) \vDash \Diamond_{(0,a)} p$ iff there is some $t' \in (t, t+a)$ such that $(w,t') \vDash p$. Thus, the value of $\chi_{\Diamond_{(0,a)} p}(w)$ at each t depends on the value of p throughout the interval $(t, t+a)$. At intervals I where p holds then the formula is satisfied in $I \ominus (0,a)$ independently of the shape of I. When p becomes untrue, there are three possible output behaviours depending on the duration of the segment where p is false.

Theorem 3.5.8. *Let a be a positive rational number. For every signal w and every $t \geq 0$ we have that $(w,t) \vDash \Diamond_{(0,a)} p$ iff $\chi_{\Diamond_{(0,a)} p}(w)(t) = 1$ iff there exists some point-segment partition $\mathcal{I}$ compatible with $w, \chi_{\Diamond_{(0,a)} p}(w)$ and some of the following holds*

1. *$t \in [t_i, t_{i+1})$ for some i and $w_i \vDash p$,*
2. *$t \in [t_i, t_{i+1})$ for some i and*
   - *$w_i \nvDash p$,*
   - *$t_{i+1} - t_i < a$,*
   - *$\dot{w}_{i+1} \vDash p$ or $w_{i+1} \vDash p$*
3. *$t \in (t_i, t_{i+1})$, for some i, and*
   - *$w_i \nvDash p$,*
   - *$t_{i+1} - t_i = a$,*
   - *$\dot{w}_{i+1} \vDash p$ or $w_{i+1} \vDash p$,*
4. *$t \in (t_{i+1} - a, t_{i+1})$ for some i, and*
   - *$w_i \nvDash p$,*
   - *$t_{i+1} - t_i > a$,*
   - *$\dot{w}_{i+1} \vDash p$ or $w_{i+1} \vDash p$.*

Proof. By definition, if some of the conditions 1-4 holds, then $\Diamond_{(0,a)} p$ holds at time t. On the other hand, if $\chi_{\Diamond_{(0,a)} p}(w)(t) = 1$, then either $t = t_i$ or $t \in (t_i, t_{i+1})$ for some $i \geq 0$. If $t = t_i$, then either

- $w_i \vDash p$ and we are in case 1, or

- $w_i \not\models p$ but $t_{i+1} - t_i < a$ and $\dot{w}_{i+1} \models p$ or $w_{i+1} \models p$ as in case 2.

If $t \in (t_i, t_{i+1})$, then it is clear to see that either 1, 2, 3 or 4 holds. □

We now define the temporal tester $\mathcal{T}_{\Diamond_{(0,a)}p} = (S, \underline{s}, P, Q, C, Inv, \Delta, \lambda, \gamma, \mathcal{F})$ with input variable p realizing the characteristic function of the formula $\Diamond_{(0,a)}p$, consisting of

- $S = \{s_0, s_1, s_2, s_3\}$
- $P = \{p\}$ and $Q = \{q\}$
- $C = \{x\}$
- $\Delta = \{\delta_1, \ldots, \delta_{17}\}$
- $\mathcal{F} = \emptyset$
- The states have the following input and output labels, and invariants

$$\begin{array}{lll} \lambda(s_0) = p & \gamma(s_0) = q & Inv(s_0) = \top \\ \lambda(s_1) = \neg p & \gamma(s_1) = q & Inv(s_1) = x < a \\ \lambda(s_2) = \neg p & \gamma(s_2) = q & Inv(s_2) = x < a \\ \lambda(s_3) = \neg p & \gamma(s_3) = \neg q & Inv(s_3) = \top \end{array}$$

- In the following table we can see at the same time all the edges in Δ and their respective input and output labels

i	δ_i	$\lambda(\delta_i)$	$\gamma(\delta_i)$
1	$(\underline{s}, \top, \emptyset, s_0)$	$\top$	q
2	$(\underline{s}, \top, \{x\}, s_1)$	$\top$	$\neg q$
3	$(\underline{s}, \top, \{x\}, s_2)$	$\top$	q
4	$(\underline{s}, \top, \emptyset, s_3)$	$\top$	$\neg q$
5	$(s_0, \top, \emptyset, s_0)$	$\neg p$	q
6	$(s_0, \top, \{x\}, s_1)$	$\top$	$\neg q$
7	$(s_0, \top, \{x\}, s_2)$	$\top$	q
8	$(s_0, x < a, \emptyset, s_3)$	$\top$	$\neg q$
9	$(s_1, x = a, \emptyset, s_0)$	$\top$	q
10	$(s_1, x = a, \{x\}, s_1)$	p	$\neg q$
11	$(s_1, x = a, \{x\}, s_2)$	p	q
12	$(s_1, x = a, \emptyset, s_3)$	p	$\neg q$
13	$(s_2, x < a, \emptyset, s_0)$	$\top$	q
14	$(s_2, x < a, \{x\}, s_1)$	p	$\neg q$
15	$(s_2, x < a, \{x\}, s_2)$	p	q
16	$(s_2, x < a, \emptyset, s_3)$	p	$\neg q$
17	$(s_3, \top, \{x\}, s_1)$	$\neg p$	$\neg q$

The temporal tester $\mathcal{T}_{\Diamond_{(0,a)}p}$ depicted in Figure 3.5.3 reads point or open segments of a signal w where p or $\neg p$ holds. In open segments where p holds, the tester is at state s_0 and the output satisfies q throughout the segment. Singular points where p doesn't hold are ignored by the automaton, a fact reflected by transition δ_5. When the automaton reads a segment where p is false, it predicts the length of that segment by means of clock x and then aborts wrong predictions. The automaton can make three possible predictions:

1. A prediction that the length of the segment is *strictly less than* a is realized by transition δ_7 to state s_2 where the clock x is reset. When at state s_2, the delay until an input satisfying p has to be strictly smaller than a, i.e. one of $\delta_{13}, \delta_{14}, \delta_{15}, \delta_{16}$ has to be taken, as $\lambda(s_0) = p$ and $\lambda(\delta_{14}) = \lambda(\delta_{15}) = \lambda(\delta_{16}) = p$. Otherwise, the run is aborted.

2. A prediction that the length of the segment is *equal to* a is realized by transition δ_6 to state s_1 where the clock x is reset. The output at the beggining of that segment satisfies $\gamma(\delta_6)) = \neg q$ and during the segment it satisfies $\gamma(s_1) = q$. When at state s_1, the delay until an input satisfying p has to be exactly a, i.e. one of $\delta_9, \delta_{10}, \delta_{11}, \delta_{12}$ has to be taken. Observe that $\lambda(s_0) = p$ and $\lambda(\delta_{10}) = \lambda(\delta_{11}) = \lambda(\delta_{12}) = p$. Any other run is aborted.

3. A prediction that the length of the segment is *strictly greater than* a is realized by transition δ_8 to state s_3. The tester makes a nondeterministic guess of the time instant t whose distance to the end of the segment is exactly a. At that point it takes transition δ_{17} to s_1 and then behaves as in case 2. If the prediction of t is wrong, then the run is aborted.

Theorem 3.5.9. *The transducer $\mathcal{T}_{\Diamond_{(0,a)}p}$ is functional. Moreover, for every signal w, we have*

$$\chi_{\Diamond_{(0,a)}p}(w) = \mathcal{T}_{\Diamond_{(0,a)}p}(w)$$

Proof. Let w be a signal over p. The existence of a $\mathcal{T}_{\Diamond_{(0,a)}p}$-output over w is again a simple but lengthy check. A $\mathcal{T}_{\Diamond_{(0,a)}p}$-output over w exists iff there exists an accepting run of $\mathcal{T}_{\Diamond_{(0,a)}p}$ over w. To prove this last assertion, we restrict to one case and show that every partial run over w ending in state s_1 can be extended; other cases are similar. Let us fix some decomposition $\dot{w}_0 \cdot w_0^{r_0} \cdot \dot{w}_1 \cdot w_1^{r_1} \cdots$ of w. We assume without loss of generality that for all $i \geq 0$, if $w_i = \dot{w}_i$, then $w_i \neq w_{i+1}$. We only allow a decomposition $w_i = \dot{w}_i = w_{i+1}$ iff p is false at all of them and w_i has length $< a$ and w_{i+1} length equal to a.

Suppose our partial run ends in configuration (s_1, v) and has duration $t = r_0 + \cdots + r_{i-1}$ having read w until some point segment $\dot{w}_i$. Then, we must check whether $\mathcal{T}_{\Diamond_{(0,a)}p}$ can take a discrete step reading $\dot{w}_i$ for every $\dot{w}_i$, which can be followed by some time step reading $w_i^{r_i}$.

Because $Inv(s_1)$ requires that $x < a$, we now that the last step of the partial run was a time step of duration $\leq a$ where $\neg p$ held. If its duration was $< a$ then $v(x) < a$ at the time of taking a transition and no possible transition is

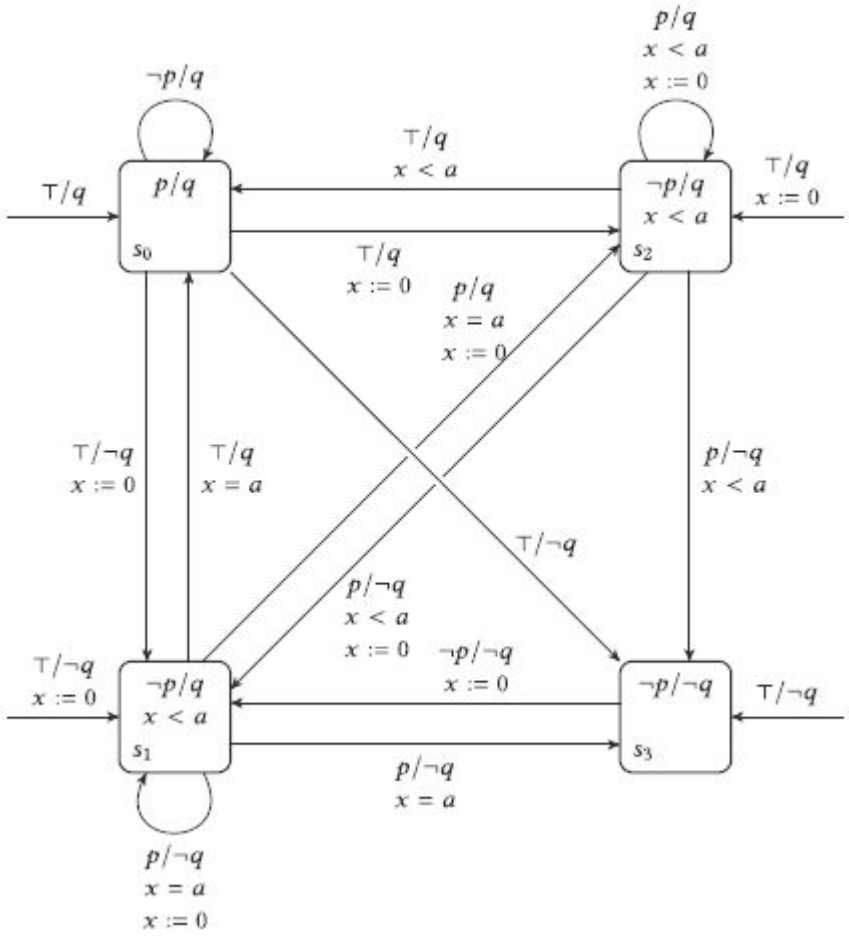


Figure 3.5.3: Tester for $\mathcal{T}_{\Diamond_{(0,a)}p}$

enabled. If $r_{i-1} = a$ then there exists another partial run which at this place takes a time step in s_1 of duration a. If $r_{i-1} < a$, argue as follows. Observe that in this case, by construction there exists another partial run until state s_2 that takes a time step of duration $< a$ (see that all guards of transitions from s_2 are $x < a$, and $\lambda(\delta_{13} = \top$ thus at least some discrete step is enabled). Because transitions δ to s_1 and to s_2 are always both allowed under the same conditions for states $\underline{s}, s_0, s_1, s_2$ (see δ_i for $i = 2, 3, 6, 7, 10, 11, 14, 15$). Only state who enables a transition to s_1 and not to s_2 under the same conditions is s_3 (see δ_{17}). But this happens where the automaton predicts that there is a compatible decomposition $w_{i-2} = \dot{w}_i = w_{i-1}$ where p is false at all of them and w_{i-2} has length $< a$ and w_{i-1} length equal to a. But this doesn't apply in our case as $r_{i-1} < a$.

If the time step was of duration a i.e. $r_{i-1} = a$, discrete transitions from s_1 are allowed (observe that $v(x) = a$ because transitions to state s_1 reset clock x, see δ_i for $i = 2, 6, 10, 14, 17$).

- Suppose $\dot{w}_i \vDash p$. Then any of the discrete transitions out of s_1 are allowed.
  - If $w_i^{r_i} \vDash p$, then take transition δ_9 and then a time step in s_0 of length r_i.
  - If $w_i^{r_i} \vDash \neg p$, then according to r_i, take transitions δ_{11} and then a time step of duration r_i if $r_i < a$, transition δ_{10} and a time step of duration r_i if $r_i = a$, or finally transition δ_{12} and then a time step of duration r_i if $r_i > a$.
- Suppose $\dot{w}_i \vDash \neg p$. Then transition δ_9 to state s_0 is allowed. From state s_0 we distinguish:
  - If $w_i^{r_i} \vDash p$ take a time step in s_0 of length r_i.
  - If $w_i^{r_i} \vDash \neg p$, then no time step is enabled since $\lambda(s_0)$ requires that p holds. But observe that, in this case, $w_{i-1}^{r_{i-1}} \vDash \neg p$, $\dot{w}_i \vDash \neg p$ and $w_i^{r_i} \vDash \neg p$. This means that $w_{i-1}, \dot{w}_i, w_i$ satisfy $\neg p$ but such a decomposition is excluded unless $r_{i-1} < a$ and $r_i = a$ which is not the case, as $r_{i-1} = a$ by assumption.

Suppose u is a $\mathcal{T}_{\Diamond_{(0,a)}p}$-output over w. We want to see that $u = \chi_{\Diamond_{(0,a)}p}(w)$. We show that if $\chi_{\Diamond_{(0,a)}p}(w)(t) = 1$, then $u(t) = 1$, and if $\chi_{\Diamond_{(0,a)}p}(w)(t) = 0$, then $u(t) = 0$.

Let us consider some $t \geq 0$, and suppose $\chi_{\Diamond_{(0,a)}p}(w)(t) = 1$. Then there is a point-segment partition compatible with $w, \chi_{\Diamond_{(0,a)}p}(w)$. Moreover, one of the four cases in Theorem 3.5.8 holds. We show that any case leads to $u(t) = 1$.

1. Case $t \in [t_i, t_{i+1})$ for some i and $w_i \vDash p$.
   - If $t = t_i$, we want to show that $\dot{u}_i = 1$. We look at discrete transitions $\delta = (s, g, Z, s')$ such that $\lambda(s') \vDash p$. Then $s' = s_0$ and $\delta = \delta_j$ for $j = 1, 5, 9, 13$ and $\gamma(\delta_j) = q$ for all j. Hence $\dot{u}_i = 1$.

- If $t \in (t_i, t_{i+1})$, we want to show that $u_i = q$. We look at time transitions starting in s such that $\lambda(s) = p$, and again $s = s_0$ and $u_i \vDash \gamma(s_0) = q$, whence $u_i = 1$.

2. Case $t \in [t_i, t_{i+1})$ for some i and $w_i \nvDash p$, $t_{i+1} - t_i < a$, $\dot{w}_{i+1} \vDash p$ or $w_{i+1} \vDash p$. We consider a segment of run

$$\cdots (s, v) \xrightarrow[\delta]{\dot{w}_i/\dot{u}_i} (s', v') \xrightarrow{w_i/u_i} (s', v' + r_i) \xrightarrow[\delta']{\dot{w}_{i+1}/\dot{u}_{i+1}} (s'', v'') \cdots$$

   - If $t = t_i$, we want to show that $\dot{u}_i = 1$. We look at discrete transitions $\delta = (s, g, Z, s')$ such that $\lambda(s') = \neg p$, followed by time transitions of duration t such that $v' + t \vDash Inv(s')$, followed by $\delta' = (s', g', Z', s'')$ such that $\lambda(\delta') = p$ or $\lambda(s'') = p$ and $x < a$ satisfies g'. If we look at these conditions, we realize that $s' = s_1, s_2, s_3$. But observe that all discrete transitions δ' coming from s_1 have $g' = (x = a)$. Then $s' \neq s_1$. Further, the only transition coming from s_3, $\delta_{17} = (s_3, \top, \{x\}, s_1)$ satisfies $\lambda(\delta_{17}) = \neg p$ and $\lambda(s_1) = \neg p$, and then $s' \neq s_3$. Then we conclude that $s' = s_2$ and we check that $\lambda(\delta) = q$ for all transitions δ ending in s_2. Indeed, $\lambda(\delta_i) = q$ for $i = 3, 7, 11, 15$. Then $\dot{u}_i \vDash q$ and $\dot{u}_i = 1$.
   - If $t \in (t_i, t_{i+1})$, looking at the discussion above, we know that $u_i \vDash \gamma(s') = \gamma(s_2) = q$, whence $u_i = 1$.

3. Case $t \in (t_i, t_{i+1})$ for some i and $w_i \nvDash p$, $t_{i+1} - t_i = a$, $\dot{w}_{i+1} \vDash p$ or $w_{i+1} \vDash p$. We consider a segment of run

$$\cdots (s, v) \xrightarrow[\delta]{\dot{w}_i/\dot{u}_i} (s', v') \xrightarrow{w_i/u_i} (s', v' + r_i) \xrightarrow[\delta']{\dot{w}_{i+1}/\dot{u}_{i+1}} (s'', v'') \cdots$$

   We look at discrete transitions $\delta = (s, g, Z, s')$ such that $\lambda(s') = \neg p$, followed by time transitions such that $v' + t \vDash Inv(s')$, followed by $\delta' = (s', g', Z', s'')$ such that $\lambda(\delta') = p$ or $\lambda(s'') = p$ and $x = a \vDash g'$. If we look at these conditions, we realize that $s' = s_1, s_2, s_3$. But observe that all discrete transitions δ' coming from s_2 have $g' = (x < a)$. Then $s' \neq s_2$. Further, the only transition coming from s_3, $\delta_{17} = (s_3, \top, \{x\}, s_1)$ satisfies $\lambda(\delta_{17}) = \neg p$ and $\lambda(s_1) = \neg p$, and then $s' \neq s_3$. In the case of s_1, all the outgoing transitions satisfy the condition since $\lambda(s_0) = \lambda(\delta_{10}) = \lambda(\delta_{11}) = \lambda(\delta_{12}) = q$. Then we conclude that $s' = s_1$ and we check that $u_i = \lambda(s_1) = q$.

4. Case $t \in (t_{i+1} - a, t_{i+1})$ for some i and $w_i \nvDash p$, $t_{i+1} - t_i = a$, $\dot{w}_{i+1} \vDash p$ or $w_{i+1} \vDash p$. We consider a segment of run

$$\cdots (s, v) \xrightarrow[\delta]{\dot{w}_i/\dot{u}_i} (s', v') \xrightarrow{w_i/u_i} (s', v' + r_i) \xrightarrow[\delta']{\dot{w}_{i+1}/\dot{u}_{i+1}} (s'', v'') \cdots$$

   We write $w_i = w'_i \dot{w}'_{i+1} w'_{i+1}$ such that $\dot{w}'_i = \dot{w}_1$ and $\dot{w}_{i+2} = \dot{w}_{i+2}$, and we want to show that $u'_{i+1} = q$. Observe that

- $w'_i \vDash \neg p$
- $\dot{w}'_{i+1} \vDash \neg p$ then δ from $s = s_1, s_2, s_3$ such that $\lambda(\delta) = \neg p$. Then $\delta = \delta_{17}$, and the run looks like

$$\cdots (s_3, v) \xrightarrow[\delta_{17}]{\dot{w}'_{i+1}/\dot{u}'_{i+1}} (s_1, 0) \xrightarrow{w'_{i+1}/u'_{i+1}} (s_1, a) \cdots$$

and $u'_{i+1} \vDash \gamma(s_1) = q$. Then $u'_{i+1} = 1$.

Let us consider some $t \geq 0$, and suppose $u(t) = 1$. Then, if $t = t_i$, the transducer must be taking a discrete transition δ such that $\gamma(\delta) = q$. Then $\delta = \delta_i$ for $i = 1, 3, 5, 7, 9, 11, 13, 15$ and the automaton goes to states s_0 or s_2. But then we have $\dot{w}_i \vDash \Diamond_{(0,a)} p$ since both s_0, s_2 correspond to cases where either (i) $w_i \vDash p$, or (ii) $\dot{w}_{i+1} \vDash p$ or $w_{i+1} \vDash p$ and the length of segment w_i is less than a. Observe that these are cases 1 and 2 from Thm. 3.5.8. If $t \in (t_i, t_{i+1})$, the transducer must be taking a time transition in a state s s.t. $\gamma(s) = q$. Then $s = s_0, s_1, s_2$ and then if s_0 we are in case 1 or 4, if $= s_1$ we are in case 3, if $s = s_2$ we are in case 2. Then by Thm. 3.5.8 $\chi_{\Diamond_{(0,a)} p}(w)(t) = 1$.

The unicity of u follows by the fact that $u = \chi_{\Diamond_{(0,a)} p}(w)$. □

For a bounded signal, the formula $\Diamond_{(0,a)} p$ is true at its last open segment if and only if p holds in the segment. If p doesn't hold then $\Diamond_{(0,a)} p$ is false in the segment. Following this observations we define $\mathcal{T}^b_{\Diamond_{(0,a)} p}$ as $\mathcal{T}_{\Diamond_{(0,a)} p}$ but with a set of final states $F = \{s_0, s_3\}$.

Theorem 3.5.10. *The transducer $\mathcal{T}^b_{\Diamond_{(0,a)} p}$ is functional. Moreover, for every bounded signal w, we have*

$$\chi_{\Diamond_{(0,a)} p}(w) = \mathcal{T}^b_{\Diamond_{(0,a)} p}(w).$$

3.5.4 Once

For every w, t, we have that $(w, t) \vDash \blacklozenge_{(0,a)} p$ iff there is some $t' \in (t - a, t)$ such that $(w, t') \vDash p$. Thus, the value of $\chi_{\blacklozenge_{(0,a)} p}(w)$ at each t depends on the value of p throughout the interval $(t - a, t)$. At intervals I where p holds then the formula is satisfied at $I \oplus (0, a)$ independently of the shape of I. When p becomes untrue, there are three possible output behaviours depending on the duration of the segment where p is false.

Theorem 3.5.11. *For every signal w and every $t \geq 0$ we have that $(w, t) \vDash \blacklozenge_{(0,a)} p$ iff $\chi_{\blacklozenge_{(0,a)} p}(w)(t) = 1$ iff there exists some point-segment partition compatible with $w, \chi_{\blacklozenge_{(0,a)} p}(w)$ and some of the following holds*

1. *$t \in (t_i, t_{i+1}]$ for some i and $w_i \vDash p$*
2. *$t \in (t_i, t_{i+1}]$ for some i and*
   - *$w_i \nvDash p$*

- $t_{i+1} - t_i < a$
- $\dot{w}_{i-1} \vDash p$ *or* $w_{i-1} \vDash p$

3. $t \in (t_i, t_{i+1})$*, for some* i*, and*

- $w_i \nvDash p$
- $t_{i+1} - t_i = a$
- $\dot{w}_{i-1} \vDash p$ *or* $w_{i-1} \vDash p$

4. $t \in (t_i, t_i + a)$*,*

- $w_i \nvDash p$
- $t_{i+1} - t_i > a$
- $\dot{w}_{i-1} \vDash p$ *or* $w_{i-1} \vDash p$

Proof. Analogous to Theorem 3.5.8. □

We now define the temporal tester $\mathcal{T}_{\blacklozenge_{(0,a)}p} = (S, \underline{s}, P, Q, C, Inv, \Delta, \lambda, \gamma, \mathcal{F})$ depicted in Figure 3.5.4 that realises the characteristic function of the formula $\Diamond_{(0,a)}p$, consisting of

- $S = \{s_0, s_1, s_2\}$ such that
- $P = \{p\}$ and $Q = \{q\}$
- $C = \{x\}$
- $\Delta = \{\delta_1, \ldots, \delta_{12}\}$
- $\mathcal{F} = \emptyset$
- The states have the following input and output labels, and invariants

$$\begin{array}{lll} \lambda(s_0) = p & \gamma(s_0) = q & Inv(s_0) = \top \\ \lambda(s_1) = \neg p & \gamma(s_1) = q & Inv(s_1) = x < a \\ \lambda(s_2) = \neg p & \gamma(s_2) = \neg q & Inv(s_2) = \top \end{array}$$

- In the following table we can see at the same time all the edges in Δ and their respective input and output labels

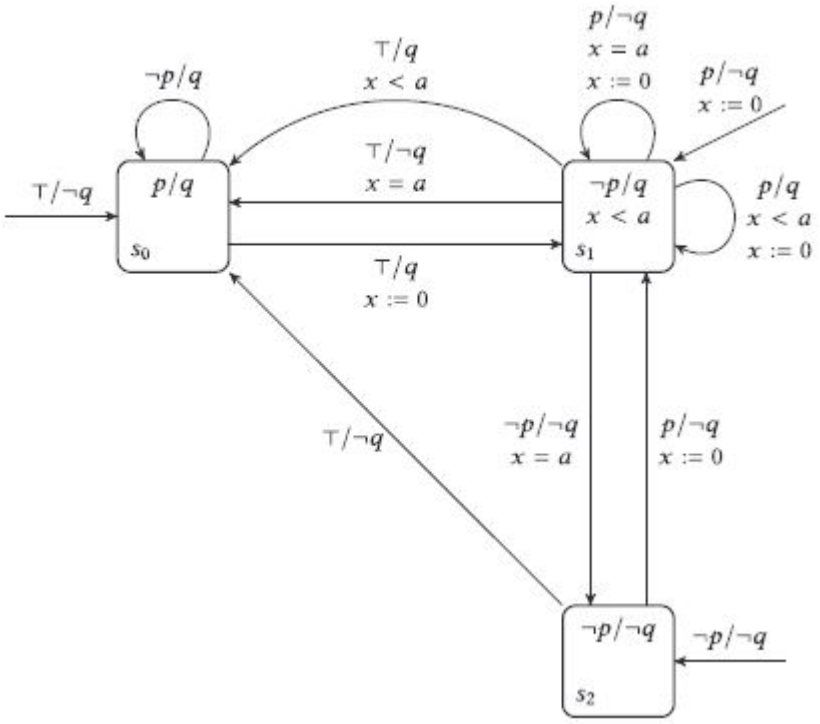


Figure 3.5.4: Tester for $\blacklozenge_{(0,a)}p$

i	δ_i	$\lambda(\delta_i)$	$\gamma(\delta_i)$
1	$(\underline{s}, \top, \emptyset, s_0)$	$\top$	$\neg q$
2	$(\underline{s}, \top, \{x\}, s_1)$	p	$\neg q$
3	$(\underline{s}, \top, \emptyset, s_2)$	$\neg p$	$\neg q$
4	$(s_0, \top, \emptyset, s_0)$	$\neg p$	q
5	$(s_0, \top, \{x\}, s_1)$	$\top$	q
6	$(s_1, x < a, \emptyset, s_0)$	$\top$	q
7	$(s_1, x = a, \emptyset, s_0)$	$\top$	$\neg q$
8	$(s_1, x < a, \{x\}, s_1)$	p	q
9	$(s_1, x = a, \{x\}, s_1)$	p	$\neg q$
10	$(s_1, x = a, \emptyset, s_2)$	$\neg p$	$\neg q$
11	$(s_2, \top, \emptyset, s_0)$	$\top$	$\neg q$
12	$(s_2, \top, \{x\}, s_1)$	p	$\neg q$

This transducer observes the input w and moves through its states. The clock x measures the time from the last time p was observed.

Theorem 3.5.12. *The transducer $\mathcal{T}_{\blacklozenge_{(0,a)}p}$ is functional. Moreover, for every*

signal w,

$$\chi_{\blacklozenge_{(0,a)}p}(w) = \mathcal{T}_{\blacklozenge_{(0,a)}p}(w)$$

Proof. Left to the reader using Lemma 3.5.11. □

Lemma 3.5.11 can easily be adapted to hold true for bounded signals by bounding the index i of the t_i and considering time points of the domain of w. Moreover, we observe that every property about the satisfiability of $\blacklozenge_{(0,a)}p$ at point t is based on past observations at some $t' < t$, and then it holds for signals iff it holds for bounded signals. Hence, the temporal tester $\mathcal{T}_{\blacklozenge_{(0,a)}p}$ also realizes the characteristic function $\chi_{\blacklozenge_{(0,a)}p}$ over bounded signals and we get the corresponding theorem.

Theorem 3.5.13. *The transducer* $\mathcal{T}_{\blacklozenge_{(0,a)}p}$ *is functional. Moreover, for every bounded signal* w *over* p_1, p_2*, we have*

$$\chi_{\blacklozenge_{(0,a)}p}(w) = \mathcal{T}_{\blacklozenge_{(0,a)}p}(w)$$

3.6 An effective translation

In this section we present the main result of this chapter, a plain correspondence between MITL formulas and temporal testers. This correspondence is the key to construct a model-checking decision procedure for MITL-definable properties.

Theorem 3.6.1. *For every MITL formula* φ*, there exists a functional timed transducer* $\mathcal{T}_\varphi$ *such that,* $\mathcal{T}_\varphi(w) = \chi_\varphi(w)$ *for every signal* w*. Moreover, if* S_φ, C_φ*, are the set of states and clocks of* $\mathcal{T}_\varphi$ *respectively, then* $|S_\varphi| \in 2^{O(s(\varphi)r(\varphi))}$ *and* $|C_\varphi| \in O(s(\varphi)r(\varphi))$ *where* s *and* r *denote the size and the resolution.*

Proof. Let φ be an arbitrary formula and let φ' be an equivalent formula in standard form. By Theorem 3.2.2 we have that $s(\varphi') \in O(s(\varphi)r(\varphi))$ and $r(\varphi') \leq 1$.

Let P' be the set of propositional variables appearing in φ'. The definition of a temporal tester for φ' with input variables P' and output variable q, denoted by $\mathcal{T}^q_{\varphi'}$, is recursive on the structure of φ'. For each subformula of φ', we construct a temporal tester for its main subformula(s) and compose it with a basic temporal tester for its main operator. Since the number of subformulas of φ' is $s(\varphi')$ we do this process for $s(\varphi')$ times.

Let p, p_1, p_2 be fresh variables. Then we define the following testers according to the form of the subformula and use Propositions 3.4.1 and 3.4.2 to prove their correctness. The temporal testers are defined as follows

- $\mathcal{T}^q_{\neg\psi} := \mathcal{T}^p_\psi; \mathcal{T}^q_{\neg p}$. Then if w is a signal over P' we have $\mathcal{T}^q_{\neg\psi}(w) = \chi_{\neg\psi}(w)$ by construction. Moreover, $S_{\neg\psi} = S_\psi \times S_{\neg p}$ and since the set $S_{\neg p}$ is unitary, then $|S_{\neg\psi}| = |S_\psi|$. On the other hand, $C_{\neg\psi} = C_\psi \cup C_{\neg p} = C_\psi$.
- $\mathcal{T}^q_{\psi_1 \vee \psi_2} := (\mathcal{T}^{p_1}_{\psi_1} || \mathcal{T}^{p_2}_{\psi_2}); \mathcal{T}^q_{p_1 \vee p_2}$. Then if w is a signal over P' we have $\mathcal{T}^q_{\psi_1 \vee \psi_2}(w) = \chi_{\psi_1 \vee \psi_2}(w)$ by construction. Moreover, $S_{\psi_1 \vee \psi_2} = S_{\psi_1} \times S_{\psi_2} \times$

$S_{p_1 \vee p_2}$, whence $|S_{\psi_1 \vee \psi_2}| = |S_{\psi_1}| \cdot |S_{\psi_2}|$. On the other hand, $C_{\psi_1 \vee \psi_2} = C_{\psi_1} \cup C_{\psi_2} \cup C_{p_1 \vee p_2}$. Therefore $|C_{\psi_1 \vee \psi_2}| = |C_{\psi_1}| + |C_{\psi_2}|$.

- $\mathcal{T}^q_{\psi_1 \mathcal{U} \psi_2} := (\mathcal{T}^{p_1}_{\psi_1} || \mathcal{T}^{p_2}_{\psi_2}); \mathcal{T}^q_{p_1 \mathcal{U} p_2}$. For all w over P', we have that $\mathcal{T}^q_{\psi_1 \mathcal{U} \psi_2}(w) = \chi_{\psi_1 \mathcal{U} \psi_2}(w)$ by construction. Moreover, $S_{\psi_1 \mathcal{U} \psi_2} = S_{\psi_1} \times S_{\psi_1} \times S_{p_1 \mathcal{U} p_2}$. Therefore, $|S_{\psi_1 \mathcal{U} \psi_2}| = |S_{\psi_1}| \cdot |S_{\psi_2}| \cdot 4$. Finally, $C_{\psi_1 \mathcal{U} \psi_2} = C_{\psi_1} \cup C_{\psi_2} \cup C_{p_1 \mathcal{U} p_2} = C_{\psi_1} \cup C_{\psi_2}$. Therefore $|C_{\psi_1 \vee \psi_2}| = |C_{\psi_1}| + |C_{\psi_2}|$.

- $\mathcal{T}^q_{\psi_1 \mathcal{S} \psi_2} := (\mathcal{T}^{p_1}_{\psi_1} || \mathcal{T}^{p_2}_{\psi_2}); \mathcal{T}^q_{p_1 \mathcal{S} p_2}$. For all w over P', we have that $\mathcal{T}^q_{\psi_1 \mathcal{S} \psi_2}(w) = \chi_{\psi_1 \mathcal{S} \psi_2}(w)$ by construction. Moreover, $S_{\psi_1 \mathcal{S} \psi_2} = S_{\psi_1} \times S_{\psi_1} \times S_{p_1 \mathcal{S} p_2}$. Therefore, $|S_{\psi_1 \mathcal{S} \psi_2}| = |S_{\psi_1}| \cdot |S_{\psi_2}| \cdot 4$. Finally, $C_{\psi_1 \mathcal{S} \psi_2} = C_{\psi_1} \cup C_{\psi_2} \cup C_{p_1 \mathcal{S} p_2}$. Therefore $|C_{\psi_1 \mathcal{S} \psi_2}| = |C_{\psi_1}| + |C_{\psi_2}|$.

- $\mathcal{T}^q_{\Diamond_{(0,a)} \psi} := \mathcal{T}^p_{\psi}; \mathcal{T}^q_{\Diamond_{(0,a)} p}$. For all w over P' we have that $\mathcal{T}^q_{\Diamond_{(0,a)} \psi}(w) = \chi_{\Diamond_{(0,a)} \psi}(w)$ by construction. Moreover, $S_{\Diamond_{(0,a)} \psi} = S_{\psi} \times S_{\Diamond_{(0,a)} p}$. Therefore, $|S_{\Diamond_{(0,a)} \psi}| = |S_{\psi}| \cdot 4$. Finally, $C_{\Diamond_{(0,a)} \psi} = C_{\psi} \cup C_{\Diamond_{(0,a)} p}$. Therefore $|C_{\Diamond_{(0,a)} \psi}| = |C_{\psi}| + 1$.

- $\mathcal{T}^q_{\blacklozenge_{(0,a)} \psi} := \mathcal{T}^p_{\psi}; \mathcal{T}^q_{\blacklozenge_{(0,a)} p}$. For all w over P' we have that $\mathcal{T}^q_{\blacklozenge_{(0,a)} \psi}(w) = \chi_{\blacklozenge_{(0,a)} \psi}(w)$ by construction. Moreover, $S_{\blacklozenge_{(0,a)} \psi} = S_{\psi} \times S_{\blacklozenge_{(0,a)} p}$. Therefore, $|S_{\blacklozenge_{(0,a)} \psi}| = |S_{\psi}| \cdot 3$. Finally, $C_{\blacklozenge_{(0,a)} \psi} = C_{\psi} \cup C_{\blacklozenge_{(0,a)} p}$. Therefore $|C_{\blacklozenge_{(0,a)} \psi}| = |C_{\psi}| + 1$.

The constructions above show that $\mathcal{T}_{\varphi'}$, the temporal tester for φ', which is also a temporal tester for φ, exists. Further, we can conclude that $|S_{\varphi'}| \in O(2^{s(\varphi')}) \in O(2^{s(\varphi) r(\varphi)})$ and $|C_{\varphi'}| \in O(s(\varphi')) \in O(s(\varphi) r(\varphi))$. $\square$

Chapter 4

Model-checking MITL-definable time properties

In this chapter, we explain what it means for an automaton to satisfy a formula, and then we provide a model-checker for this question in the particular case where $\mathbb{A}$ is a timed Büchi automaton and φ is a MITL formula. To this end, we first use Theorem 3.6.1 to reduce the model-checking problem to the emptiness problem of timed (signal-based) automata, and then we construct an algorithm that provides an answer.

The emptiness problem of timed automata was studied by Alur and Dill in [1] and the decidability was proven. In the paper, they provide an algorithm that relies on the construction of a *region automaton*, a Büchi automaton that accepts exactly the set of untimed words that are consistent with the set of timed words accepted by a timed automaton. Here, we adapt their work for the case of signal-based automata and calculate the complexity of the corresponding model-checking algorithm. We shall obtain the following estimation.

Theorem 4.0.1. *For every TBA $\mathbb{A}$ and MITL formula φ, the model-checking problem of whether $\mathbb{A} \vDash \varphi$ can be checked in time*

$$O((n_\varphi r_\varphi + C_\mathbb{A})! \cdot 2^{O((E_\varphi + E_\mathbb{A}) \cdot (n_\varphi r_\varphi + C_\mathbb{A}))} \cdot S_\mathbb{A}).$$

4.1 Problem Statement

In general, we say that a finite automaton $\mathbb{A}$ satisfies a formula φ if every word $w \in L(\mathbb{A})$ satisfies φ. For a timed Büchi automaton $\mathbb{A} = (S, \underline{s}, C, \Sigma, \Delta, \lambda, \gamma, Inv, \mathcal{F})$ and a MITL formula φ, we write $\mathbb{A} \vDash \varphi$ iff every signal w in the timed language $L_t(\mathbb{A})$ satisfies φ. Then we consider the following model-checking problems

Input: $\mathbb{A}$ TBA, φ MITL
Problem: does every *signal* w in $L_t(\mathbb{A})$ satisfy φ?

and also

Input: $\mathbb{A}$ TA, φ MITL
Problem: does every *bounded* signal w in $L_t(\mathbb{A})$ satisfy φ?

The answer to these model-cheking problems will be denoted by $MC(\mathbb{A}, \varphi) \in \{0, 1\}$ and $BMC(\mathbb{A}, \varphi) \in \{0, 1\}$ where 1 represents a yes-answer and 0 a no-answer.

Given a MITL formula φ, we recall that a signal w satisfies φ iff $(w, 0) \vDash \varphi$ iff $\mathcal{T}_\varphi(w)(0) = 1$. From the temporal tester $\mathcal{T}_\varphi$ we define a new automaton $\widetilde{\mathcal{T}}_\varphi$ accepting those signals w that don't satisfy φ.

Lemma 4.1.1. *If φ is a MITL formula and $\mathcal{T}_\varphi = (S, \underline{s}, P, Q, C, Inv, \Delta, \lambda, \gamma, \mathcal{F})$ is the corresponding temporal tester, then the automaton $\widetilde{\mathcal{T}}_\varphi = (S, \underline{s}, P, Q, C, Inv, \widetilde{\Delta}, \widetilde{\lambda}, \widetilde{\gamma}, \widetilde{\mathcal{F}})$ where*

- $\widetilde{\Delta} = \Delta \backslash \{\delta \in \Delta : \delta = (\underline{s}, g, R, s)$ *for some* g, R, s *and* $\gamma(\delta) = 1\}$,
- $\widetilde{\lambda}, \widetilde{\gamma}$ *are the restrictions to* $S \cup \widetilde{\Delta}$ *of* λ, γ, *and*
- $\widetilde{F} \in \widetilde{\mathcal{F}}$ *iff there is some* $F \in \mathcal{F}$ *such that* $\widetilde{F} = F \cap \widetilde{\Delta}$

satisfies that $w \in L_t(\widetilde{\mathcal{T}}_\varphi)$ iff $(w, 0) \nvDash \varphi$.

Proof. Observe that $w \in L_t(\widetilde{\mathcal{T}}_\varphi)$ iff $w \in L_t(\mathcal{T}_\varphi)$ and $\mathcal{T}_\varphi(w)(0) = 0$. But by definition of $\mathcal{T}_\varphi$, $w \in L_t(\mathcal{T}_\varphi)$ is trivially satisfied and $\mathcal{T}_\varphi(w)(0) = 0$ implies $(w, 0) \nvDash \varphi$. □

We emphasize that the number of states, clocks and the constants appearing in clock constraints remain equal after the modifications of the lemma above.

For the complexity analysis of a model-checker for $\mathbb{A} \vDash \varphi$, we consider the following parameters

- the *size* of φ, denoted by n_φ,
- the *resolution* of φ, denoted by r_φ,
- the *maximum rational constant* appearing in a subscript of a temporal modality of φ', the standard form of φ, denoted by M_φ,
- the *number of states, clocks, and the maximum rational number* appearing as a constant in a clock constraint of $\mathbb{A}$, denoted by $S_\mathbb{A}$, $C_\mathbb{A}$, $M_\mathbb{A}$ respectively,
- the *least common multiple* (lcm) of integers $b_1, \ldots, b_n$ where $a_1/b_1, \ldots, a_n/b_n$ are rational numbers appearing in some clock constraint of $\mathbb{A}$, denoted $t_\mathbb{A}$,

- the *least common multiple* (lcm) of integers $b_1, \ldots, b_n$ where $a_1/b_1, \ldots, a_n/b_n$ are rational numbers appearing in a subscript of φ, denoted t_φ.

In the next proposition we show that our model-checking problem is reducible to the emptiness problem for TBA.

Proposition 4.1.1. *For every TBA $\mathbb{A}$ and MITL formula φ, there exists a TBA $\mathbb{B}_{\mathbb{A},\varphi}$ such that $\mathbb{A} \vDash \varphi$ iff $L_t(\mathbb{B}_{\mathbb{A},\varphi}) = \emptyset$. Moreover, the number of states of $\mathbb{B}_{\mathbb{A},\varphi}$ is in $O(2^{O(n_\varphi r_\varphi)} \cdot S_\mathbb{A})$, the number of clocks is in $O(n_\varphi r_\varphi + C_\mathbb{A})$ and the maximum constant is $\max\{M_\mathbb{A}, M\}$.*

Proof. Let φ' be the standard form of φ. We compute the temporal tester $\mathcal{T}_{\varphi'}$, let $\widetilde{\mathcal{T}}_{\varphi'}$ be as in Lemma 4.1.1 and use them to define $\mathbb{B}_{\mathbb{A},\varphi} := \mathbb{A}||\widetilde{\mathcal{T}}_{\varphi'}$. Following the definition of the parallel composition, we have $States(\mathbb{B}_{\mathbb{A},\varphi}) = States(\mathbb{A}) \times States(\widetilde{\mathcal{T}}_{\varphi'})$ and $Clocks(\mathbb{B}_{\mathbb{A},\varphi}) = Clocks(\mathbb{A}) \cup Clocks(\widetilde{\mathcal{T}}_{\varphi'})$. The clock constraints in $\mathbb{B}_{\mathbb{A},\varphi}$ arise as conjunctions of clock constraints in $\mathbb{A}$ and $\widetilde{\mathcal{T}}_{\varphi'}$. Then by the bounds from Theorem 3.6.1, the estimation follows. Moreover, $L_t(\mathbb{B}_{\mathbb{A},\varphi}) = L_t(\mathbb{A}) \cap L_t(\widetilde{\mathcal{T}}_{\varphi'})$. □

The rest of this chapter is thus devoted to the development of an emptiness-checker, an algorithm to check the emptiness of a signal language. In the same spirit of Dill's construction [1], the key for our construction is the definition of the *Region automaton* $\mathcal{R}(\mathbb{A})$, a Büchi automaton accepting some untimed language $L \in Untime[L_t(\mathbb{A})]$ as defined in the next section.

4.2 Restriction to integer constants

Definition 4.2.1. For a signal w over Σ, we say that an ω-word $u = u_0u_1u_2\cdots$ over Σ belongs to $Untime[w]$ iff there is some compatible point-segment partition $\mathcal{I} = I_0I_1I_2\cdots$ such that for all $t \in I_i$, $w(t) = u_i$. For a signal language L_t over Σ, $Untime[L_t] = \bigcup_{w\in L_t} Untime[w]$.

For our purposes, we will restrict ourselves to automata whose guards and invariants contain integer constants. In the following lemma, we show that this can be done without loss of generality.

Definition 4.2.2. For a point-segment sequence $\mathcal{I} = \{0\}(0,t_0)\{t_0\}(t_0,t_1)\{t_1\}\ldots$ and $t \in \mathbb{Q}_0^+$, we define $t \cdot \mathcal{I}$ as the point-segment sequence $\{0\}(0, t\cdot t_0)\{t\cdot t_0\}(t\cdot t_0, t\cdot t_1)\{t\cdot t_1\}\ldots$

Lemma 4.2.1. *Consider a TBA $\mathbb{A}$, a signal w decomposed with respect to $\mathcal{I}$ as $\dot{w}_0 w_0^{r_0} \dot{w}_1 a_1^{r_1} \dot{w}_2 \cdots$, and $t \in \mathbb{Q}^+$. Then*

$$r : (\underline{s}, 0) \xrightarrow{\dot{w}_0} (s_0, v_0) \xrightarrow{w_0^{r_0}} (s_0, v_0 + r_0) \xrightarrow{\dot{w}_1} (s_1, v_1) \xrightarrow{w_1^{r_1}} (s_1, v_1 + r_1)\cdots$$

is an accepting run over w for $\mathbb{A}$ iff

$$r_t : (\underline{s}, 0) \xrightarrow{\dot{w}_0} (s_0, t\cdot v_0) \xrightarrow{w_0^{t\cdot r_0}} (s_0, t\cdot(v_0+r_0)) \xrightarrow{\dot{w}_1} (s_1, t\cdot v_1) \xrightarrow{w_1^{t\cdot r_1}} (s_1, t\cdot(v_1+r_1))\cdots$$

is an accepting run over u for $\mathbb{A}_t$ where

- *u is decomposed with respect to $t \cdot \mathcal{I}$ as $\dot{w}_0 a_0^{t \cdot r_0} \dot{w}_1 a_1^{t \cdot r_1} \dot{w}_2 \cdots$, and*
- *$\mathbb{A}_t$ is the same as $\mathbb{A}$ but replacing each value k by $t \cdot k$ in the clock constraints of $\mathbb{A}$.*

Proof. A discrete step of $\mathbb{A}$ of the form $(s_i, v_i + r_i) \xrightarrow{\dot{w}_{i+1}} (s_{i+1}, v_{i+1})$ for some $r_i > 0$ is allowed iff there is some $\delta = (s_i, g, R, s_{i+1})$ such that $v_i + r_i$ satisfies g and $v_{i+1} = R(v_i)$. But δ is a transition of $\mathbb{A}$ iff $\delta_t = (s_i, g_t, R, s_{i+1})$ is a transition of $\mathbb{A}_t$ where g_t is obtained from g by replacing every rational constant k appearing in it by $t \cdot k$. Then $(s_i, t \cdot (v_i + r_i)) \xrightarrow{\dot{w}_{i+1}} (s_{i+1}, t \cdot v_{i+1})$ is a suitable discrete step of $\mathbb{A}_t$. Therefore the initial step $(\underline{s}, 0) \xrightarrow{\dot{w}_0} (s_0, v_0)$ is allowed by $\mathbb{A}$ iff $(\underline{s}, 0) \xrightarrow{\dot{w}_0} (s_0, t \cdot v_0)$ is allowed by $\mathbb{A}_t$. By a similar argument, $(s_i, v_i) \xrightarrow{w_i^{r_i}} (s_{i+1}, v_i + r_i)$ is a time step of $\mathbb{A}$ iff $(s_i, t \cdot v_i) \xrightarrow{w_i^{r_i}} (s_{i+1}, t \cdot (v_i + r_i))$ is a time step of $\mathbb{A}_t$. □

Proposition 4.2.1. *For every TBA $\mathbb{A}$ and MITL formula φ, there exists a TBA $\mathbb{B}^{int}_{\mathbb{A},\varphi}$ with integer constants such that $\mathbb{A} \vDash \varphi$ iff $L_t(\mathbb{B}^{int}_{\mathbb{A},\varphi}) = \emptyset$. Moreover, the number of states of $\mathbb{B}^{int}_{\mathbb{A},\varphi}$ is in $O(2^{O(n_\varphi r_\varphi)} \cdot S_\mathbb{A})$, the number of clocks is in $O(n_\varphi r_\varphi + C_\mathbb{A})$ and the maximum constant in clock constraints is $t_{\mathbb{B}_{\mathbb{A},\varphi}} \cdot \max(M_\mathbb{A}, M)$ where $\mathbb{B}_{\mathbb{A},\varphi}$ is as in Proposition 4.1.1.*

Proof. Let $\mathbb{B}_{\mathbb{A},\varphi}$ be as in Proposition 4.1.1. Let $\{c_1, \ldots, c_n\} \subseteq \mathbb{Q}_0^+$ be all the constants appearing in the clock constraints of $\mathbb{B}_{\mathbb{A},\varphi}$. Then each c_i can be written as $\frac{a_i}{b_i}$ for some $a_i, b_i \in \mathbb{N}$ with $b_i > 0$. Let $k = \mathrm{lcm}(b_1, \ldots, b_n)$. Therefore, the automaton $(\mathbb{B}_{\mathbb{A},\varphi})_k$ as defined in Lemma 4.2.1 has only integer constants in its clock constraints and satisfies $L_t(\mathbb{B}_{\mathbb{A},\varphi}) = \emptyset$ iff $L_t((\mathbb{B}_{\mathbb{A},\varphi})_t) = \emptyset$ and $Untime[L_t(\mathbb{B}_{\mathbb{A},\varphi})] = Untime[L_t((\mathbb{B}_{\mathbb{A},\varphi})_t)]$. Moreover, the maximum constant appearing in a clock constraint is $lcm(b_1, \ldots, b_n) \cdot \max(M_\mathbb{A}, M)$. □

4.3 The Region automaton

In the remainder of the chapter we consider only clock constraints where clocks are compared to *integer values* and we assume that every clock appears in some clock constraint. For a TBA $\mathbb{A}$ and a clock $x \in C$, we denote by $M_x \in \mathbb{Z}_0$ the largest integer value to which x is compared to in any clock constraint of $\mathbb{A}$, and by $M_\mathbb{A} = \max_{x \in C} M_x$. Moreover, we decompose every real number into its integer and fractional part as $r = \lfloor r \rfloor + frac(r)$, e.g. $5.43 = 5 + 0.43$.

We define the *region equivalence* $\sim$ between two time valuations. We say that $v \sim v'$ iff

1. for all $x \in C$, $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$ or both $v(x)$ and $v'(x)$ are greater than M_x,
2. for all $x, y \in C$ with $v(x) \leq M_x$ and $v(y) \leq M_y$, $frac(v(x)) \leq frac(v(y))$ iff $frac(v'(x)) \leq frac(v'(y))$,

3. for all $x \in C$ with $v(x) \leq M_x$, $frac(v(x)) = 0$ iff $frac(v'(x)) = 0$.

Definition 4.3.1. A *clock region*, denoted by $[v]$, is an equivalence class under $\sim$.

The set of clock regions is finite. A bound on the number of clock regions is offered in the following lemma, whose proof can be found in [1].

Lemma 4.3.1. *For a given TBA $\mathbb{A}$ with set of clock variables C, the number of clock regions is bounded by $O[|C|! \cdot 2^{|C|} \cdot \prod_{x \in C}(2M_x + 2)]$, where M_x is the largest integer value to which x is compared to in any clock constraint of $\mathbb{A}$.*

Then if $M_{\mathbb{A}}$ is the maximum of all M_x, we infer that $|C|! \cdot 2^{|C|} \cdot \prod_{x \in C}(2M_{\mathbb{A}}+2)$ is a bound for the number of clock regions. Moreover, we can rewrite it into $|C|! \cdot 2^{|C|} \cdot (2M_{\mathbb{A}} + 2)^{|C|}$ which is $|C|! \cdot 2^{2|C|} \cdot (M_{\mathbb{A}} + 1)^{|C|}$.

For a clock region $[v]$ and a clock constraint $\phi(C)$, we say that $[v]$ *satisfies* $\phi(C)$ iff v satisfies $\phi(C)$ or equivalently iff every $v' \in [v]$ satisfies $\phi(C)$. For a clock region $[v]$ and a reset condition $R \subseteq C$, we define $R([v]) := [R(v)]$. Observe that if $v \sim v'$ and $\phi(C), R$ are arbitrary, then $[v]$ satisfies $\phi(C)$ iff $[v']$ satisfies $\phi(C)$, and $R([v])$ coincides with $R([v'])$.

Definition 4.3.2. We say that $[v_2]$ is a *time successor* of $[v_1]$ iff for each $v \in [v_1]$ there is $t_v > 0$ such that $v + t_v \in [v_2]$.

Proposition 4.3.1. *If $r > 0$ then $[v + r]$ is a time successor of $[v]$.*

Proof. Let $v' \sim v$. We have to show that there is some r' such that $v'+r' \sim v+r$. We can assume without loss of generality that $r < 1$.

If for every $x \in C$, both $v'(x), v(x)$ are greater than M_x, then pick r' arbitrary.

Now suppose there is some $x_0 \in C$ such that $r = 1 - frac(v(x_0))$. Then put $r' = 1 - frac(v'(x_0))$. Then

- for every $x \in C$, if $v(x) > M_x$, then $v'(x) > M_x$, whence $v(x) + r > M_x$ and $v'(x) + r' > M_x$. Otherwise suppose both $v(x), v'(x)$ are below M_x. Then we have $\lfloor v(x) + r \rfloor = \lfloor v(x) \rfloor + \lfloor r \rfloor + \lfloor frac(v(x)) + frac(r) \rfloor = \lfloor v'(x) \rfloor + 0 + \lfloor frac(v(x)) + r \rfloor$. Since $v \sim v'$, then it is equal to $\lfloor v'(x) \rfloor + \lfloor frac(v(x)) + 1 - frac(v(x_0)) \rfloor$. But on the other hand, $\lfloor v'(x) + r' \rfloor = \lfloor v'(x) \rfloor + \lfloor frac(v'(x)) + 1 - frac(v'(x_0)) \rfloor$. Then it only remains to prove that

  $$\lfloor frac(v(x)) + 1 - frac(v(x_0)) \rfloor = \lfloor frac(v'(x)) + 1 - frac(v'(x_0)) \rfloor.$$

  Clearly, the above expressions are either 1 or 0. Observe that $\lfloor frac(v(x)) + 1 - frac(v(x_0)) \rfloor = 1$ iff $frac(v(x)) \leq frac(v(x_0))$ and because $v \sim v'$ this is equivalent to $frac(v'(x)) \leq frac(v'(x_0))$ iff $\lfloor frac(v'(x)) + 1 - frac(v'(x_0)) \rfloor = 1$.

  In the case that $v(x) + r \geq M_x$, this means $v(x) + 1 - frac(v(x_0)) \geq M_x$. Then if $v(x) \geq M_x$, then $v'(x) \geq M_x$ and clearly $v'(x) + r' \geq M_x$. Now

suppose $v(x) < M_x \leq v(x) + 1 - frac(v(x_0))$. Since M_x is an integer and $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$, then $v'(x) + r' \geq M_x$ iff $\lfloor v'(x) + r' \rfloor \geq M_x$. But as both $v(x), v'(x)$ are below M_x, we have seen that $\lfloor v'(x) + r' \rfloor = \lfloor v(x) + r \rfloor$. Hence, $v'(x) + r' \geq M_x$.

- We leave the checking of points 2 and 3 of the definition of region equivalence to the reader.

Otherwise if there is no $x_0 \in C$ as above, order clocks $x_1 \leq \cdots \leq x_n$ according to $frac(v(x))$ i.e. $x_i \leq x_j$ iff $frac(v(x_i)) \leq frac(v(x_j))$ (equivalently for $frac(v'(x))$. Choose an index $i \leq n+1$ such that when adding r, the values $v(x_i), \ldots, v(x_n)$ are moved to the next region equivalence above but not $v(x_1), \ldots, v(x_{i-1})$. Then choose r' that does the same for v'. □

Corollary 4.3.0.1. A *clock region* $[v']$ is a time successor of $[v]$ iff there exists some $r > 0$ such that $[v'] = [v + r]$.

Proof. If $[v'] = [v + r]$ for some $r > 0$, then $[v']$ is a time successor of $[v]$, and conversely if $[v']$ is a time successor of $[v]$, then there is some $r > 0$ such that $v + r \in [v']$, whence $[v + r] = [v']$. □

Definition 4.3.3. For a TBA $\mathbb{A} = (S, \underline{s}, C, \Sigma, \Delta, Inv, \lambda, \gamma, \mathcal{F})$ and for some $M > M_{\mathbb{A}}$, the *region automaton* $\mathcal{R}(\mathbb{A})$ is the generalized BA defined as $(S', S'_0, \Sigma, \Delta', \lambda', \gamma', \mathcal{F}')$

- $S' = \{\langle s, [v] \rangle_t : s \in S\} \cup \{\langle s, [v] \rangle_d : s \in S \cup \{\underline{s}\}\} \cup \{\langle s, [v + M] \rangle_\infty : s \in S\}$
- $S_0 = \{\langle \underline{s}, [0] \rangle_d\}$
- Δ' has elements of the following form
  - $(\langle s, [v] \rangle_t, a, \langle s, [v'] \rangle_d)$ iff $[v']$ is a time successor of $[v]$, for all $t \in (0, t_v)$ (where t_v is s.t. $v + t_v \in [v']$), $[v + t]$ satisfies $Inv(s)$ and $a = \lambda(s)$
  - $(\langle s, [v] \rangle_d, a, \langle s', [R(v)] \rangle_t)$ iff there is some $\delta = (s, g, R, s') \in \Delta$ where $[v]$ satisfies g and $a = \lambda(\delta)$
  - $(\langle s, [v] \rangle_t, a, \langle s, [v'] \rangle_\infty)$ iff $[v'] = [v + M]$ and $a = \lambda(s)$, and $v + t \vDash Inv(s)$ for all $t > 0$
  - $(\langle s, [v] \rangle_\infty, a, \langle s, [v'] \rangle_\infty)$ iff $[v'] = [v + M]$ and $a = \lambda(s)$
- for a state $\langle s, [v] \rangle_{t,d,\infty}$, we define $\lambda'(\langle s, [v] \rangle_{t,d,\infty}) = \lambda(s)$
- for an edge $\delta' = (\langle s, [v] \rangle_x, a, \langle s', [v'] \rangle_y)$ where $x, y \in \{t, d, \infty\}$, we define $\lambda'(\delta') = a$
- $\gamma' = \gamma$
- $\mathcal{F}' = \{\{\langle s, [v] \rangle_{t,d,\infty} : s \in F\} : F \in \mathcal{F}\}$

The region automaton mimicks the runs of $\mathbb{A}$ in a certain way. We extend the states $\langle s, [v] \rangle$ of the region automaton in [1] with subscripts t and d to force an alternation among transitions of type "t" and "d", corresponding to time and discrete steps of $\mathbb{A}$. When the automaton jumps into an ∞-state, then it is forced to loop forever in that state.

Moreover, by Lemma 4.3.1, the region automaton $\mathcal{R}(\mathbb{A})$ has a state for every state of $\mathbb{A}$, clock region $[v]$ and subscript t, d, ∞, and then $|States(\mathcal{R}(\mathbb{A}))| \in O(|C_{\mathbb{A}}|! \cdot 2^{2|C_{\mathbb{A}}|} \cdot (M_{\mathbb{A}} + 1)^{|C_{\mathbb{A}}|} \cdot S_{\mathbb{A}})$.

In the following, we use the notation r_t for runs of timed automata and r for untimed automata.

Definition 4.3.4. We define the *projection* of an infinite run of a TBA $\mathbb{A}$ over a signal w

$$r_t : (\underline{s}, 0) \xrightarrow{\dot{w}_0} (s_0, v_0) \xrightarrow{w_0^{r_0}} (s_0, v_0 + r_0) \xrightarrow{\dot{w}_1} (s_1, v_1) \cdots$$

as the run

$$r = [r_t] : \langle \underline{s}, [0] \rangle_d \xrightarrow{\dot{w}_0} \langle s_0, [v_0] \rangle_t \xrightarrow{w_0} \langle s_0, [v_0 + r_0] \rangle_d \xrightarrow{\dot{w}_1} \langle s_1, [v_1] \rangle_t \cdots$$

We define the *projection* of a finite run of a TBA $\mathbb{A}$ over a signal w

$$r_t : (\underline{s}, 0) \xrightarrow{\dot{w}_0} (s_0, v_0) \xrightarrow{w_0^{r_0}} (s_0, v_0 + r_0) \to \cdots \to (s_i, v_i)^{w_i^{r_i}}$$

as the run

$$r = [r_t] : \langle \underline{s}, [0] \rangle_d \xrightarrow{\dot{w}_0} \cdots \langle s_i, [v_i] \rangle_t \to \langle s_i, [v_i + M] \rangle_\infty \xrightarrow{w_i} \langle s_i, [v_i + M + M] \rangle_\infty \xrightarrow{w_i} \cdots$$

From the definition of the edge relation of $\mathcal{R}(\mathbb{A})$, it follows that $[r_t]$ is a run of $\mathcal{R}(\mathbb{A})$ over w. However, not every run of the region automaton is a run of $\mathbb{A}$.

Definition 4.3.5. A run of the region automaton $\mathcal{R}(\mathbb{A})$ of the form

$$r : \langle \underline{s}, [0] \rangle_d \xrightarrow{w_0} \langle s_0, [v_0] \rangle_t \xrightarrow{w_1} \langle s_1, [v_1] \rangle_d \xrightarrow{w_2} \langle s_2, [v_2] \rangle_t \xrightarrow{w_3} \langle s_3, [v_3] \rangle_d \cdots$$

is *progressive* iff for every clock $x \in C$ there are infinitely many $i \geq 0$ s.t. $[v_i]$ satisfies $(x = 0 \vee x \geq M_x)$.

Proposition 4.3.2. *If r_t is a run of $\mathbb{A}$ over w, then the projection $r = [r_t]$ is a progressive run of $\mathcal{R}(\mathbb{A})$ over some $u \in Untime[w]$.*

Proof. By definition of a run r_t of $\mathbb{A}$, the sum $\sum_{i \in \mathbb{N}} r_i$ of the durations of all time steps along the run diverges. Consider some clock $x \in C$ and pick some $i \geq 0$. Suppose for a contradiction that there is no $i' \geq i$ such that $v_{i'}$ satisfies $v_{i'}(x) = 0$ or $v_{i'}(x) \geq M_x$. Then from step i onwards x is never reset, and clearly after a certain number of time steps the automaton will reach some configuration satisfying $x \geq M_x$. But this is a contradiction. Hence, there is some $i' \geq i$ s.t. $v_{i'}$ satisfying $v_{i'}(x) = 0$ or $v_{i'}(x) \geq M_x$, and the same holds for $[v_{i'}]$ at $[r_t]$. □

In the next lemma, we show that exactly the progressive runs of $\mathcal{R}(\mathbb{A})$ correspond to projections of runs of $\mathbb{A}$.

Lemma 4.3.2. *If r is a progressive run of $\mathcal{R}(\mathbb{A})$ over u, then there exists a run r_t of $\mathbb{A}$ over a signal w such that $r = [r_t]$ and $u \in Untime[w]$.*

Proof. Let $r : \langle \underline{s}, [0] \rangle_d \xrightarrow{w_0} \langle s_0, [v_0] \rangle_t \xrightarrow{w_1} \langle s_1, [v_1] \rangle_d \cdots$ be a progressive run of $\mathcal{R}(\mathbb{A})$ over $u = w_0 w_1 w_2 \cdots$. We define the run r_t and the signal w with respect to $\mathcal{I}$ recursively for i where i indexes states of r. For all $i \geq 0$ until the first j such that $\langle s_j, [v_j] \rangle$ has index ∞ (or for every $i \geq 0$ if j doesn't exist), we construct w_i and u_i such that

- $u_i \in [v_i]$
- $(\underline{s}, 0) \xrightarrow{\dot{w}_0} (s_0, u_0) \xrightarrow{w_0^{r_0}} (s_1, u_1) \to \cdots \to (s_i, u_i)$ is a partial run of $\mathbb{A}$ reading w_i of duration $r_0 + r_1 + \cdots + r_i$
- $w_i = \dot{w}_0 \cdot w_0^{r_0} \cdot \dot{w}_1 \cdot w_1^{r_1} \cdots w_i^{r_i}$

We define r_0 as $(\underline{s}, 0)$ and we set $w_0 = \dot{w}_0$. Now suppose we have constructed the partial run r_i of $\mathbb{A}$ until configuration (s_i, u_i) with $u_i \in [v_i]$. Then we distinguish three cases depending on what is the transition taken in r at step i

1. If it is of the form $(\langle s_i, [v_i] \rangle_d, a_{i+1}, \langle s_{i+1}, [v_{i+1}] \rangle_t)$, then there is some $\delta = (s, g, R, s') \in \Delta$ where $[v_i]$ satisfies g and $a_{i+1} = \lambda(\delta)$. Then construct r_{i+1} by adding the following discrete transition to r_i

   $$((s_i, u_i), a_{i+1}, (s_{i+1}, R(u_i)))$$

   and put $w_{i+1} = w_i \cdot a_{i+1}$. Then by definition $R(u_i) \in [v_{i+1}] = [R(v_i)]$ as $u_i \in [v_i]$ by induction.

2. If it is of the form $(\langle s_i, [v_i] \rangle_t, a_{i+1}, \langle s_i, [v_{i+1}] \rangle_d)$ then $[v_{i+1}]$ is a time successor of $[v_i]$, whence $u_i + r_{u_i} \sim v_{i+1}$ for some $r_{u_i} > 0$. Then construct the partial run r_{i+1} by adding the following time transition to r_i

   $$((s_i, u_i), a_{i+1}^{r_{u_i}}, (s_i, u_i + r_{u_i}))$$

   and put $w_{i+1} = w_i \cdot a_{i+1}^{r_{u_i}}$.

3. If it is of the form $(\langle s_i, [v_i] \rangle_t, a_{i+1}, \langle s_i, [v_i + M] \rangle_\infty)$, then the partial run r_{i+1} coincides with r_i, and the automaton stays at the same configuration by reading the infinite open segment a_{i+1}^∞

   $$(s_i, u_i)^{a_{i+1}^\infty}$$

   and $w_{i+1} = w_i \cdot a_{i+1}^\infty$, for a_{i+1}^∞ the unbounded open segment with constant value a_{i+1}. Observe that $u_i + M \in [v_i + M]$ since $M > M_{\mathbb{A}}$

Following this construction, we obtain a sequence of configurations r_t of $\mathbb{A}$ by joining all the r_i over w such that $[r_t] = r$. However, r_t is a *run* of $\mathbb{A}$ iff $\sum_{n\in\mathbb{N}} r_n$ diverges. Assume, to get a contradiction, that every partial sum converges. In this case, we use the fact that r is a progressive run to define another sequence $\{r'_n\}_{n\in\omega}$ of durations such that $\sum_{n\in\omega} r'_n \to \infty$ and another run r'' satisfying that r'' is a run of $\mathbb{A}$ over w and $[r''] = r$.

From $\{r_n\}_{n\in\omega}$ we define an auxiliary sequence $\{t_n\}_{n\in\omega}$ as $t_0 = 0$ and $t_n = \sum_{i<n} r_n$. Observe that $\{t_n\}_{n\in\omega}$ converges. Let C_0 be the set of clocks x reset infinitely often along r. By definition of convergence and the progressivity of r, the following holds

- there is some $n_0 \in \mathbb{N}$ s.t. $\forall n \geq n_0$, $t_{n+1} - t_n < 1$
- there exists $m_0 \in \mathbb{N}$ s.t. $\forall k > m_0$, $t_k - t_m < 0.5$
- if $x \notin C_0$, there exists $i_0 \in \mathbb{N}$ s.t. $\forall i \geq i_0$, $[v_i]$ satisfies $x > M_x$

Let $j = max\{n_0, m_0, i_0\}$ and let $\{k_n\}_{n\in\mathbb{N}}$ be an strictly increasing sequence of integers with $k_0 = j$ s.t. each clock $x \in C_0$ gets reset at least once between the k_ith and the k_{i+1}th transition steps along r. Then define $\{r'_n\}_{n\in\mathbb{N}}$ as $r'_n = 0.5$ if $n \in \{j, k_1, k_2, \dots\}$ and $r'_n = r_n$ otherwise. We observe that the sequence of the r'_n doesn't converge to zero since infinitely many terms have the value 0.5.

From $\{r'_n\}_{n\in\mathbb{N}}$ and r', the run r'' is defined along the same sequence of states as r_t. Moreover, the discrete transitions remain the same. To obtain a run of $\mathbb{A}$, we modify the duration of the time steps according to $\{r'_n\}_{n\in\mathbb{N}}$. Thus, if r' has a time step of the form $((s_i, u_i), a_{i+1}^{r_i}, (s_i, u_i + r_i))$ at step i, then put $((s_i, u_i), a_{i+1}^{r'_i}, (s_i, u_i + r'_i))$ at the i-th step of r''.

To conclude, we have to prove that r'' is a run of $\mathbb{A}$ s.t. $[r''] = [r_t]$, by showing that for every time step $i > j$ of r'', $u_i + r_i \sim u_i + r'_i$. If $i \notin \{k_1, k_2, k_3, \dots\}$, then $r'_i = r_i$, whence $u_i + r_i \sim u_i + r'_i$. Otherwise, suppose $i = k_n$ for some $n > 0$. We check that

1. for all $x \in C$, $\lfloor (u_i + r_i)(x) \rfloor = \lfloor (u_i + r'_i)(x) \rfloor$ or both $(u_i + r_i)(x)$ and $(u_i + r'_i)(x)$ are greater than M_x,
2. for all $x, y \in C$ with $(u_i + r_i)(x) \leq M_x$ and $(u_i + r_i)(y) \leq M_y$, we have $frac((u_i + r_i)(x)) \leq frac((u_i + r_i)(y))$ iff $frac((u_i + r'_i)(x)) \leq frac((u_i + r'_i)(y))$,
3. for all $x \in C$ with $(u_i + r_i)(x) \leq M_x$, $frac((u_i + r_i)(x)) = 0$ iff $frac((u_i + r'_i)(x)) = 0$.

To this end, we divide C into C_0 and $C \backslash C_0$. To prove (1), let $x \in C_0$. We denote by $k > k_{n-1}$ the index of the last step before k_n where x was reseted. Then $k \geq j$, whence $t_{i-1} - t_k < 0.5$. Hence, the value of x at step $i - 1$ is less than 0.5, and since $r'_i = 0.5$, it is less than 1 after the step i. Now let $x \in C \backslash C_0$. Since $i - 1 \geq j$, then $u_i(x) \geq M_x$, whence $(u_i + r'_i)(x) = u_i(x) + r'_i \geq M_x$. Therefore (1) holds. Moreover, we have proven that the values of the clocks in C_0 are continuously less than 1 after the transition step j.

To prove (2), the only case is where $x, y \in C_0$. Since every valuation v satisfies $v(x), v(y) < 1$ every fractionary part in the formula is the number itself. Thus, it is trivially true that $u_i(x) + r_i \leq u_i(y) + r_i$ iff $u_i(x) + r'_i \leq u_i(y) + r'_i$.

To prove (3), just note that $frac((u_i + r_i)(x)) > 0$ and $frac((u_i + r'_i)(x)) > 0$ for all $x \in C_0$.

Hence, we conclude $[r''] = [r_t] = r$ and r'' is a run of $\mathbb{A}$ over w where $u \in Untime[w]$. Therefore r'' is the run required by the lemma. □

Theorem 4.3.1. *For every TBA $\mathbb{A}$ over Σ there exists a BA $\mathbb{B}$ over Σ such that $L(\mathbb{B}) = Untime[L_t(\mathbb{A})]$. Moreover, the number of states of $\mathbb{B}$ is in $O(|C_{\mathbb{A}}|! \cdot 2^{2|C_{\mathbb{A}}|} \cdot (t_{\mathbb{A}} \cdot M_{\mathbb{A}} + 1)^{|C_{\mathbb{A}}|} \cdot S_{\mathbb{A}})$.*

Proof. For a TBA $\mathbb{A}$, let $\{c_1, \ldots, c_n\} \subseteq \mathbb{Q}_0^+$ be all the constants appearing in the clock constraints of $\mathbb{A}$. Then each $c_i = \dfrac{a_i}{b_i}$ for some $a_i, b_i \in \mathbb{N}$ with $b_i > 0$. Let $t = \text{lcm}(b_1, \ldots, b_n)$. Therefore, the automaton $\mathbb{A}_t = (S, \underline{s}, C, \Sigma, \Delta, Inv, \lambda, \gamma, \mathcal{F})$ defined in Lemma 4.2.1 has only integer constants in its clock constraints and $L(\mathbb{A}) = \emptyset$ iff $L(\mathbb{A}_t) = \emptyset$. Then compute $\mathcal{R}(\mathbb{A}_t) = (S', S'_0, \Sigma, \Delta', \lambda', \gamma', \mathcal{F}')$. If $\mathbb{A}_t$ has an accepting run r_t over w, then $[r_t]$ is a progressive and accepting run of $\mathcal{R}(\mathbb{A}_t)$ over some $u \in Untime[w]$. By Lemma 4.3.2, an accepting progressive run r of $\mathcal{R}(\mathbb{A}_t)$ over u can be associated to an accepting run r_t of $\mathbb{A}_t$ over some w where $u \in Untime[w]$ and $[r_t] = r$. Hence, $u \in Untime[L_t(\mathbb{A}_t)]$ iff there is some progressive run of $\mathcal{R}(\mathbb{A}_t)$ over u.

For every $x \in C$, we define the set $F_x = \{\langle s, [v]\rangle \in S' : [v] \text{ satisfies } (x = 0 \vee x > M_x)\}$ and let $\mathbb{B}$ be the same as $\mathcal{R}(\mathbb{A}_t)$ but with a set $\mathcal{F}$ of accepting sets defined by $F \in \mathcal{F}$ iff $F = F_x$ for some $x \in C$ or $F \in \mathcal{F}'$. Then $Untime[L(\mathbb{A}_t)] = L(\mathbb{B})$. □

Corollary 4.3.1.1. For a TBA $\mathbb{A} = (S, \underline{s}, C, \Sigma, \Delta, \lambda, \gamma, Inv, \mathcal{F})$ and a MITL formula φ, there exists a BA $\mathbb{B}$ such that $\mathbb{A} \vDash \varphi$ iff $L(\mathbb{B}) = \emptyset$. Moreover, the number of states of $\mathbb{B}$ is in $O[(n_\varphi r_\varphi + C_{\mathbb{A}})! \cdot 2^{O(n_\varphi r_\varphi) + C_{\mathbb{A}}} \cdot (t \cdot \max(M_{\mathbb{A}}, M_\varphi))^{n_\varphi r_\varphi + C_{\mathbb{A}}} \cdot S_{\mathbb{A}}]$.

Proof. Let $\mathbb{A}$ be a TBA and φ be a MITL formula. Let φ' be the standard form of φ. We have seen that the temporal tester $\mathcal{T}_{\varphi'}$ has number of states in $O(2^{O(n_\varphi r_\varphi)})$ and number of clocks in $O(n_\varphi r_\varphi)$. Moreover, the constants appearing in clock constraints of $\mathcal{T}_{\varphi'}$ depend on the subscripts of temporal modalities in φ'. For an interval I we denote by a_I, b_I the right and left end-points of I. Then the maximum constant M_φ appearing in a clock constraint of $\mathcal{T}_\varphi$ is the maximum of the set $\{a_I, b_I - a_I : I \text{ is a subscript from some operator in } \varphi\}$. We write all rational constants appearing in $\mathbb{A}, \mathcal{T}_{\varphi'}$ as a/b for $a, b \in \mathbb{Z}$ and $b > 0$ and enumerate these denominators as $b_1, \ldots, b_n$.

For a TBA $\mathbb{B}$, we denote by $\mathbb{B}_t$ the construction from Lemma 4.2.1 where we multiply by an integer $t > 0$ all the constants appearing in clock constraints from $\mathbb{B}$. By Proposition 4.1.1 and Proposition 4.2.1, we know that the TBA $(\mathbb{A}||\widetilde{\mathcal{T}}_{\varphi'})_t$ where $\widetilde{\mathcal{T}}_\varphi$ is as in Lemma 4.1.1 and $t = lcm(b, b_1, \ldots, b_n)$, satisfies $\mathbb{A} \vDash \varphi$ iff $L_t((\mathbb{A}||\widetilde{\mathcal{T}}_\varphi)_t) = \emptyset$. Moreover, the number of states of $(\mathbb{A}||\widetilde{\mathcal{T}}_\varphi)_t$ is in

$O(2^{O(n_\varphi r_\varphi)} \cdot S_\mathbb{A})$, the number of clocks is in $O(n_\varphi r_\varphi + C_\mathbb{A})$ and the maximum constant in clock constraints is $t \cdot \max(M_\mathbb{A}, M)$.

We compute the automaton $\mathcal{R}((\mathbb{A}||\widetilde{\mathcal{T}}_\varphi)_t)$ constructed in the proof of Theorem 4.3.1 s.t. $\mathbb{A} \vDash \varphi$ iff $L(\mathcal{R}((\mathbb{A}||\widetilde{\mathcal{T}}_\varphi)_t)) = \emptyset$. This automaton has number of states in $O(|C_B|! \cdot 2^{|C_B|} \cdot (M_B + 1)^{|C_B|} \cdot S_\mathbb{B})$ where $\mathbb{B} = (\mathbb{A}||\widetilde{\mathcal{T}}_\varphi)_t$. But then we rewrite it as $O[(n_\varphi r_\varphi + C_\mathbb{A})! \cdot 2^{O(n_\varphi r_\varphi)+C_\mathbb{A}} \cdot (t \cdot \max(M_\mathbb{A}, M_\varphi))^{n_\varphi r_\varphi + |C_\mathbb{A}|} \cdot 2^{O(n_\varphi r_\varphi)} \cdot S_\mathbb{A}]$ or equivalently $O[(n_\varphi r_\varphi + C_\mathbb{A})! \cdot 2^{O(n_\varphi r_\varphi)+C_\mathbb{A}} \cdot (t \cdot \max(M_\mathbb{A}, M_\varphi))^{O(n_\varphi r_\varphi + C_\mathbb{A})} \cdot S_\mathbb{A}]$. □

To compute our model-checking time bound we define a new set of parameters:

- let $N_\mathbb{A}$ be the maximum length of the binary code of a constant in $\mathbb{A}$
- let $D_\mathbb{A}$ be the number of constants in $\mathbb{A}$
- let N_φ, D_φ defined similarly for φ

We define by $E_\varphi := N_\varphi \cdot D_\varphi$ the total binary code length of numbers in φ. The number $E_\mathbb{A}$ is defined analogously.

Theorem 4.3.2. *For every TBA $\mathbb{A}$ and MITL formula φ, the model-checking problem of whether $\mathbb{A} \vDash \varphi$ can be checked in time*

$$O((n_\varphi r_\varphi + C_\mathbb{A})! \cdot 2^{O((E_\varphi + E_\mathbb{A}) \cdot (n_\varphi r_\varphi + C_\mathbb{A}))} \cdot S_\mathbb{A}).$$

Proof. Let φ' be the standard form of φ. Then the constants appearing in $\widetilde{\mathcal{T}}_{\varphi'}$ are linear combinations of numbers appearing in subscripts from φ. Then we can assume that the length in binary code of $t_{\varphi'}$ is of the same order as the length of t_φ. Moreover, we can bound $t_{\varphi'}$ by the product of all denominators b_i from rational constants in φ'. For $|\cdot|$ the length in binary code, we obtain $|t_{\varphi'}| \leq |b_1 \cdot b_2 \cdots b_n| \leq |M_{\varphi'}| \cdot D_{\varphi'} = N_{\varphi'} \cdot D_{\varphi'} \leq N_\varphi \cdot D_\varphi \leq E_\varphi$. Similarly, $E_\mathbb{A} = N_\mathbb{A} \cdot D_\mathbb{A}$ is a bound for $|t_\mathbb{A}|$.

Hence, the maximum length of a constant in $(\mathbb{A}||\widetilde{\mathcal{T}}_{\varphi'})_{t_{\mathbb{A}||\widetilde{\mathcal{T}}'_\varphi}}$ is in $O((E_\varphi + E_\mathbb{A}) + (N_\varphi + N_\mathbb{A}))$ which is in $O(E_\varphi + E_\mathbb{A})$. We denote $k := t_{\mathbb{A}||\widetilde{\mathcal{T}}_\varphi}$ Therefore the constants of $(\mathbb{A}||\widetilde{\mathcal{T}}_\varphi)_k$ are at most numbers in $2^{O(E_\varphi + E_\mathbb{A})}$.

Let $\mathbb{B} := \mathcal{R}((\mathbb{A}||\widetilde{\mathcal{T}}_\varphi)_k)$ be the Büchi automaton constructed in the proof of Corollary 4.3.1.1. Then $\mathbb{A} \vDash \varphi$ iff $L(\mathbb{B}) \neq \emptyset$. The language $L(\mathbb{B})$ is nonempty iff there is a state s of $\mathbb{B}$ such that s is accessible from some start state and from itself, by a path containing at least one state from each $F \in \mathcal{F}_\mathbb{B}$. This can be checked in polynomial time in the states of $\mathbb{B}$ [27].

By Corollary 4.3.1.1, the number of states of $\mathcal{R}((\mathbb{A}||\widetilde{\mathcal{T}}_\varphi)_k)$ is in

$$O((n_\varphi r_\varphi + C_\mathbb{A})! \cdot 2^{O(n_\varphi r_\varphi)+C_\mathbb{A}} \cdot 2^{O(E_\varphi + E_\mathbb{A}) \cdot (n_\varphi r_\varphi + C_\mathbb{A})} \cdot 2^{O(n_\varphi r_\varphi)} \cdot S_\mathbb{A})$$

which is in

$$O((n_\varphi r_\varphi + C_\mathbb{A})! \cdot 2^{O((E_\varphi + E_\mathbb{A}) \cdot (n_\varphi r_\varphi + C_\mathbb{A}))} \cdot S_\mathbb{A}).$$

It is clear that the automaton $\mathbb{B}$ is computable in time polynomial in $(n_\varphi r_\varphi + C_\mathbb{A})! \cdot 2^{(E_\varphi + E_\mathbb{A}) \cdot (n_\varphi r_\varphi + C_\mathbb{A}))} \cdot S_\mathbb{A}$.

□

Chapter 5

Emptiness Undecidability of Stopwatch Automata

In many contexts there is an interest in measuring the overall accumulated time that a systems spends in some state. One solution is to augment timed automata with clocks that are allowed to be inactive at certain states, the so-called *stopwatches*. This leads us to the class of Stopwatch Automata (SWA), whose expressive power is very interesting in the context of reasoning about real-time durations.

In this chapter, we present the undecidability of the emptiness problem for Stopwatch Automata by a particular and adapted version of Theorem 4.1 from [18]. Concretely, we obtain undecidability for SWA with a single stopwatch as stated in the following theorem.

Theorem 5.0.1. *If $\mathbb{A}$ is a stopwatch automaton with a single stopwatch, the emptiness problem for $\mathbb{A}$ is undecidable.*

The strategy of the proof is by reduction from the reachability problem for SWA to the halting problem for two-counter machines, which is known to be undecidable. As a consequence, we obtain undecidability of the model-checking problem for SWA. Before giving the proof, we show the utility of Stopwatch Automata in the frame of Regulation 561 [13].

5.1 Why stopwatches?

The first sentence of Article 7 from Reg. 561 regulates the maximum duration of driving periods. According to the law, a truck driver can drive at most 4.5 hours without taking a rest. However, it can do other work between two continuous driving periods until the next rest period, in which case the accumulated durations of these driving periods shall not exceed 4.5 hours.

We will say that a MITL formula *is equivalent to* or *formalizes* a given sentence from the article iff for every signal w and time point t, the formula

φ is true at (w,t) iff (w,t) satisfies the given sentence. If we restrict to the

Article 7

After a driving period of four and a half hours a driver shall take an uninterrupted break of not less than 45 minutes, unless he takes a rest period.

This break may be replaced by a break of at least 15 minutes followed by a break of at least 30 minutes each distributed over the period in such a way as to comply with the provisions of the first paragraph.

case where a driver can only switch between *drive* and *rest* periods (originally, a driver can also do *other work*), then the sentence

> $S1$: *"after a driving period of four hours a driver shall take an uninterrumpted break of not less than 45 minutes"*

from Art. 7 is naturally formalized in MITL as

$$\varphi : drive \rightarrow \Diamond_{[0,4.5h]}\Box_{(0,0.75h)}rest$$

where 4.5h and 0.75h are notations for the number of minutes in these time quantities. Indeed, for any point of time t where *drive* is true, there is some point of time $t' \in [t, t+4.5h]$ such that *rest* continuously holds in $(t', t'+0.75h)$.

In the presence of a third activity, *other work*, the situation is rather different. Think of a driver interleaving periods of driving and doing other work. If the overall accumulated duration of driving periods between some break and the following is less than 4.5h, the driver is in accordance with the law. But formula φ becomes false if the accumulated duration of drive and other work periods is more than 4.5h. For example, consider a driving of 4h duration followed by 1h of other work and then 1h of resting. This sequence of activities is clearly within the law, but all times instants t from the first 0.5h of the driving period doesn't satisfy formula 5.1. That is because they can see no $t' \in [t, t+4.5h]$ where *rest* holds.

We infer that formula φ is not equivalent to sentence $S1$ in the presence of three activities. One can see that finding a formula in MITL that formalizes S_1, is not an straightforward task. Even if possible, the automata accepting the models of the formula is susceptible of having a state-space explosion. However, with a stopwatch automaton with a single stopwatch x_d and a clock x_r, we can count the overall duration of a driving period by letting x_d be active if *drive* is true and inactive otherwise. By a simple check of whether $x_d \leq 4.5h \wedge x_r \geq 0.75h$ holds, we set the condition for a run to be accepting. We will not go into further detail here, but it is clear how useful is the expressive power of SWA in the task of formalizing sentences of this kind; see also Article 6 in Fig. 0.0.2.

5.2 Stopwatch Automata

Stopwatch automata are extensions of finite automata. Its computations happen in time, where the automaton can stay in a state for a while or take an instantaneous transition to another state. The time constraints that regulate transitions of the automaton are described by clock variables that can either be active or inactive at a state. These variables are called *stopwatches*. When they are active, their value increases as time increases, while they remain fixed when inactive. The automaton accepts a given word over Σ if there exists a run from a special state *start* that reads the word, and ends in a special state *accept*.

Definition 5.2.1. A *stopwatch automaton* SWA is a tuple $(S, \Sigma, C, \lambda, d, \Delta, Inv)$ where
- S is a finite set of states including two special states *start* and *accept*;
- C is a set of *stopwatches*;
- $\lambda : S \to \Sigma$;
- $d : S \times C \to \{0, 1\}$ assigns a *slope* to every state and every stopwatch;
- Δ is a transition relation consisting of elements of the form $(s, g, R, s') \in S \times \phi(C) \times \mathcal{P}(C) \times S$;
- $Inv : S \to \phi(C)$ is the invariance map.

We say that $x \in C$ is a *clock* if $d(s, x) = 1$ for every state s. For a stopwatch x we use the notation $\dot{x} = 1$ to indicate that x is active at some state; $\dot{x} = 0$ otherwise. Also, we use the notation $x := 0$ on an edge to indicate a reset of x.

For a state s and assignment v of C, a *step* of the automaton is one of the following:

- A *discrete step* $(s, v) \xrightarrow{0} (s', v')$ where there exists $\delta = (s, g, R, s') \in \Delta$ such that v satisfies g and v' is the result of reseting the clocks in R.

- A *time step* $(s, v) \xrightarrow{t} (s, v')$ for $t > 0$, where for all $x \in C$

$$v'(x) = \begin{cases} v(x) + t & \text{if } d((s, x)) = 1 \\ v(x) & \text{if } d((s, x)) = 0 \end{cases}$$

  as long as v' satisfies $Inv(s)$.

A *run* of the automaton starting from configuration (s_0, v_0) is a finite sequence of discrete and time steps

$$(s_0, v_0) \xrightarrow{t_1} (s_1, v_1) \xrightarrow{t_2} \cdots \xrightarrow{t_m} (s_m, v_m)$$

where $q_i \neq accept$ if $i < m$. The time length of this run is $t_1 + t_2 + \cdots + t_m$. A run is *accepting* if $q_0 = start$, v_0 is constantly 0, and $q_m = accept$. The run *reads* the word

$$\lambda(s_0)^{t_0} \cdots \lambda(s_m)^{t_m}.$$

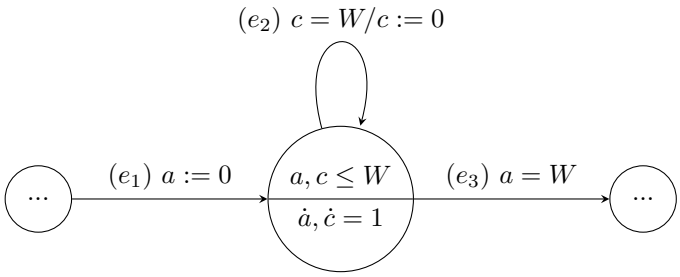


Figure 5.3.1: Wrapping lemma for c

5.3 Undecidability proof

In the following lemmas we introduce three techniques that are necessary for our undecidability proof. The first one shows a mechanism to simulate the inactivity of an active clock during a time step.

Lemma 5.3.1 (Wrapping lemma)**.** *Let W be a positive rational number. Consider the fragment of automaton from Figure 5.3.1. Then if the value of the clock c is x when the edge e_1 is traversed, where $x \leq W$, then c has the value x when traversing e_3.*

Proof. After traversing the edge e_1, the clock a has value 0, and suppose c has value $x \leq W$. Then, because of the invariance condition over c, the automaton necessarily takes a number of time steps of accumulated duration $t = W - x$ enabling edge e_2 to be taken. At this point the value of c is W (observe that a has value $W - x$), which produces a reset of the clock c after traversing e_2. The invariance condition over a leads exclusively to edge e_3 after a period of time of duration x, which leaves a with value W and c with value x. □

For a positive rational number W, a *W-wrapping edge* for a clock c and a state s is an edge from s to itself that is annotated with the guarded command $c = W \to c := 0$. Enriching this technique by forcing two simultaneous W-wrapping edges, we obtain a mechanism to check if the values of two clocks are the same.

Lemma 5.3.2 (Equality lemma)**.** *Let W be a positive rational number. Consider the fragment of a stopwatch automaton from Figure 5.3.2. Suppose that the value of c is x and the value of d is y when the edge e_1 is traversed, where $0 \leq x, y \leq W$. Then the edge e_3 can be traversed later iff $x = y$, and when it is traversed both c and d have the value x (which is equal to y).*

Proof. After traversing the edge e_1, the clock a has value 0, and suppose c has value $x \leq W$ and d has value $y \leq W$. Then, because of the invariance condition over c and d, the automaton necessarily takes a number of time steps of accumulated duration $t = W - \max\{x, y\}$, after wich the clock c has value

$x + t$ and d has value $y + t$. But the edge e_2 is enabled iff both c and d have value W iff $x = y$. At this point the value of both c and d is W (observe that a has value $W - x$), which produces a reset of both clocks after traversing e_2. The invariance condition over a leads exclusively to edge e_3 after a period of time of duration x, which leaves a with value W and both c and d with value x. □

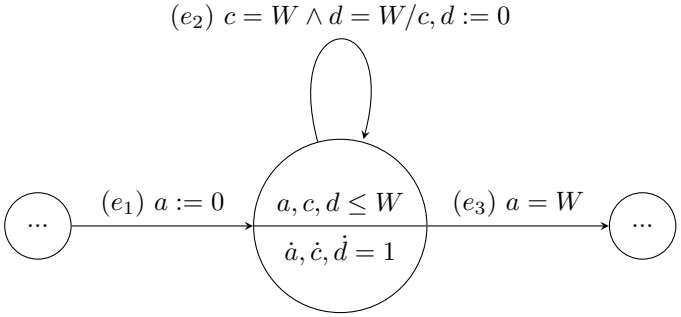


Figure 5.3.2: Equality lemma for $c =?d$

The last lemma allows us to copy the value of a clock to another clock while mantaining the value of the first.

Lemma 5.3.3 (Assignment lemma)**.** *Let W be a positive rational number. Consider the fragment of automaton from Figure 5.3.3. If the value of c is x when the edge e_1 is traversed, where $0 \leq x \leq W$, then the next time e_3 is traversed, the value of c is again x and the value of d is x.*

Proof. After traversing the edge e_1, the clock a has value 0, and suppose c has value $x \leq W$ and d has value $y \leq W$. Then, because of the invariance condition over c and d, the automaton necessarily takes a number of time steps of accumulated duration $t = W - x$ enabling the edge e_2. At this point the value of c is W (observe that a has value $W - x$ and d has value $W - x + y$), which produces a reset of both c and d after traversing e_2. The invariance condition over a leads exclusively to edge e_3 after a period of time of duration x, which leaves a with value W and both c and d with value x. □

The *reachability problem* is the problem of determining whether a certain state of a system or computer program is reachable from a given initial state of the system. We say that a configuration (s, v) of a stopwatch automaton is *reachable* iff there exists a sequence of discrete and time steps starting in configuration $(start, 0)$, where 0 represents the valuation which is constantly 0, and ending in (s, v) where v satisfies $Inv(s)$. Now we can state our main theorem.

Theorem 5.3.1. *If $\mathbb{A}$ is a stopwatch automaton with final state* accept*, then the problem of deciding whether* (accept, v) *is reachable is undecidable.*

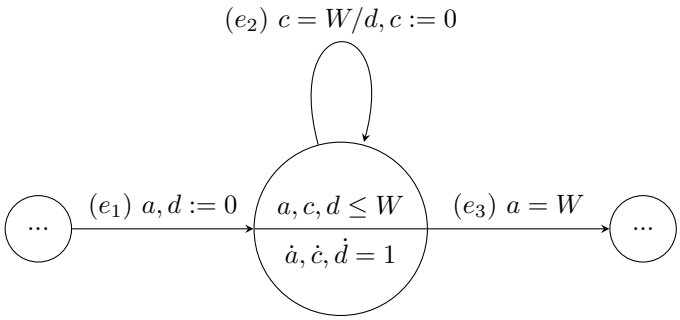


Figure 5.3.3: Assignment lemma for $d := c$

The argument of the proof is by reducibility of the halting problem for two-counter machines to the reachability problem for stopwatch automata.

A *two-counter machine* M is a computational model that consists of

- two variables c, d, called *counters*, that range over $\mathbb{N}$
- a list of n instructions π_M (or *program*) among the following: *increment counter*, *decrement counter* (which leaves unchaged a counter value zero), *if counter is zero then go to instruction* j_0 *else go to instruction* j_1 (where $1 \leq j_0, j_1 \leq n$), and *halt* (which only corresponds to instruction n)
- a variable or *register* $j \in \{1, \ldots, n\}$ that stores the current instruction to be executed

We denote *configurations* of two-counter machines as $((c, d), j)$ and we say that a configuration $((c', d'), j')$ *succeeds* another configuration $((c, d), j)$ (for $j < n$) if (c', d') is the result of applying j to (c, d). In other words, if we are in one of the following cases

- $j =$ "decrement counter c", $c' = c - 1$ if $c > 0$, or $c' = c$ if $c = 0$, $d' = d$ and $j' = j + 1$, or
- $j =$ "increment counter c", $c' = c + 1$, $d' = d$ and $j' = j + 1$, or
- $j =$ "if $c = 0$, go to instruction j_0, else go to instruction j_1", and $c = 0$, $j' = j_0$ or $c > 0$ and $j' = j_1$, and in both cases $d' = d$, or
- $j =$ "decrement counter d", $d' = d - 1$ if $d > 0$, or $d' = d$ if $d = 0$, $c' = c$ and $j' = j + 1$, or
- $j =$ "increment counter d", $d' = d + 1$, $c' = c$ and $j' = j + 1$, or
- $j =$ "if $d = 0$, go to instruction j_0, else go to instruction j_1", and $d = 0$, $j' = j_0$ or $d > 0$ and $j' = j_1$, and in both cases $c' = c$.

A *computation* of a two-counter machine is a finite sequence of configurations

$$((c_0, d_0), j_0)((c_1, d_1), j_1) \cdots ((c_l, d_l), j_l)$$

such that every two consecutive configurations are successors, the initial configurations is $((0,0),1)$, $j_m < n$ for all $m < l$, and $j_l = n = halt$.

The *halting problem* for two-counter machines doesn't have an input and it consists of determining whether the halting condition is reached given a particular program that starts with both counters at 0 and follows the set of instructions specified by the machine. Alan Turing proved in 1936 that this problem is undecidable and therefore that there exists no general algorithm to decide the halting problem [30].

For a given two-counter machine M, we provide a computable translation to a stopwatch automaton $\mathbb{A}_M$ such that

> M halts iff the configuration $(accept, v)$ is reachable for $\mathbb{A}_M$ iff $\mathbb{A}_M$ accepts some word.

Let M be a two-counter machine with counters C, D. The values of the counters and the actions performed by the machine are going to be represented by different elements of the automaton. We encode counter value x by clock value 2^{1-x} and we use auxiliary clocks and a unique stopwatch to define $\mathbb{A}_M$.

The automaton $\mathbb{A}_M$ consists of:

- A unitary alphabet $\Sigma = \{\alpha\}$
- S defined below, with initial state *start* and final state *accept*
- Δ defined below
- A set of stopwatches $C = \{a, b, b', c, d, z\}$ where
  - a, b, b', c, d are *clocks* i.e. active at every state
  - z is a *stopwatch* i.e. $d((s,z))$ may be 0 at some state s (unless we explicitly write $\dot{z} = 0$, assume $\dot{z} = 1$)
  - clocks c and d are associated with values 2^{1-C} and 2^{1-D}, where C, D are the counters of M
- for every state s, $\lambda(s) = \alpha$
- for every state s, $Inv(s)$ is the conjunction of $x \leq 4$ for every clock $x \in \{a, b, b', c, d, z\}$ i.e. every clock must remain in the region $[0, 4]$

$\mathbb{A}_M$ is a concatenation of automata in a certain sense. For every instruction j of the counter machine, it has a certain effect on C or D. Then we construct an automaton $\mathbb{A}_j$ mimicking the effect of j on C, D in terms of clocks c, d. We have to be careful and always leave the value of a clock unchanged after a run of $\mathbb{A}_j$ if the corresponding counter is unchanged during instruction j. Then, with respect to the program π_M, we join all the $\mathbb{A}_j$ in suitable order to construct $\mathbb{A}_M$.

Since by definition both counters start at value 0, the corresponding clocks should hold value $2^{1-0} = 2$ at the starting configuration. Therefore we want to impose an initialization condition to c, d by means of a guard. We define A_0 as in Figure 5.3.4.

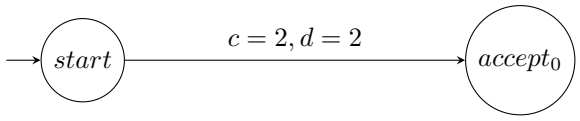


Figure 5.3.4: The automaton $\mathbb{A}_0$

Lemma 5.3.4. *Let $\mathbb{A}$ be a stopwatch automaton and let r, W be two positive rational numbers such that $r \leq W$ and $Inv(s) = \bigwedge_{x \in X} x \leq W$ for every clock x and state s of $\mathbb{A}$. Then assume d is a fresh clock for $\mathbb{A}$. Now let $\mathbb{A}'$ be the automaton $\mathbb{A}$ such that*

- $C' = C \cup \{d\}$
- $Inv'(s) = Inv(s) \wedge d \leq W$ *for every* $s \in S$
- *wrapping loops $d = r/d := 0$ to every state of the automaton i.e. $\delta = (s, d = r, d, s)$ belongs to Δ' for every $s \in S$,*

Then for every run of $\mathbb{A}'$ of duration r, d holds the same value both at the initial and at the final configurations of the run.

Proof. Consider such a run and let us denote by $d_0 \leq W$ the initial value of clock d. Appart from the wrapping loops, clock d doesn't appear at any edge throughout the run. Since the run has duration r, at some point before the run is ended, d will reach value r. Exactly, this will happen after $r - d_0$ units of time, where at some state the self-loop is enabled and clock d is reset. After the transition, the run continues for some time $r - (r - d_0) = d_0$, whence the value of d at exit is d_0. □

Claim 5.3.1.1. For every instruction j of the program π_M for $0 < j < n$, there is an automaton $\mathbb{A}_j$ such that

- Σ, C, Inv, λ are as stated for $\mathbb{A}_M$ above
- it has initial and final states $start_j$ and $final_j$

Moreover, for every computation of M if the configuration of the machine M is $((C, D), j)$ before j is performed and $((C', D'), j')$ after, then any run of $\mathbb{A}_j$ starting at configuration

$$(start_j, (a_0, b_0, b_0', 2^{1-C}, 2^{1-D}, z_0))$$

and spending 0 time in state $start_j$ ends at some configuration of the form

$$(accept_j, (a_1, b_1, b_1', 2^{1-C'}, 2^{1-D'}, z_1)),$$

for some q_k, a_k, b_k, b_k', z_k, for $k = 0, 1$.

We consider the different instructions over counter C, for counter D we do the same construction interchanging c and d.

1. **Case j = "decrement counter C"**. The operation of decrementing 1 unit to counter C is encoded through multiplication by 2, since $2^{1-(C-1)} = 2^{2-C} = 2 \cdot 2^{1-C}$. The automaton $\mathbb{A}_j$ that performs this multiplication is depicted in Figure 5.3.5 and it is divided in three steps. The first step is intended to assign $z := c$, then doubling the value of z, and finally assigning $c := z$. In the special case where clock c has value 2, its value won't be affected by the multiplication, corresponding with the case where we leave unchanged a counter value 0. Let us explain and prove our

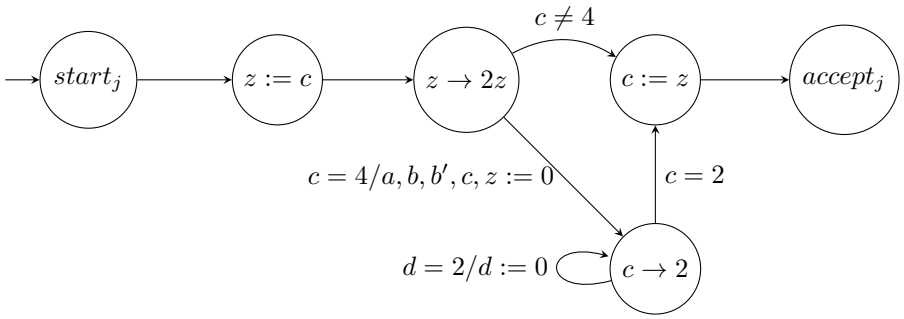


Figure 5.3.5: $c \to 2 \cdot c$

construction with more detail. We omit any reference to the states and describe how are the runs by saying what are the values of the clocks along time.

- We represent by circled $z := c$ (detailed in Fig. 5.3.6) the fragment of automaton showed in the Assignment Lemma 5.3.3 for variables z and c.

  Suppose we are at configuration

  $$(a = a_0, b = b_0, b' = b'_0, c = 2^{1-C}, d = 2^{1-D}, z = z_0).$$

  Then the run over the fragment of automaton from Figure 5.3.6 has duration 4 because there is a clock (a) whose value at entry is 0 and 4 at exit, with no reset in between. Then, when traversing this fragment, the configuration of the automaton is

  $$(a = 0, b = 0, b' = 0, c = 2^{1-C}, d = 2^{1-D}, z = 2^{1-C}),$$

  as d remains unchanged by Lemma 5.3.4 and z is assigned the value of c.
- The fragment for doubling the value of z is depicted in Figure 5.3.7. The first edge of this fragment coincides whith the last from previous

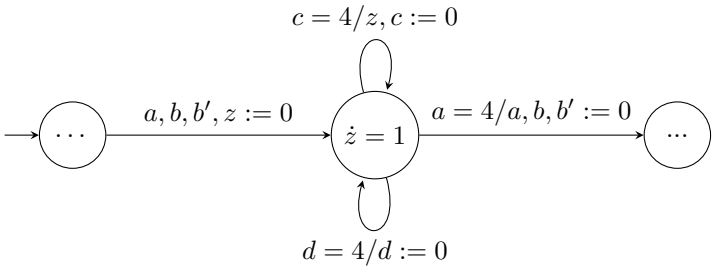


Figure 5.3.6: Assignment lemma for $z := c$ leaving d unchanged

fragment. Then the configuration of $\mathbb{A}_j$ is

$$(a = 0, b = 0, b' = 0, c = 2^{1-C}, d = 2^{1-D}, z = 2^{1-C}),$$

after the respective resets. When entering the state with $\dot{z} = 0$, the automaton spends some time $4 - 2^{1-C}$ at the state until edge e_1 is enabled. At this point $c = 4$, z is unchanged, and $a = b = b' = 4 - 2^{1-C}$. After e_1 clock c is reset and we enter the state where z is now active. Then after a period of time $4 - (4 - 2^{1-C}) = 2^{1-C}$ which enables edge e_2, the clocks satisfy $a = b = b' = 4$, $c = 2^{1-C}$, and $z = 2^{1-C} + 2^{1-C}$. Since this computation lasts 4 units of time, the value of clock d remains unchanged by Lemma 5.3.4. Hence, the automaton is at configuration

$$(a = 4, b = 4, b' = 4, c = 2^{1-C}, d = 2^{1-D}, z = 2 \cdot 2^{1-C})$$

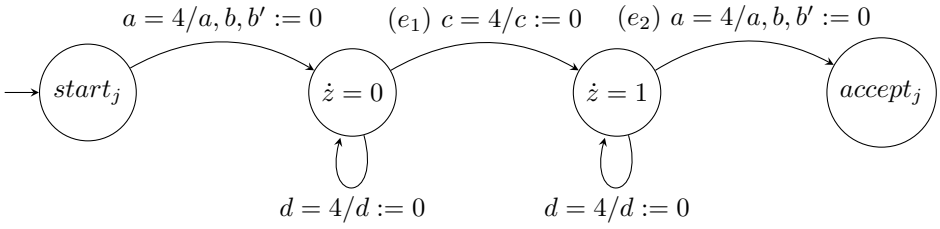


Figure 5.3.7: $z \to 2 \cdot z$

- We represent by circled $c := z$ (detailed in Fig. 5.3.8) the fragment of automaton showed in the Assignment Lemma 5.3.3 for variables c and z.

  We are at configuration

  $$(a = 0, b = 0, b' = 0, c = 2^{1-C}, d = 2^{1-D}, z = 2 \cdot 2^{1-C}).$$

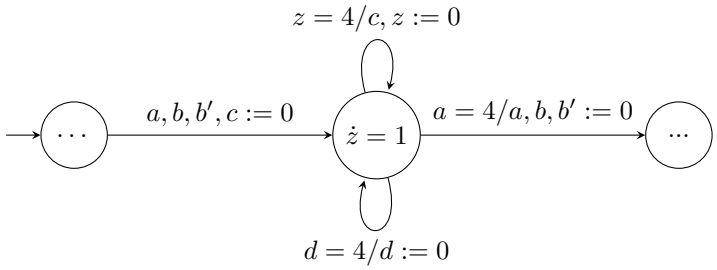


Figure 5.3.8: Assignment lemma for $c := z$ leaving d unchanged

Then the run over the fragment of automaton from Figure 5.3.8 has duration 4 because there is a clock (a) whose value at entry is 0 and 4 at exit, with no reset in between. Then, after traversing this fragment, if $c \neq 4$, the configuration of the automaton is

$$(a = 0, b = 0, b' = 0, c = 2 \cdot 2^{1-C}, d = 2^{1-D}, z = 2 \cdot 2^{1-C}),$$

as d remains unchanged by Lemma 5.3.4 and c is assigned the value of z. Then $c = 2^{1-C'}$ for $C' = C + 1$ and $d = 2^{1-D'}$ for $D' = D$, and then we are done.

- In the special case where clock c has value 2 it should remain unchanged. Since after the run it has value 4, in this particular case we traverse the edge guarded with $c = 4$ to force clock c to recuperate its original value 2. This costs 2 units of time, whence this state has a 2-wrapping edge leaving the value of d unaltered. Then the configuration of the automaton is

  $$(a = 2, b = 2, b' = 2, c = 2, d = 2^{1-D}, z = 2).$$

  Observe that $c = 2^{1-C'}$ for $C' = C = 0$ and $d = 2^{1-D'}$ for $D' = D$, and then we are done.

2. **Case $j =$ "increment counter C"**. The operation of incrementing one unit to counter C is encoded through division by two, since $2^{1-(C+1)} = 2^{-C} = \frac{1}{2} \cdot 2^{1-C}$. The automaton $\mathbb{A}_j$ that performs this multiplication is depicted in Figure 5.3.9 and it operates as follows. First, we give an arbitrary value to b and multiply it by 2 using the doubling procedure above and copy the value on variable b' using Assignment Lemma 5.3.3. Finally, we check if b' is equal to c with the Equality lemma 5.3.2. If so, we perform the assignation $c := b$. Otherwise the run is aborted.

   The only new construction corresponds to the part where we assign a value to b. This segment is detailed in Figure 5.3.10 whose computations last exactly 4 units of time as a is reset at entry and required to be 4 at exit. Because of this we infer by Lemma 5.3.4 that c, d are unchanged along

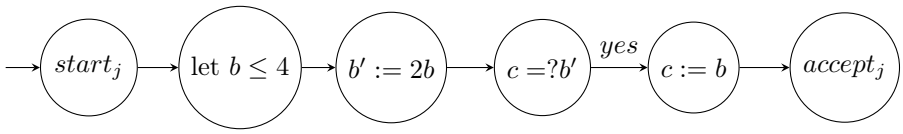


Figure 5.3.9: $c \to \frac{1}{2} \cdot c$

this segment. Moreover, if b has arbitrary value ≤ 4 at entry, then after spending some time $t \leq 4$ at state s_0, it takes a transition to state s_1 after which b is reset and a has value t. Then when leaving state s_1, the clock a has value 4 and b value $4 - t$. Since t is arbitrary, $4 - t$ is arbitrary.

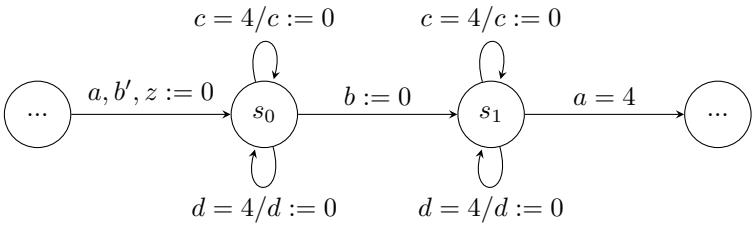


Figure 5.3.10: Assign a value to b leaving c, d unchanged

3. **Case** $j =$ **"if** $c = 0$, **go to instruction** j_0, **else go to instruction** j_1**"**. The automaton $\mathbb{A}_j$ that mimicks this process is depicted in Figure 5.3.11 and it branches according to the values of the clock c.

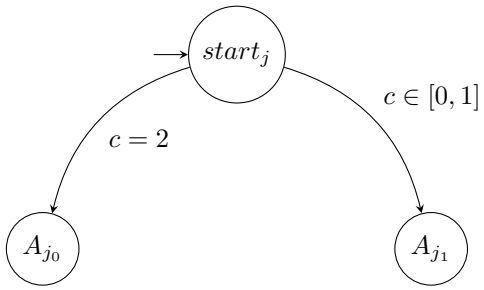


Figure 5.3.11: Test for zero

For instruction $j = n = halt$, we define the automaton A_n (Fig. 5.3.12). Then the automaton $\mathbb{A}_M$ is obtained with respect to π_M by putting $accept_j = start_{j+1}$ for every $0 \leq j \leq n$ and making them be wrapping edges as in Fig. 5.3.13. Finally, we join all the $\mathbb{A}_j$ to obtain $\mathbb{A}_M$.

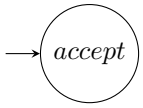


Figure 5.3.12: The automaton $\mathbb{A}_n$

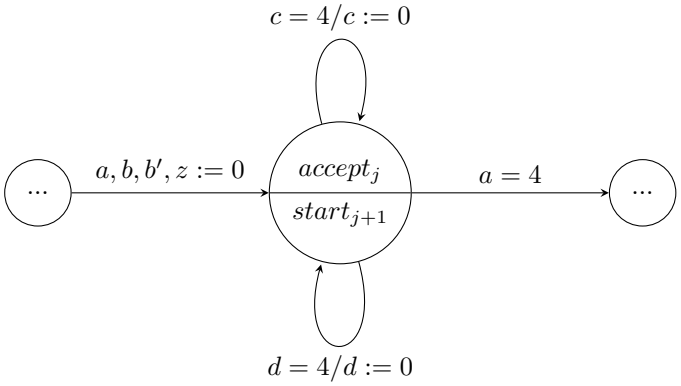


Figure 5.3.13: Gluing with wrapping states

References

[1] Rajeev Alur and David L Dill. "A theory of timed automata". In: *Theoretical computer science* 126.2 (1994), pp. 183–235.

[2] Rajeev Alur, Tomás Feder, and Thomas A Henzinger. "The benefits of relaxing punctuality". In: *Journal of the ACM (JACM)* 43.1 (1996), pp. 116–146.

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

[4] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[5] Béatrice Bérard et al. *Systems and software verification: model-checking techniques and tools*. Springer Science & Business Media, 2013.

[6] J Richard Büchi. "Weak second-order arithmetic and finite automata". In: *Mathematical Logic Quarterly* 6.1-6 (1960).

[7] Edmund M Clarke and E Allen Emerson. "Design and synthesis of synchronization skeletons using branching time temporal logic". In: *Workshop on logic of programs*. Springer. 1981, pp. 52–71.

[8] Edmund M Clarke et al. *Handbook of model checking*. Vol. 10. Springer, 2018.

[9] N. Dershowitz. *Software horror stories*. URL: `http://www.cs.tau.ac.il/%20nachumd/verify/horror.html`.

[10] Cătălin Dima. "Timed shuffle expressions". In: *International Conference on Concurrency Theory*. Springer. 2005, pp. 95–109.

[11] H. Ebbinghaus. "Extended Logics: The General Framework." In: *Model-Theoretic Logics (Perspectives in Logic)*. Cambridge: Cambridge University Press. 2017, pp. 25–76.

[12] Calvin C Elgot. "Decision problems of finite automata design and related arithmetics". In: *Transactions of the American Mathematical Society* 98.1 (1961), pp. 21–51.

[13] *European Parlament and Council of the European Union. Regulation (ec) no 561/2006 of the European Parliament and of the Council of 15 march 2006 on the harmonisation of certain social legislation relating to road transport.* Official Journal of the European Union, 2008.

[14] Thomas Ferrere et al. "From real-time logic to timed automata". In: *Journal of the ACM (JACM)* 66.3 (2019), pp. 1–31.

[15] Paul Gastin and Denis Oddoux. "Fast LTL to Büchi automata translation". In: *International Conference on Computer Aided Verification*. Springer. 2001, pp. 53–65.

[16] Rob Gerth et al. "Simple on-the-fly automatic verification of linear temporal logic". In: *International Conference on Protocol Specification, Testing and Verification*. Springer. 1995, pp. 3–18.

[17] Dimitra Giannakopoulou and Flavio Lerda. "From states to transitions: Improving translation of LTL formulae to Büchi automata". In: *International Conference on Formal Techniques for Networked and Distributed Systems*. Springer. 2002, pp. 308–326.

[18] Thomas A Henzinger et al. "What's decidable about hybrid automata?" In: *Journal of computer and system sciences* 57.1 (1998), pp. 94–124.

[19] Yonit Kesten, Amir Pnueli, and Li-on Raviv. "Algorithmic verification of linear temporal logic specifications". In: *International Colloquium on Automata, Languages, and Programming*. Springer. 1998, pp. 1–16.

[20] Ron Koymans. "Specifying real-time properties with metric temporal logic". In: *Real-time systems* 2.4 (1990), pp. 255–299.

[21] Oded Maler and Amir Pnueli. "On recognizable timed languages". In: *International Conference on Foundations of Software Science and Computation Structures*. Springer. 2004, pp. 348–362.

[22] Zohar Manna and Amir Pnueli. *The temporal logic of reactive and concurrent systems: Specification*. Springer Science & Business Media, 2012.

[23] Robert McNaughton. "Testing and generating infinite sequences by a finite automaton". In: *Information and control* 9.5 (1966), pp. 521–530.

[24] Joël Ouaknine and James Worrell. "On the decidability of metric temporal logic". In: *20th Annual IEEE Symposium on Logic in Computer Science (LICS'05)*. IEEE. 2005, pp. 188–197.

[25] Jean-Pierre Queille and Joseph Sifakis. "Specification and verification of concurrent systems in CESAR". In: *International Symposium on programming*. Springer. 1982, pp. 337–351.

[26] Michael O Rabin. "Decidability of second-order theories and automata on infinite trees." In: *Transactions of the american Mathematical Society* 141 (1969), pp. 1–35.

[27] A Prasad Sistla, Moshe Y Vardi, and Pierre Wolper. "The complementation problem for Büchi automata with applications to temporal logic". In: *Theoretical Computer Science* 49.2-3 (1987), pp. 217–237.

[28] Fabio Somenzi and Roderick Bloem. "Efficient Büchi automata from LTL formulae". In: *International Conference on Computer Aided Verification*. Springer. 2000, pp. 248–263.

[29] Boris A Trakhtenbrot. "Understanding basic automata theory in the continuous time setting". In: *Fundamenta Informaticae* 62.1 (2004), pp. 69–121.

[30] Alan Mathison Turing et al. "On computable numbers, with an application to the Entscheidungsproblem". In: *J. of Math* 58.345-363 (1936), p. 5.

[31] Moshe Y Vardi and Pierre Wolper. "An automata-theoretic approach to automatic program verification". In: *Proceedings of the First Symposium on Logic in Computer Science*. IEEE Computer Society. 1986, pp. 322–331.