Politecnico di Milano
Dipartimento di Elettronica e Informazione
Dottorato di Ricerca in Ingegneria dell'Informazione

Bounded Approaches for Verification of Infinite-State Systems

Doctoral Dissertation of:
Marcello M. Bersani

Advisor:
Prof. Pierluigi San Pietro
Tutor:
Prof. Gianpaolo Cugola
Supervisor of the Doctoral Program:
Prof. Carlo Fiorini

2011 – XXIV

Politecnico di Milano
Dipartimento di Elettronica e Informazione
Piazza Leonardo da Vinci, 32 I-20133 — Milano

Acknowledgements

I would like to thanks my mother, my father, my brother and my sister-in-law; I am grateful for the support they gave me during all these years which allows me to reach this important achievement. Their help was fundamental to face with difficult moments. I give thanks to all my friends who help me and make all my hours more funny. A special thanks to Luca Cavallaro and Achille Frigeri who shared with me joys and sorrows.

I am grateful to all collaborators who worked along with me. I thanks my advisor, Pierluigi San Pietro, who takes care of my work during these years and supported me in difficulties. I thanks Stéphane Demri as well as Arnaud Sangnier for their confidence and precious collaboration we had during my visiting period at LSV in the first semester of 2011.

Abstract. Many verification problems involve checking and synthesis of infinite state systems. In this thesis we study how to solve general verification problems by means of instances of, possibly different, problems of bounded size. Informally, we say that a problem is of bounded size when the object (or solution) which we take into consideration while solving the problem can be defined by a bounded representation with respect to the dimension of the initial problem. In particular, we will face with the satisfiability problem of qualitative specification defined by formulae of linear temporal logic and model-checking problem defined for a specific class of counter systems.

We define the problem of k-bounded satisfiability for formulae of LTL(**FO**), which is a language obtained by adding to linear temporal logic (LTL) a first-order language, and we give an encoding in the decidable theory of equality and uninterpreted functions. Moreover, we consider a fragment of this general logic and we show that the satisfiability problem for LTL (with past operators) over arithmetical constraints can be answered by solving a finite amount of instances of *bounded* satisfiability problems when atomic formulae belong to certain suitable fragments of Presburger arithmetic. A formula is boundedly satisfiable when it admits an ultimately periodic model of the form uv^ω, where u and v are finite sequences of *symbolic valuations*. Therefore, for every formula there exists a *completeness bound* c, such that, if there is no ultimately periodic model with $|uv| \leq c$, then the formula is unsatisfiable. In this case, we say that the language has the completeness property. Whenever a fragment of LTL(**FO**) benefits of such a completeness property then k-bounded satisfiability can be exploited to solve the satisfiability problem for the language.

Most verification problems on counter systems are known to be undecidable in general; decidability can be retained by considering a more specific problem than the general one which is defined with respect to runs with bounded features. We study model-checking problems on counter systems when the specification languages are LTL-like dialects with arithmetical constraints. Guards characterizing transitions of counter systems are Presburger definable formulae and runs are restricted to reversal-bounded ones. We introduce a generalization of reversal-boundedness that captures both original Ibarra's notion as well as more recent ones. We show the NEXPTIME-completeness of the reversal-bounded model-checking problem and the reversal-bounded reachability problem. We show the effective Presburger definability for reachability sets and for sets of configurations for which there is a reversal-bounded run verifying a given temporal formula. Moreover, we show that reversal-bounded model-checking problem can be solved by looking for ultimately

periodic runs with bounded length satisfying a given property. Therefore, since we are restricting the analysis to bounded runs, we can exploit a reduction to k-bounded satisfiability of a temporal formula encoding the transition relation of a counter systems and the semantics of a given temporal specification over ultimately periodic runs.

Abstract. Molti problemi di verifica consistono nell'analisi e sintesi di sistemi a stati infiniti. In questa tesi viene presentato un metodo di risoluzione di una classe di questi problemi che utilizza istanze di problemi di dimensione bounded. Informalmente, diciamo che un problema ha dimensione bounded quando la sua soluzione può essere rappresentata in modo finito rispetto alla dimensione del problema iniziale. In dettaglio, i due principali problemi studiati sono il problema di soddisfacibilità di formule definite mediante formule di logiche temporali ed il problema di model-checking definito rispetto ad una classe specifica di counter systems.

In prima istanza, definiremo il problema di k-bounded satisfiability per formule di LTL(**FO**), che è il linguaggio ottenuto estendendo Linear Temporal Logic (LTL) mediante un linguaggio del primo ordine, e proporremo un encoding nella teoria (decidibile) dell'uguaglianza combinata alla teoria delle funzioni non interpretate. Inoltre, considereremo un frammento di questa logica generale e mostreremo che il problema di soddisfacibilità di LTL (con operatori al passato) con vincoli aritmetici può essere risolto mediante un numero finito di istanze di problemi di k-bounded satisfiability quando le formule atomiche di tale logica appartengono ad uno specifico frammento della logica di Presburger. Diremo che una formula è *bounded* satisfiable quando ammette un modello ultimamente periodico della forma uv^ω, dove u e v sono sequenze finite di *symbolic valuations*. Per ogni formula esiste un *bound di completezza* c tale che, se non esiste alcun modello ultimamemente periodico con $|uv| \leq c$, allora la formula è insoddifacibile. In questo caso, diremo che il linguaggio gode della proprietà di completezza. Quando un frammento di LTL(**FO**) gode della proprietà di completezza allora il problema di k-bounded satisfiability può essere utilizzato per risolvere il problema di soddisfacibilità generale.

Molti problemi definiti su counter systems sono dimostrati essere indecidibili in generale; tuttavia, è possibile preservarne la decidibilità considerando un problema più specifico definito rispetto ad un sottinsieme dei run del sistema soddisfacenti specifiche caratteristiche. Studieremo il problema di model-checking per una classe di counter systems quando il linguaggio di specifica è un frammento di LTL con vincoli aritmetici. Le formule caratterizzanti le transizioni dei counter systems di questa classe sono formule definibili nell'aritmetica di Presburger ed i run sono ristretti a quelli soddisfacenti la proprietà di reversal-boundedness. Introdurremo una generalizzazione della nozione di reversal-boundedness, che include sia l'originale proposta da Ibarra, che altre di più recente definizione. Dimostreremo che il problema di reversal-bounded model-checking e il problema di reversal-bounded reachability risultano NEXPTIME-completi.

Inoltre, dimostreremo che il reachability set e l'insieme di configurazioni raggiungibili da un run reversal-bounded che soddisfa una proprietà definita da una formula temporale nel linguaggio considerato sono definibile nell'aritmetica di Presburger. Il problema di reversal-bounded model-checking può essere risolto ricercando i run ultimamente periodici del counter system in analisi soddisfacenti la proprietà. Pertanto, restringendo l'analisi ai run di lunghezza bounded ed ultimamente periodici, è possibile ridurre il problema di reversal-bounded model-checking al problema di k-bounded satisfiability di una formula temporale che codifica la relazione di transizione del counter system e la semantica della formula definita per run ultimamente periodici.

Contents

List of Figures

List of Tables

1. Introduction

In this thesis we study infinite-state verification of reactive systems. The class of infinite-state systems consists of many subclasses which are usually defined by specific formalisms. A first rough partition can be done by considering the nature of the formalism used to represent a class, although very often, strict equivalences subsist between them. Descriptive formalisms represent systems by using sentences, or formulae, which are defined according to a precise language with a formally defined syntax and semantics. Operational formalisms rely on graph-based structure where edges represent transitions of systems, i.e. they define atomic operations, and sequences of transitions are finite, or infinite, representations of their computations. The (existential) L model-checking problem for an element M of a class of system $\mathcal{C}$ is the problem of checking whether there is a computation of M satisfying a property expressed by a formula in the language L. Various methods are known to solve model-checking problems. They are specific to the class of systems considered and often they depends on the language used to specify properties. On the other hand, given a formula ϕ of a logical language, and a notion of satisfiability with respect to the semantics of the language, the satisfiability is the problem of finding an assignment (model) of values of the domain to each object defining ϕ which satisfies the formula. In this thesis, we consider both the approaches. The undecidability of the halting problem of Turing machines establishes the theoretical barrier to automated verification. Even the simplest class of automata with two nonnegative integer variables and a zero-test instruction (Minsky machine) can simulate Turing machines. Therefore, since the state space is potentially infinite, answering the problem whether such a system will reach any particular state, is not possible to determine. Two main problems which are explored both in finite and infinite state verification are the equivalence-checking and model-checking problems. The former aims at establishing whether two systems are equivalent with respect to a given notion of semantic equivalence. Whereas, the latter amounts to check whether a system satisfies some property which is typically presented in some modal or temporal logic. Different approaches can be adopted to handle infinite state verification. A way to investigate properties of systems is modeling the (symbolic) language that they generate. When a

system has a finite-state space, it can typically be represented by a finite state automaton. In this case, equivalence-checking and model-checking are almost of immediate resolution. However, when the class of language become more expressive insomuch as they are not regular, some decidable problems for finite state systems results to be undecidable. For instance, general equivalence problem for Context-Free languages is proved to be undecidable by Bar-Hillel et al. in [1] but when we reduce the set to deterministic context-free languages, then decidability can be retained [2]. However, extending expressiveness does not lead immediately to undecidable results. Process Algebras and Petri Nets are two different formalisms which are used to model infinite state systems which generates languages that are proved to be contained in the set of context-sensitive languages. However, some of their subclasses benefit of decidable model-checking problem with respect to temporal languages like linear temporal logic LTL (see Section 3). A different point of view in infinite state verification consists in representing configurations of systems as tuples of values from a specific domain. A tuple is a specific assignment of values to variables involved in a systems and formulae belonging to a given formal language L over the set of variables are a symbolic description of the state space. Decidability of Presburger arithmetic plays a fundamental role in this context since it benefits of closures with respect to set-theoretical operations, like intersection and complementation, and decidable satisfiability problem. Whenever we are able to represent (the transition relation of) systems in form of Presburger formulae, we can answer verification problem by reducing them to satisfiability of an arithmetical formula. For instance, this is the case of Petri nets, and the equivalent class of VAS(S), or of some classes of counter systems. A counter system is a finite state automaton equipped with variables ranging over an infinite domains where transitions are labeled with formulae of a language over the set of variables. For these classes, model-checking problem can often be reduced to a reachability problem, which is the problem of checking whether a given system reaches a specific configuration, starting from an initial one. A common way to define formalisms which benefit of decidable properties is to reduce their expressiveness in various way; for instance, one can restrict the structure of the graph defining automata or the semantic behavior of object which transitions manipulate. For instance, reversal-bounded counter systems of Ibarra [3] enforce a semantic restriction over runs by imposing that counters do not change monotonicity more than a given number of times or flat counter system of Finkel and Leroux [4] for which the underlying graph are flat. In these cases, verification questions can be answered by exploiting Presburger definability of transition relation. Beside the

representation of the system (model), an instance of a model-checking problem is given in terms of temporal specification, i.e., a formula of a temporal language. Decidability of model-checking problem is function both of expressiveness of models and of languages used to define specifications. Beside descriptive approaches like Process Algebras, verification problems over infinite state systems can be defined in terms of qualitative specification by means of (temporal) languages enriched with object taking values from infinite set. Qualitative verification is, therefore, performed by checking satisfiability of formulae. Satisfiability problem is, in general, undecidable for temporal languages enriched with variables over infinite domains, since they can encode increments and decrements of Minsky machines, as shown by Comon and Cortier in [5]. The main source of undecidability is hided behind the possibility of defining comparisons among variables at different position of time which allows one to encode runs of Minsky machines. In order to regain decidability, the expressiveness of the language have to be reduced either by restricting the set of relations and functions which define atoms of formulae or by syntactically/semantically reducing the expressiveness of temporal operators. For instance, it can be shown (see Demri and D'Souza [6]) that any relation between two variables which can be represented by a direct acyclic graph, over elements of the domain, leads to undecidability.

In Chapter 3, we consider richer variants of the well known Linear Temporal Logic (LTL) which are fragments of first-order linear temporal logic, denoted LTL(**FO**). The language LTL(**FO**) is a linear temporal logic endowed with first-order objects over infinite (or finite) domains where atoms of formulae belong to a first-order language. Temporal modalities are the same as LTL along with a special modality Xτ which can be used over terms τ to represent the value of τ at the next position of time. It is already known that the satisfiability problem for temporal languages with such a rich alphabet is, in general, undecidable since comparing values of objects at different position along the time allows one to encode computation of (Turing-complete) Minsky machines. However, by reducing the expressiveness of the first-order language of atomic formulae or by imposing suitable syntactic restrictions on the language, decidability of satisfiability problem can be retained. Decidability is obtained, in some case, by reducing the problem to satisfiability of (equisatisfiable) formulae of other formalism, for which there already exists a prove for decidability, or by means of automata-based reduction. In the latter case, the satisfiability problem is solved by means of a reduction to an instance of a different problem, typically a reachability problem, with respect to an automaton representing the formula.

- In Section 3.2 we give a different notion of satisfiability which we call k-bounded satisfiability. Informally, the problem amounts to check whether a formula admits a finite model ρ of the form u or uv which represents an infinite symbolic model $u\Sigma^\omega$ or uv^ω, where Σ is the symbolic alphabet of the symbolic part of the model which consists of atomic formulae disregarding the information from the first-order interpretation. Moreover, we require that ρ admits a finite first-order model of length k, i.e., a finite sequence of k assignment to non-logical objects involved in symbols of Σ. k-bounded satisfiability problem for (existential) LTL(**FO**) is decidable since it can be reduced in polynomial time to satisfiability of (decidable) theory of quantifiers-free uninterpreted functions with equality.

- We present the encoding of LTL(**FO**) over k-bounded models in Chapter 5 which is implemented in the tool `ae2zot`, presented in Chapter 7. The encoding for LTL(**FO**) revises, in some part, the already known linear encoding for LTL and generalize it to the more expressive LTL(**FO**) language. It results intuitive and more concise than the encoding for LTL and it is well-suited to be implemented on SMT-solvers.

Whenever the logical language admits ultimately periodic models in the underlying first-order language, then a bounded symbolic model still represents exactly the infinite first-order model. In other words, a k-bounded symbolic model is representative of an infinite model, even for the first-order part, by iterating v infinitely many times towards the future. However, in full generality, since we adopt a bounded representation of first-order models, k-bounded satisfiability may results not enough to check whether a formula is not satisfiable. In fact, there does not exist, a priori, a bound limiting the length of k of models which one has to check for satisfiability, i.e., a bound on the number of k-bounded satisfiability problem to solve. Nonetheless, we give an example of a fragment of LTL(**FO**) with past-time temporal modalities, denoted CLTLB(L), such that the language of atomic formulae is a fragment of quantifier-free Presburger arithmetic (QFP), for which satisfiability problem can be solved with k-bounded satisfiability. For this language, a proof of decidability, which is based on the automata construction, already exists.

- However, we prove in Section 3.6 that the fragment benefits of the completeness property which states that a formula is satisfiable if, and only if, it is k-bounded satisfiable for some $k \leq K$, where K is a finite nonnegative integer called completeness bound. In gen-

eral, k-bounded satisfiability problem for CLTLB(L) is decidable, provided that the theory of quantifiers-free uninterpreted function with equality combined with L is decidable. Consequently, we obtain that whenever model-checking can be reduced to satisfiability of formulae in CLTLB(L) then k-bounded model-checking problem is complete.

- We study the extension of CLTLB(L) with past-time temporal modalities Y over terms. We prove that formulae of CLTLB(L) using Y can be rewritten to equivalent formulae of CLTLB(L) without Y and that formulae of CLTLB($L \cup \mathbf{AP}$), where $\mathbf{AP}$ denotes the propositional language, are equivalent to formulae in CLTLB(L) (see Section 3.5). Moreover, we show that formulae of CLTLB(L) are initially equivalent to formulae of CLTL(L), which is the fragment of CLTLB(L) without past-time temporal modalities.

k-bounded satisfiability problem is suitable when we are solving reachability problems, since it deals with k-bounded models.

- We exploit k-bounded satisfiability in synthesis of adapters between services in the contest of Service Oriented Applications. CLTLB descriptive specification are exploited to model the behavior of adapters and to constraint the parameters interchange between two services; while services are represented by means of operational models whose transition relation is unwound for a finite number of steps (see Section 5.2).

Verifying a reachability problem on a generic infinite-state system by checking only finite prefixes of length k is, in general, semi-decidable. In fact, the procedure is successful when the configuration is actually reached within k steps.

- However, k-bounded satisfiability becomes effective, i.e. complete, to solve general model-checking problems over infinite runs, for a class $\mathcal{C}$ of infinite-state systems, when two conditions are satisfied: (i) the model-checking problem for the class $\mathcal{C}$ can be reduced to a k-bounded satisfiability problem for decidable fragments of LTL($\mathbf{FO}$) and (i) the length of runs of systems in $\mathcal{C}$, satisfying a given property, have bounded length.

This is the case of the operational formalism which we introduce in Chapter 4, and also other formalisms like, for instance, one-counter automata which we do not consider in this thesis. We focus the analysis on operational formalisms, called counter systems, having finite control state

set and transitions labeled with formulae belonging to the quantifier-free fragment of Presburger arithmetic. In particular, operational models studied in the Chapter 4 form a generalization of Minsky machines: each transition is equipped with a guard from quantifier-free Presburger arithmetic (QFP) and with an update vector in $\mathbb{Z}^n$. Runs are restricted to reversal-bounded runs in which each counter witnesses a bounded number of reversals.

- We introduce a new concept for reversal-boundedness that makes explicit the role of arithmetical terms and it captures previous notions on reversal-boundedness (see Section 2.6).

Model-checking problems are then considered on such counter systems by considering only r-reversal-bounded runs ($r \geq 0$ is part of the input) and a version of linear-time temporal logic with future and past-time operators, and arithmetical constraints at the atomic level. Our main contributions concern new decidability results for richer classes of counter systems and specification languages. The linear-time temporal logic is a fragment of LTL(**FO**) with past-time temporal modalities and the modality X over terms, where control states are atomic formulae as well as arithmetical constraints from fragments of QFP. Moreover, we are able obtain optimal complexity results by translation into Presburger arithmetic. We reduce model-checking problems to reachability problems (first, by synchronization of the counter system and the automaton representing the temporal formula and, then, we reduce the model-checking problems to reachability problems).

- We show that the reversal-bounded model-checking problem for counter systems with guards in QFP (quantifier-free fragment of Presburger arithmetic) and temporal formulae with atomic formulae in QFP is decidable and NEXPTIME-complete (see Theorem 4.4). The same complexity applies to reversal-bounded control state repeated reachability problem and reversal-bounded reachability problem (see Corollary 89).

We prove a fundamental result which justifies the bounded approach of Chapter 3 when we are solving model-checking problems over reversal-bounded counter systems with respect to CLTL specification.

- We show that the existence of reversal-bounded runs satisfying a temporal property implies the existence of reversal-bounded runs of bounded length that are ultimately periodic, i.e. the sequences of transitions are of the form uv^ω where u and v are finite sequences (see Section 4.3.4).

Besides, our complexity results provide as by-products that reachability sets for reversal-bounded counter systems are effectively Presburger definable (see Corollary 85) and sets of configurations for which there is a reversal-bounded run verifying a temporal formula are also effectively Presburger definable (Theorem 90).

Part of the work presented in Chapter 3 is argument of the paper [7] (TIME 2010) and [8] (RP 2011). In particular, Sections 3.5.1, and the encoding presented in Sections 3.3.1 and 3.3.5 (partially) are presented in [7]. Completeness problem for CLTL, which is argument of Section 3.6, is given in [8]. Chapter 4 is based on the work [9] (FroCoS 2011) which can be found in [10] as technical report.

2. Preliminaries

2.1. First Order Language

In this section we remind main concepts related to first order language **FO** which is essential to define Presburger aritmetic in Section 2.2 and extensions of Linear Temporal Logic LTL in Section 3. A first-order language is defined with respect to an alphabet which consists of:

- a set of individual constant symbols $\mathcal{C} = \{c_0, c_1, \ldots\}$,
- a set of function symbols $\mathcal{F} = \{f, g, \ldots\}$,
- a set of predicate symbols $\mathcal{R} = \{P, Q, \ldots\}$,
- boolean connectives $\neg, \wedge$,
- the existential quantifiers symbols $\exists$.

The non-logical symbols represent predicates (relations), functions and constants on the domain of discourse. They are the *signature* of the language which distinguishes first-order languages. Usually, the set $\mathcal{C}$ is understood since it is defined by elements of the domain when a structure $\mathcal{M} = (D, \mathcal{I})$ is defined as we will explain later. Then, the signature of a language is simply $\Sigma = (\mathcal{F}, \mathcal{R})$. Each symbol is associated with a non-negative integer, called "arity", by means of a function $arity : \mathcal{F} \cup \mathcal{R} \to \mathbb{N}$. When $arity(f) = 0$, with $f \in \mathcal{F}$, then f is a variable symbols and, similarly, $arity(R) = 0$, with $R \in \mathcal{R}$, denotes a propositional variable. We will assume that $\mathcal{F} \neq \emptyset$ or $\mathcal{R} \neq \emptyset$. Let V be the set of variables. A *term* τ is defined by the following language:

$$\tau := c \mid f(\tau_1, \ldots, \tau_n)$$

where $c \in \mathcal{C}$ and $f \in \mathcal{F}$ such that $arity(f) = n$. *Atomic formulae*, of the language are relations over terms:

$$\alpha := R(\tau_1, \ldots, \tau_n)$$

where $R \in \mathcal{R}$ is a n-ary predicate symbol. Well-formed *formulae* are defined by the following grammar:

$$\phi := \alpha \mid \neg\phi \mid \phi \wedge \phi \mid \exists y\ \phi$$

where $y \in V$ is a free variable occurring in ϕ. A variable is *free* when it is not in the scope of quantifier $\exists$. The set of all free variables occurring in a formula ϕ is denoted by $free(\phi)$. A formula ϕ is *closed* when there are no occurrences of free variables, i.e., $free(\phi) = \emptyset$.

Semantic of a first-order language is defined with respect to a structure $\mathcal{M} = (D, \mathcal{I})$ where D is a nonempty domain and $\mathcal{I}$ is an interpretation of symbols of the signature such that:

- $\mathcal{I}(c) \in D$ for every $c \in \mathcal{C}$
- $\mathcal{I}(f) : D^{arity(f)} \to D$ for every $f \in \mathcal{F}$
- $\mathcal{I}(R) : D^{arity(f)} \to \{true, false\}$ for every $R \in \mathcal{R}$

To correctly interpret a first-order formula ϕ such that $free(\phi) \neq \emptyset$, the *environment* $\varepsilon : V \to D$ fixes a value of the domain for every element of $free(\phi)$. Let T be the set of terms τ occurring in a formula ϕ. Values of terms with respect to the structure $(\mathcal{M}, \varepsilon)$ are recursively defined by means of the function $[]_\varepsilon : T \to D$ as follows:

$$[\tau]_\varepsilon = \varepsilon(x) \text{ if } \tau \text{ is a variable (0-arity function)}$$
$$[\tau]_\varepsilon = f([\tau_1]_\varepsilon, \ldots, [\tau_n]_\varepsilon) \text{ if } \tau \in \mathcal{F} \text{ such that } arity(f) = n$$

The logic value of a first-order formula is defined in terms of a structure $(\mathcal{M}, \varepsilon)$. The truth relation $\models_\varepsilon$, between a structure $(\mathcal{M}, \varepsilon)$ and a first-order formula ϕ, is inductively defined as:

$$\begin{aligned}
\mathcal{M} \models_\varepsilon R(\tau_1, \ldots, \tau_n) &\overset{\text{def}}{\Leftrightarrow} R([\tau_1]_\varepsilon, \ldots, [\tau_n]_\varepsilon) \\
\mathcal{M} \models_\varepsilon \neg\phi &\overset{\text{def}}{\Leftrightarrow} \mathcal{M} \not\models_\varepsilon \neg\phi \\
\mathcal{M} \models_\varepsilon \phi \wedge \psi &\overset{\text{def}}{\Leftrightarrow} \mathcal{M} \models_\varepsilon \phi \text{ and } \mathcal{M} \models_\varepsilon \psi \\
\mathcal{M} \models_\varepsilon \exists y\ \phi &\overset{\text{def}}{\Leftrightarrow} \mathcal{M} \models_{\varepsilon(y) \leftarrow d} \phi \text{ for some } d \in D
\end{aligned}$$

where $\varepsilon(y) \leftarrow d$ means that $\varepsilon(y)$ is the same as ε except for y which has value d. Given a structure $\mathcal{M}$, we say that a formula ϕ is *satisfiable* with respect to $\mathcal{M}$ when there exists an environment ε such that $\mathcal{M} \models_\varepsilon \phi$. When a structure is not provided, we say that a formula ϕ is satisfiable when there exists a structure $\mathcal{M}$ such that ψ is satisfiable with respect to $\mathcal{M}$. We say that a formula ϕ *holds* with respect to $\mathcal{M}$ when for all environment ε, $\mathcal{M} \models_\varepsilon \phi$. In this case, the logical value of a closed formula do not depend on the environment ε; then, in this case, we simply write $\mathcal{M} \models \phi$ when ϕ holds in $\mathcal{M}$. Given a structure $\mathcal{M}$, we define (first-order) theory $\mathcal{T}_\mathcal{M}$ the set of all closed formulae ϕ such that $\mathcal{M} \models \phi$. Satisfiability problem for a theory $\mathcal{T}_\mathcal{M}$ consists in defining an environment ε such that, given a formula ϕ, $\mathcal{M} \models_\varepsilon \phi$.

2.2. Presburger Arithmetic

In this thesis we use the following notation. The set defining natural (respectively integer) numbers is denoted as usual by $\mathbb{N}$ (respectively $\mathbb{Z}$). Given a vector $\boldsymbol{x} \in \mathbb{Z}^n$, we write $\boldsymbol{x}(i)$, with $1 \leq i \leq n$ to denote the i-th value of $\boldsymbol{x}$.

The following section in mainly inspired to Section 1.3.2 of [11] by Demri and some notation are shared with [9] by Bersani and Demri. Let $\text{VAR} = \{x_0, x_1, \ldots\}$ be a set of variables. Presburger arithmetic [12] (PA) is the first-order theory of $(\mathbb{N}, +, <)$ (respectively $(\mathbb{Z}, +, <)$) where predicate $<$ for the order relation and the symbols $+$ for binary function sum, for $\mathbb{N}$ (and $\mathbb{Z}$). Let x be a variables in VAR and $\mathtt{0}$, $\mathtt{1}$ two symbols. *Terms* t of the language are defined from the grammar below:

$$t := \mathtt{0} \mid \mathtt{1} \mid x \mid t + t$$

where $x \in \text{VAR}$, $a \in \mathbb{Z}$. A *valuation* $\mathbf{val}$ is a map $\mathbf{val} : \text{VAR} \to \mathbb{N}$; it can be extended naturally to the set of all terms as follows: $\mathbf{val}(\mathtt{0}) = 0$, $\mathbf{val}(\mathtt{1}) = 1$, $\mathbf{val}(t + t') = \mathbf{val}(t) + \mathbf{val}(t')$.

Formulae ξ of PA are defined from the grammar below:

$$\xi := \top \mid t < t \mid \ t \equiv_c t \mid \neg\xi \mid \ \xi \wedge \xi \mid \exists x\ \xi \mid \forall x\ \xi$$

where $\top$ is the truth constant, $c \in \mathbb{N} \setminus \{0, 1\}$. A variable x is *free* when it does not occur in the scope of quantifiers. The satisfaction relation $\models_{\text{PA}}$ for PA formulae is defined according to the valuation $\mathbf{val}$ and it is briefly recalled below:

- $\mathbf{val} \models_{\text{PA}} t \equiv_c t' \overset{\text{def}}{\Leftrightarrow}$ there is $n \in \mathbb{Z}$ such that $nc + \mathbf{val}(t) = \mathbf{val}(t')$,
- $\mathbf{val} \models_{\text{PA}} t \leq t \overset{\text{def}}{\Leftrightarrow} \mathbf{val}(t) \leq \mathbf{val}(t')$,
- $\mathbf{val} \models_{\text{PA}} \neg\phi \overset{\text{def}}{\Leftrightarrow} \mathbf{val} \not\models \phi$,
- $\mathbf{val} \models_{\text{PA}} \xi \wedge \xi' \overset{\text{def}}{\Leftrightarrow} \mathbf{val} \models_{\text{PA}} \xi$ and $\mathbf{val} \models_{\text{PA}} \xi'$,
- $\mathbf{val} \models_{\text{PA}} \exists x\ \xi \overset{\text{def}}{\Leftrightarrow}$ there is $n \in \mathbb{N}$ such that $\mathbf{val}_{x \to n} \models_{\text{PA}} \xi$ where $\mathbf{val}_{x \to n}$ is equal to $\mathbf{val}$ except that x is mapped to n.
- $\mathbf{val} \models_{\text{PA}} \forall x\ \xi \overset{\text{def}}{\Leftrightarrow}$ for every$n \in \mathbb{N}$ $\mathbf{val}_{x \to n} \models_{\text{PA}} \xi$

A valuation $\mathbf{val}$ restricted to variables in $V = \{x_1, \ldots, x_n\} \subseteq \text{VAR}$ can be also represented by a vector $\boldsymbol{x} \in \mathbb{N}^n$, where $\mathbf{val}(x_j) = \boldsymbol{x}(j)$ for $j \in [1, n]$. The satisfaction relation can equivalently be written with respect to a vector of values $\boldsymbol{x} \models_{\text{PA}} \phi$.

Symbols $\mathtt{0}$, $\mathtt{1}$ and the relations $<$ and $\equiv_c$ are first order definable. They are introduced as primitive symbols in the grammar but they can

be removed from the language preserving the same expressiveness. In fact, the value 0 can be defined by the formula $\mathtt{0} = \mathtt{0} + \mathtt{0}$ which holds if, and only if, $\mathbf{val}(\mathtt{0}) = 0$ and $\mathtt{1}$ by the formula $\neg\exists y\ 0 < \mathtt{1} \wedge \mathtt{1} < y$ which holds if, and only if, $\mathbf{val}(\mathtt{1}) = 1$. We consider also symbols $<$ and $\equiv_c$ to represent the total order relation among naturals and the modulo relation. The formula $t < t'$ is an abbreviation of $\exists x\ t + x = t' \wedge \neg(x = 0)$, where $\xi \vee \xi'$ is, again, $\neg(\neg\xi \wedge \neg\xi')$.

A formula ξ is *satisfiable* when there is a valuation $\mathbf{val}$ such that $\mathbf{val} \models_{\mathrm{PA}} \xi$. A formula ξ is *valid* when for all valuations $\mathbf{val}$, $\mathbf{val} \models_{\mathrm{PA}} \xi$ holds.

Given a Presburger formula ξ, we write $free(\xi)$ to denote the set of free variables of ξ. Any formula with $u \geq 0$ free variables $x_1, \ldots, x_u$ defines a set of u-tuples:

$$S_{PA}(\xi) \stackrel{\text{def}}{\equiv} \{(\mathbf{val}(x_1), \ldots, \mathbf{val}(x_u)) \in \mathbb{N}^n \mid \mathbf{val} \models_{\mathrm{PA}} \xi\}.$$

Presburger arithmetic has been shown decidable by Presburger in [12]. The satisfiability problem for PA can be solved in triple exponential upper bound [13] in the length of the formula. The lower bound for the problem is studied in [14] by Fischer and Rabin: they provided a decision procedure for satisfiability of PA formulae which is 2EXPTIME-hard and 2EXPSPACE-hard. Moreover, the previous results applies also in the case of nondeterministic algorithms. In [15], Ferrante and Rackoff provided a decision procedure for the first order theory of the structure $(\mathbb{R}, +, <)$ based on quantifiers elimination. Results for $(\mathbb{R}, +, <)$ can be extended to PA, which is proved to be in 2EXPSPACE.

It is worth noticing that in the work of Presburger [12] the first-order theory of $(\mathbb{Z}, +)$ with equality which extends the one over $(\mathbb{N}, +)$ is shown to be decidable. The complexity of the satisfiability problem for the existential fragment of PA is shown NP-complete in [16] by Scarpellini.

2.2.1. Fragments of Presburger Arithmetic

First, we consider is the *quantifier-free* fragment QFP of PA. Essentially, the grammar for QFP is:

$$\xi := \top \mid t < t \mid t \equiv_c t \mid \neg\xi \mid \xi \wedge \xi.$$

The satisfaction relation is the same as $\models_{\mathrm{PA}}$. In [12] the fragment QFP is shown to be equivalent to PA. In fact, there exists a quantifier elimination procedure which transform a PA formula ξ into a QFP formula ξ' such that ξ is satisfiable/valid if, and only if, ξ' is satisfiable/valid.

Though the two languages have the same expressiveness, the satisfiability problem for QFP is known to be NP-complete, see e.g. Papadimitriou [17]. Moreover, the quantifier elimination procedure yields formulae which are exponentially greater than the original quantified formula.

The second fragment we consider limits the number of variables occurring in the order relation. Formulae of *Difference Logic* DL are defined by the following grammar:

$$\xi := \top \mid x < y + d \mid \neg\xi \mid \xi \wedge \xi.$$

with $d \in \mathbb{Z}$. Periodicity constraints of the form $x \equiv_c y + d$ and $x \equiv_c d$ with $d \in \mathbb{N}, d \in \mathbb{N} \in \{0, 1\}$, can be added to DL. This fragment is denoted by DL^+ (the following notation is used also in [11] by Demri).

The Integer Periodic Constraints (IPC^*) or its fragments (e.g., $(\mathbb{Z}, <, =)$ or $(\mathbb{N}, <, =)$) is defined by the following grammar:

$$\xi := \theta \mid x < y \mid \xi \wedge \xi \mid \neg\xi$$
$$\theta := x \equiv_c d \mid x \equiv_c y + d \mid x = y \mid x < d \mid x = d \mid \theta \wedge \theta \mid \neg\theta$$

where $x, y \in V$, $c \in \mathbb{N}^+$ and $d \in \mathbb{Z}$. The first definition of IPC^* can be found in [18] by Demri and Gascon; it is different from ours since it allows the existentially quantified formulae (i.e., $\theta := \exists x\, \theta$) to be part of the language. However, since IPC^* is a fragment of Presburger arithmetic, it has the same expressivity of the above quantifier-free version (but with an exponential blow-up to remove quantifiers). The restriction IPC^{++}, which is introduced in [18], is the language defined by considering θ, rather than ξ, as the axiom in the above grammar.

Presburger formulae can be equivalently written with a more concise representation by allowing constant over $\mathbb{Z}$:

$$\xi := \sum_{1}^{n} a_i x_i \sim d \mid \neg\xi \mid \xi \wedge \xi$$

where $\sim = \{<, >, =, \equiv_c\}$ with $c \in \mathbb{N}$ and $a_i \in \mathbb{Z}$.

2.2.2. Semilinear Sets

The class of semilinear sets was first considered in [19] (later reprinted in [20]) and extensively studied in connection with languages. Given a subset B and P of $\mathbb{N}^n$, the set $L(B, P)$ is defined by all the element $\boldsymbol{x} \in S$ can be represented by the form

$$\boldsymbol{x} = \boldsymbol{b} + \boldsymbol{p}_1 + \cdots + \boldsymbol{p}_m$$

where $\boldsymbol{b} \in B$ and $\boldsymbol{p}_1 \ldots \boldsymbol{p}_m$ is a (possibly) finite sequence of m vectors belonging to P.

Definition 1. *A set $S \subseteq \mathbb{N}^n$ is* linear *if there exist an element $\boldsymbol{b}$ and a finite subset $P \subset \mathbb{N}^n$ such that $S = L(\boldsymbol{b}, P)$.*

Equivalently, a linear set S can be defined as

$$S = \{\boldsymbol{b} + \sum_{i=1}^{m} n_i \boldsymbol{p}_1 \mid n_1, \ldots, n_m \in \mathbb{N}\}.$$

Definition 2. *A set $S \subseteq \mathbb{N}^n$ is* semilinear *if it is a finite union of linear sets.*

Two fundamental results for the theory of semilinear sets were shown by Ginsburg and Spanier in [21].

Theorem 3 (theorem 1.1, [21])**.** *The family of semilinear sets of $\mathbb{N}^n$ is closed with respect to union, intersection and complementation.*

The next result draws the connection between Presburger definable sets of tuples and semilinear sets.

Theorem 4 (theorem 1.3, [21])**.** *The family of Presburger sets on $\mathbb{N}^n$ is identical with the family of semilinear sets of $\mathbb{N}^n$. Moreover, each description is effectively definable from the other.*

In other words, for every semilinear set $S \subseteq \mathbb{N}^n$ there is a Presburger formula ξ such that $S = S_{PA}(\xi)$ and, conversely, for every Presburger formula ξ, with at least one free variable, the set $S_{PA}(\xi)$ is a semilinear subset of $\mathbb{N}^n$.

Proof. The proof is a summary of the proof of theorem 1.3 in [21]. By definition of linear set, it easy to see that every linear set is a Presburger set. Since Presburger arithmetic is closed by union then every semilinear set is a Presburger set.

Conversely, in order to prove that each Presburger formula defines a semilinear set, it is possible to show that given a quantifier-free PA formula ξ over $\mathbb{N}^n$ of the form

$$\xi := \sum_{1}^{n} p_i x_i = b$$

the set of its nonnegative solutions is semilinear. Let

- C be the set of minimal solutions of ξ,
- and P be the set of minimal solution over $\mathbb{N}^n \setminus \{0\}^n$ of $\sum_1^n p_i x_i = 0$.

By Lemma 6.1 of [22] both the set C and P are finite and effectively calculable. Then, the linear set defined by ξ is given by the finite union $\bigcup_{\boldsymbol{c} \in C} L(\boldsymbol{c}, P)$. □

2.3. Theory combination

Most of the work presented in this thesis makes use of Satisfiability Modulo Theories solvers (SMT-solvers) which give a concrete resolution of problems over infinite-state systems defined in Chapter 3. An instance of SMT problem is a generalization of SAT problem over booleans, where atoms are formulae of an underlying (possibly decidable) theory. In general, a first-order formula is defined by logical symbols, like connectives, quantifiers and parenthesis, and non-logical symbols which are functions and predicates symbols. As explained in Section 2.1, the logical value of formulae is interpreted with respect to a model. Satisfiability problem amounts to check whether there exists a model, i.e. an assignment to variables, functions and predicates, such that the interpretation of formula is true. When objects defining formulae are interpreted with respect to a background theory, we are solving a satisfiability problem *modulo* theory. Satisfiability of Presburger arithmetic is an example. Verification problems over non trivial systems aim at checking properties of behaviors usually defined by objects with a different nature, interacting with one another. For instance, we would like to represent the behavior of a program which manipulates arrays where updates of indexes are defined by arithmetic formulae or we are interested in describing a software component by means of the function which it realizes in terms of its input. In other words, we are motivated to combine different specialized theories to obtain a richer language representing different aspects of system. McCarthy in [23] was probably the first who propose a first attempt to model recursive programs by defining an uninterpreted function representing their behavior. We have to mention Burch and Dill [24] who define and implement a decision procedure to verify combinational ALUs by means of uninterpreted function and equality. Core problem of Satisfiability Modulo Theories is combining separate solvers for the theories $\mathcal{T}_1$ and $\mathcal{T}_2$ into one for the union $\mathcal{T}_1 \cup \mathcal{T}_2$. Nelson-Oppen combination method identifies sufficient conditions for combining two theories over disjoint signatures. We say that two theories $\mathcal{T}_1$ and $\mathcal{T}_2$ have disjoint signatures when $\Sigma_1 \cap \Sigma_2 = \{=\}$ where $=$ is the symbol for the binary equality relation. A theory $\mathcal{T}$ is *stably infinite* when every satisfiable quantifier-free formula is satisfiable in an infinite model. For instance, if a theory $\mathcal{T}'$ includes only the formula $\exists x\ x = 0$ then $\mathcal{T}'$ is not stably infinite since the formula $x = 0$ is satisfiable only for $\varepsilon(x) = 0$. Conversely, the theory $\mathcal{T}''$ defined by $\exists x\ x > 0$ is stably infinite. Let H be a set of first-order formulae. Then, we write $H \models \phi$ when for all structure (M, ε) such that $M \models_\varepsilon H$ then $M \models_\varepsilon \phi$. A *literal* is an atomic formula α or its negation $\neg\alpha$. A conjunction of literals H is *convex* when for all

non empty disjunctions of variables $\bigvee_{i \in I} x = y$ then $H \models \bigvee_{x,y \in V} x = y$ if, and only if, $H \models z = w$ for some pair $z, w \in V$. A theory $\mathcal{T}$ is convex when all the conjunctions of literals are convex. Intuitively, a theory is convex when for every satisfiable set of literals there is a model where variables which are not implied to be equal have a distinct assignment $\varepsilon(x)$. Nelson and Oppen provide [25] a method for combining theories which are stably infinite and have disjoint signatures. In particular, let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two consistent, stably infinite theories over disjoint theories. Let $T_1(n)$ and $T_2(n)$ be the time complexity of satisfiability problem of conjunctions of literals. Then, the combined theory $\mathcal{T}_1 \cup \mathcal{T}_2$ is consistent and stably infinite and:

- satisfiability of conjunction of literals of $\mathcal{T}_1 \cup \mathcal{T}_2$ can be decided in $\mathcal{O}(2^{n^2}(T_1(n) + T_2(n)))$;
- when both $T_1(n)$ and $T_2(n)$ are convex then
  - $\mathcal{T}_1 \cup \mathcal{T}_2$ is convex and
  - satisfiability of conjunction of literals of $\mathcal{T}_1 \cup \mathcal{T}_2$ can be decided in $\mathcal{O}(n^4(T_1(n) + T_2(n)))$.

Relevant theories which have concrete decision procedure in most SMT-solvers are the theory of equality and uninterpreted functions (EUF), the theory of quantifier-free linear arithmetic over $\{\mathbb{Z}, \mathbb{Q}\}$ (LIA, LRA) and the difference logic over $\{\mathbb{Z}, \mathbb{Q}\}$ (IDL,RDL), the theory of arrays, bit-vectors and non-linear arithmetic (which is, in general, undecidable). In Sections 3.3.1 and 3.3.5 we will use the theory of equality and uninterpreted functions combined with IDL, RDL or LIA to encode satisfiability problem for an extension of linear temporal logic. Algorithms which solve satisfiability of formulae in EUF are usually based on the congruence closure of graphs. Terms of a formula are encoded by nodes of a directed graph G. Equalities between terms and dependency relation between composed functions (for instance, in $f(a, g(b))$ the function f depends on a and $g(b)$) are encoded with edges between nodes of G. Satisfiability reduces to checking whether the congruence closure is compatible with the structure of the formula. Let G be a graph of n vertices and m edges; the congruence closure for G can be computed in time $\mathcal{O}(m \log m)$ and space $\mathcal{O}(nm)$, in the worst case. Details about congruence closure algorithms and implementations using different data structures can be found in [26] by Downey et al.. Negative cycle detection problem is exploited by Lahiri and Musuvathi [27] to solve satisfiability of conjunctions of IDL or RDL constraints of the form $ax + by \leq d$ where $a, b \in \{-1, 0, +1\}$. Constraints are represented by a direct graph G. Nodes of G represent

variables and DL constraints define edges between nodes. A conjunction of constraints is satisfiable if the graph G does not contain negative cycles. Complexity, in the worst case, is $\mathcal{O}(nm)$ where n is the number of variables involved in conjunction of m constraints. Algorithms for negative cycles detections are deeply studied in literature; for instance, we can cite Cherkassky and Goldberg [28]. Although conjunctions of IDL constraints is solved in polynomial time, the theory of IDL is not convex. Therefore, the combination with EUF still leads to a decidable theory but, by Nelson-Oppen theorem, it is solved in exponential time. NP-completeness for the decision procedure of satisfiability of formulae in the theory of equality with uninterpreted functions combined with the theory of conjunctions of difference logic constraints over $(\mathbb{Z}, <)$ is given by Pratt in [29]. RDL has the property of convexity. Therefore, satisfiability for the combined theory EUF with RDL is solved in polynomial time.

2.4. Büchi automata and ω-regular languages

This subsection recalls main results and definitions concerning automata theory and languages. An *alphabet* is a nonempty finite set of distinct symbols. A *finite word* of length n over an alphabet Σ is a finite sequence of symbols of Σ, i.e., mapping $w : \{0, \ldots, n-1\} \to \Sigma$. Naturally, this definition extends to words of infinite length: an *infinite word* over an alphabet Σ is a mapping $w : \mathbb{N} \to \Sigma$. The length of a word w (or of a sequence s of element) is denoted by $|w|$. A finite word of length n is often represented by its sequence $w = w(0) \ldots w(n-1)$ where $w(i) \in \Sigma$ is the element of w in position i. The set of finite (respectively infinite) words over the alphabet Σ is denoted by Σ^* (respectively Σ^ω).

A *finite-state* automaton over Σ is a tuple $\mathcal{A} = (\Sigma, Q, Q_0, \delta, F)$ where:

- Q is a finite set of control states,
- $Q_0 \subseteq Q$ is set of initial control state,
- $\delta \subseteq Q \times \Sigma \times Q$ is a *finite transition relation* and
- $F \subseteq Q$ is the set (possibly empty) of accepting states.

A *finite run* on a (finite) word w is a finite sequence of control states $q_0 q_1 \ldots$, i.e., a labeling $\rho : \{0, \ldots, n-1\} \to Q$ such that $\rho(0) \in Q_0$ and $\rho(i) w(i) \rho(i+1) \in \delta$, also written $\rho(i) \xrightarrow{w(i)} \rho(i+1) \in \delta$. A finite run of length n is *accepting* when $\rho(n) \in F$. A *infinite run* on a (infinite) word w is a infinite sequence of control states $q_0 q_1 \ldots$, i.e.,

a labeling $\rho : \mathbb{N} \to Q$ such that $\rho(0) \in Q_0$ and $\rho(i)w(i)\rho(i+1) \in \delta$, also written $\rho(i) \xrightarrow{w(i)} \rho(i+1) \in \delta$. While the acceptance condition of automata over finite word is canonical, there are different possibilities of defining acceptance over infinite words. Acceptance conditions are usually defined with respect to a set of control states which are repeated infinitely often along runs. We define $inf(\rho) \subseteq Q$ to be the set $\{q \in Q \mid \rho(i) = q \text{ and } |\{i \in \mathbb{N}\}| \text{ is infinite}\}$. One can define different acceptance conditions. The most frequently used ones are acceptance according to Büchi, Muller, Street and Rabin conditions. According to the respective accepting condition, one can define Büchi, Muller, Street and Rabin automata. In the thesis we will use the Büchi acceptance condition which is defined as $inf(\rho) \cap F \neq \emptyset$. Given an automaton $\mathcal{A}$ we write $\mathscr{L}(\mathcal{A})$ to denote the language accepted by $\mathcal{A}$.

An automaton may behave nondeterministically on an input word, since it may have many initial states and the transition relation may specify many possible transitions for each state and symbol. In particular, of $|Q_0| = 1$ and there is at most one $q' \in Q$ such that $(q, a, q') \in \delta$ then the automaton is *deterministic*. In the sequel, we focus on Büchi automata.

The class of languages accepted by Büchi automata forms the class of (Büchi recognizable) ω-regular languages. It admits various characterization: ω-regular expressions, monadic second order logic over $(\mathbb{N}, <)$ [30] and LTL with fixed-point quantification [31] or automata-based temporal operators [32].

Although every regular language can be accepted by a deterministic finite-state automaton, this is not true for ω-regular language. In fact, given $\Sigma = \{a, b\}$ the alphabet, the language $(a+b)^*a^\omega$ is not recognizable by any deterministic Büchi automaton.

Theorem 5. *There exists an ω-regular language which is not accepted by any deterministic Büchi automaton. Then, the class of languages recognized by deterministic Büchi automata is strictly contained in the class of languages recognized by nondeterministic Büchi automata.*

Nonetheless, ω-regular languages benefit of nice closures properties.

Theorem 6 ([30])**.** *The family of ω-regular languages is closed under intersection, union and complementation.*

Nonemptiness problem for Büchi automata is decidable and the problem is shown to be NLOGSPACE-complete in the work of [33]. A fundamental property related to results in Section 3.6 and 5, is related to the notion of ultimately periodicity of a word. A word $w \in \Sigma^\omega$ is *ultimately-periodic* if it is of the form $w = uv^\omega$ where $u, v \in \Sigma^*$.

Theorem 7 ([30]). *Every nonempty ω-regular language contains an ultimately periodic word.*

Büchi accepting condition can be extended to set of accepting sets instead of one set. A *generalized* Büchi automaton is a tuple $\mathcal{A} = (\Sigma, Q, Q_0, \delta, \{F_1, \ldots, F_k\})$ such that $F_i \subseteq Q$. A run ρ is accepting when $inf(\rho) \cap F_i \neq \emptyset$ for every F_i. Generalized Büchi automata are as expressive as Büchi automata, i.e, the class of languages accepted by generalized Büchi automata is the same as the class of languages accepted by Büchi automata.

Theorem 8 ([30]). *Let $\mathcal{A} = (\Sigma, Q, Q_0, \delta, \{F_1, \ldots, F_k\})$ be a generalized Büchi automaton. There exists a (standard) Büchi automaton $\mathcal{A}' = (\Sigma, Q', Q'_0, \delta', F)$ such that $\mathscr{L}(\mathcal{A}) = \mathscr{L}(\mathcal{A}')$.*

Proof. The automaton $\mathcal{A}'$ consists of k copies of $\mathcal{A}$ which are used to simulate the generalized accepting condition by passing from one copy to another.

- $Q' = Q \times \{1, \ldots, k\}$
- $Q'_0 = Q_0 \times \{1\}$
- $F' = F_1 \times \{1\}$
- $(q, i) \xrightarrow{a} \beta \in \delta$:
  - $\beta \in \{(q', i) : (q, a, q') \in \delta \text{ and } q \notin F_i\}$
  - $\beta \in \{(q', (i+1) \bmod k) : (q, a, q') \in \delta \text{ and } q \in F_i\}$

□

The automaton $\mathcal{A}'$ can be computed in logarithmic space in the dimension of $\mathcal{A}$.

2.5. An historical model: Minksy machines

In this section, we give a shallow intuition of the source of undecidability which is inherently embedded within formalisms which are enough expressive to represent two unbounded positive integer variables, increment/decrement and zero tests. This is strongly related to problems that we consider in Chapter 3 and 4 since the extension of LTL and the class of counter systems we consider in the thesis are enough expressive to encode runs of such machines. Therefore, their fundamental decision problems become immediately undecidable.

The same notion of computability defined by Turing can be realized within a restricted class of Turing machines constituted by machines having only two tapes which can neither write on nor erase the tapes. In his work [34] Minsky proposed a two tapes non-writing machine composed of (i) a finite automaton and (ii) two semi-infinite tapes, each of which has a single special character denoting the end. These machines can move each tape in both directions and test when a tape reaches the end. Intuitively, the two tapes represent two nonnegative integer counters. A move, on the right or on the left, of heads over the tapes represents arithmetic operations $+1$ and -1, respectively, and testing if the head reaches the end of the tape realizes a test against zero for the counter representing the tape. By defining a suitable encoding representing configurations of a generic Turing machine it is possible to prove the equivalence between the class of machine introduced in [34] and the one of Turing machines. The main result concerning the equivalence is given by the next theorem:

Theorem 9 (theorem 1a, [34]). *Any partial recursive function $T(n)$ can be represented by a program operating on two positive integers c_1 and c_2 using only instructions of the form:*

(i): *$c_i + 1$; goto I.*

(ii): *$c_i - 1$; if $(c_i = 0)$ goto I_i else goto I_j.*

The program is defined by a set of instructions such that if it starts at I_{start} with $c_1 = 2^n$ and $c_2 = 0$ the program will eventually terminates at I_{end} with $c_1 = 2^{T(n)}$ and $c_2 = 0$.

Undecidability of the halting problem for the class of two-tape non-writing machines is a direct consequence of the theorem. Actually, Turing machines can be thought as programmed computer and programs as a sequence of instructions each of which specifies (i) an operation to be performed and (ii) the next instruction to be executed. Therefore, any formalism which is able to simulate such class of machine inherits full expressiveness and undecidability from Turing machines.

2.6. Counter Systems

Counter systems are finite-state automata endowed with a finite set of counters $V = \{x_1, \ldots, x_n\}$ over a specific domain D where $D \in \{\mathbb{N}, \mathbb{Z}, \mathbb{R}\}$. Some notations of this section are inspired to Section 1.4 in [11] by Demri and sometimes taken from [9] by Bersani and Demri. The list of problems in next Section 2.6.1 shares similarities from Section 1.4.2 in [11] by Demri.

Definition 10. *A* counter system *is a tuple* (Q, n, δ) *such that:*

- Q *is a finite set of control states;*
- $n \geq 1$ *is the* dimension *of the system, i.e., the number of counters involved in the system;*
- δ *is the* transition relation *defined as a finite set of triples of the form* (q, ξ, q') *where* $q, q' \in Q$ *are two control state and* ξ *is a Presburger formula with free variables in the set* $\{x_1, \ldots, x_n\} \cup \{x'_1, \ldots, x'_n\}$.

Elements $(q, \xi, q') \in \delta$ are called *transitions*. A *configuration* of (Q, n, δ) is defined as a pair $(q, \boldsymbol{x}) \in Q \times D^n$, where $\boldsymbol{x}$ is the vector of counters values. The *one-step transition relation* $\rightarrow \subseteq Q \times \mathcal{D}^n \times Q \times \mathcal{D}^n$ is defined between a pair of configurations such that $((q, \boldsymbol{x}), (q', \boldsymbol{x}')) \in \rightarrow$ when all the following conditions hold:

- there is a transition $t = q \xrightarrow{\xi} q'$ in δ,
- there exists a valuation $\mathbf{val}$ such that $\mathbf{val} \models_{\mathrm{PA}} \xi$ and
- for all $1 \leq i \leq n$, $\mathbf{val}(x_i) = \boldsymbol{x}_i$ and $\mathbf{val}(x'_i) = \boldsymbol{x}'_i$.

The (*binary*) *reachability relation* , written $\rightarrow^*$ is the reflexive and transitive closure of $\rightarrow$. A *run* ρ is a (possibly infinite) sequence of configurations $(q_0, \boldsymbol{x}_0), (q_1, \boldsymbol{x}_1) \ldots$ such that two successive configurations agree with δ, i.e. for $i \geq 0$, we have $(q_i, \boldsymbol{x}_i) \xrightarrow{t} (q_{i+1}, \boldsymbol{x}_{i+1})$, for some $t \in \delta$. When $D = \mathbb{N}$, runs are finite or infinite sequences of configurations such that $\boldsymbol{x}_i \geq 0$ for all positions in the run. We write $\rho(i)$ to denote the configuration $(q_i, \boldsymbol{x}_i)$ in position i and $\mathit{length}(\rho) \in \mathbb{N} \cup \infty$ to denote the length of ρ. An *initialized* counter system is a pair $(\mathcal{S}, (q, \boldsymbol{x}))$ such that $\mathcal{S}$ is a counter system and $(q, \boldsymbol{x})$ is an *initial configuration* (with $\boldsymbol{x} \geq 0$).

Given a subset L of PA, we write CS(L) to denote the class of counter systems for which transitions are restricted to guards in L and D agree with the definition of L.

A fundamental notion for studying counter systems is the reachability set of control states and reachability relation between pair of control states. Informally, given a control state $q \in Q$, it consists of the minimal set of all vectors $\boldsymbol{x}'$ which the system can reach by means of runs starting from a configuration $(q, \boldsymbol{x})$.

Definition 11. *Let* $(\mathcal{S}, (q, \boldsymbol{x}))$ *be an initialized counter system. The* reachability set *for the control state* $q' \in Q$ *is the set:*

$$R_q = \{\boldsymbol{x}' \in D^n \mid (q, \boldsymbol{x}) \xrightarrow{*} (q', \boldsymbol{x}')\}$$

Definition 12. *Let* $(\mathcal{S}, (q, \boldsymbol{x}))$*, where* $\mathcal{S} = (Q, n, \delta)$*. The* reachability relation *between two control states* $q, q' \in Q$ *is the set of pairs:*

$$R_{(q,q')} = \{(\boldsymbol{x}, \boldsymbol{x}') \in D^n \times D^n \mid (q, \boldsymbol{x}) \xrightarrow{*} (q', \boldsymbol{x}')\}$$

As we will see later, some subclasses of CS(PA) are characterized by semilinear reachability sets and reachability relations. When we provide effective procedure defining exactly the sets, effective verification can be performed by taking advantage of tool manipulating Presburger arithmetic.

2.6.1. Problems on counter systems

In this section, we briefly list some interesting problems concerning verification of counter systems which are involved in Chapter 4.

REACHABILITY PROBLEM:

Input	- a counter system $\mathcal{S} = (Q, n, \delta)$ - two configurations $(q, \boldsymbol{x})$, $(q', \boldsymbol{x}')$
Problem	does there exist a *finite* run ρ of $\mathcal{S}$ of length $length(\rho) = l$ such that $\rho(0) = (q, \boldsymbol{x})$ and $\rho(l) = (q', \boldsymbol{x}')$?

CONTROL STATE REACHABILITY PROBLEM:

Input	- a counter system $\mathcal{S} = (Q, n, \delta)$, - a configurations $(q, \boldsymbol{x})$ and a control state $q' \in Q$
Problem	does there exist a *finite* run ρ of $\mathcal{S}$ of length $length(\rho) = l$ such that $\rho(0) = (q, \boldsymbol{x})$ and $\rho(l) = (q', \boldsymbol{x}')$?

L MODEL-CHECKING PROBLEM:

Input	- a counter system $\mathcal{S} = (Q, n, \delta)$ - a configurations $(q, \boldsymbol{x})$ - a formula φ in a logical formalism L
Problem	**Existential problem:** does there exist a run ρ of $\mathcal{S}$ such that $\rho(0) = (q, \boldsymbol{x})$ and satifying φ, written $\rho \models \varphi$? **Universal problem:** is the case that all runs ρ of $\mathcal{S}$ such that $\rho(0) = (q, \boldsymbol{x})$ and satifying φ, written $\rho \models \varphi$?

The definition of counter systems provided above can be extended by adding a finite alphabet Σ whose symbols are involved in the definition of the transition relation δ. A counter system is, then, a tuple (Q, n, δ, Σ) where Q, n, δ are the control states set, the dimension of the system and the transition relation and $\Sigma = \{a_1, \ldots, a_m\}$ a finite set of symbols.

It is worth noticing that the class of Minsky machines constitutes a fragment of the class of counter systems are included in CS(QFP). Therefore, most interesting problems on counter systems are undecidable. For this reason, in order to get decidability, some restrictions have to be imposed on the nature of the systems. Briefly, additional requirements which are used to regain decidability are of the form:

- syntactic restrictions on formulae,
- restrictions on control graph defining the finite-state automata,
- semantic restrictions on runs.

Various classes are defined by combining previous restrictions on automata. The following section Section 2.6.2 considers reversal-bounded counter systems along with some fundamental known results concerning decidability and complexity issues involved in verification. Specific results concerning the work presented in the thesis will be shown in Section 4.

2.6.2. Reversal bounded counter systems

A careful analysis of the undecidability of the halting problem for two-tapes (i.e., two-counters) machines explained in Section 2.5 reveals that during computations, counters operates by incrementing/decrementing phases which start from a nonnegative value and ends to 0 and viceversa. Counters alternate nonincreasing to nondecreasing phases which are delimited by zero tests. The notion of reversal-boundedness introduced in [3] is based on a semantical restriction, that entails the decidability of reachability problems, imposing a limit on the number of changes between consecutive nonincreasing and nondecreasing phases. Informally, a *reversal* for a counter occurs in a run when there is an alternation from nonincreasing to nondecreasing mode. For instance, the sequence of values

$$000112222332\mathbf{2}21\mathbf{2}233344444\mathbf{3}333$$

is characterized by three reversals. We now define formally the notion of reversal-boundedness and we draw some main related results.

Following notations are taken from [9] by Bersani and Demri. Let $\mathcal{S} = (Q, n, \delta)$ be a counter system and $\{x_1, \ldots, x_n\}$ be the set of counters. From a run $\rho = (q_0, \boldsymbol{x}_0), (q_1, \boldsymbol{x}_1), \ldots$ of $\mathcal{S}$, in order to describe the behavior of counters and terms varying along ρ, we define a sequence of *mode vectors* $\boldsymbol{m}_0, \boldsymbol{m}_1, \ldots$ (of the same length as ρ) such that each $\boldsymbol{m}_i$ belongs to $\{\nearrow, \searrow\}^n$. Intuitively, each value in a mode vector records whether a counter is currently in an increasing phases or in an decreasing phases.

We are now ready to define the sequence $\boldsymbol{m}_0, \boldsymbol{m}_1, \ldots$

- By convention, $\boldsymbol{m}_0$ is the unique vector in $\{\nearrow\}^n$.
- For $j \geq 0$ and $i \in [1, n]$ we have:
  1. $\boldsymbol{m}_{j+1}(i) \stackrel{\text{def}}{=} \boldsymbol{m}_j(i)$ when $\boldsymbol{x}_j(i) = \boldsymbol{x}_{j+1}(i)$,
  2. $\boldsymbol{m}_{j+1}(i) \stackrel{\text{def}}{=} \nearrow$ when $\boldsymbol{x}_{j+1}(i) - \boldsymbol{x}_j(i) > 0$,
  3. $\boldsymbol{m}_{j+1}(i) \stackrel{\text{def}}{=} \searrow$ when $\boldsymbol{x}_{j+1}(i) - \boldsymbol{x}_j(i) < 0$.

Let $Rev_i = \{j \in \mathbb{N} : \boldsymbol{m}_j(i) \neq \boldsymbol{m}_{j+1}(i)\}$.

Definition 13. *A ρ is* r-reversal-bounded *for some $r \geq 0$ $\stackrel{def}{\Leftrightarrow}$ for all $i \in [1, n]$,* $\text{card}(Rev_i) \leq r$.

The definition of reversal-boundedness over runs can be naturally extended to a counter systems by considering the set of all initialized runs from an initial configuration.

Definition 14. *Let $(q, \boldsymbol{x}) \subseteq Q \times \mathbb{N}^n$ be a configuration. An initialized counter system $(\mathcal{S}, (q, \boldsymbol{x}))$ is* reversal-bounded *when there is $r \geq 0$ such that every run from $(q, \boldsymbol{x})$ is r-reversal-bounded. A counter system $\mathcal{S}$ is* globally *reversal-bounded when it is reversal-bounded for all initial configurations.*

Reversal-boundedness is a semantic criterion restricting the set of runs of a counter system. Therefore, the reversal-boundedness detection problem is undecidable.

REVERSAL-BOUNDEDNESS DETECTION PROBLEM

Input	an initialized counter system $(\mathcal{S}, (q, \boldsymbol{x}))$ where $\mathcal{S} = (Q, n, \delta)$
Problem	Is $(\mathcal{S}, (q, \boldsymbol{x}))$ reversal-bounded?

Theorem 15 ([35]). *Reversal-boundedness detection problem is undecidable.*

In [3], Ibarra shows a fundamental result of decidability which exploits reversal-boundedness of runs. The fragment of CS(PA) considered is defined by counter systems (Q, n, δ) such that the transition relation δ is defined by tuples of the form $(q, (zero(\boldsymbol{b}), \boldsymbol{d}), q')$ which are a shorthand for (q, ξ, q') where:

$$\xi := \bigwedge_{i\in[1,n]:\vec{b}(i)=1} x_i = 0 \ \wedge \bigwedge_{i\in[1,n]:\vec{b}(i)=0} x_i \neq 0 \ \wedge \bigwedge_{i\in[1,n]} x'_i = x + \boldsymbol{d}(i).$$

Vector $\boldsymbol{d}$ is called *update vector*.

Theorem 16 ([3]). *Let $(\mathcal{S}, (q, \boldsymbol{x}))$ be an initialized counter system which is of the form defined above and r-reversal-bounded for some $r \geq 0$. Then, for every $q' \in Q$, the set $R_{q'}$ is semilinear. When $\mathcal{S}$ is globally r-reversal-bounded then, for any pair (q, q') where $q, q' \in Q$, the set $R_{(q,q')}$ is semilinear.*

The immediate consequence of the theorem is that one can compute a Presburger formula φ_q with $free(\varphi_q) = \{x_1 \ldots, x_n\}$ which defines exactly the reachable configurations characterizing q.

Corollary 17 ([3]). *Let $\mathcal{S} = (Q, n, \delta)$ be a counter system, $q, q' \in Q$ two control states, $\boldsymbol{x} \in \mathbb{N}^n$ and $r \geq 0$.*

- *if $\mathcal{S}$ is globally reversal-bounded, then it is possible to compute a Presburger formula $\varphi_{(q,q')}$ with $free(\varphi_{(q,q')}) = \{x_1 \ldots, x_n\} \cup \{x'_1 \ldots, x'_n\}$ such that for every valuation $(\boldsymbol{x}, \boldsymbol{x}')$*

$$(\boldsymbol{x}, \boldsymbol{x}') \models_{\text{PA}} \varphi_{(q,q')} \Leftrightarrow (\boldsymbol{x}, \boldsymbol{x}') \in R_{(q,q')}$$

- *if $(\mathcal{S}, (q, \boldsymbol{x}))$ is r-reversal-bounded, then it is possible to compute a Presburger formula $\varphi_{q'}$ with $free(\varphi_{q'}) = \{x'_1 \ldots, x'_n\}$ such that for every valuation $\boldsymbol{x}'$*

$$\boldsymbol{x}' \models_{\text{PA}} \varphi_{q'} \Leftrightarrow \boldsymbol{x}' \in R_{q'}$$

As immediate consequence of previous corollary we have that reachability problem for r-reversal-bounded counter systems is decidable. In Chapter 4 we present an analogous result for a richer reversal-boundedness property.

The model presented so far can be enriched with a free counter for which no restrictions on the number of reversal are imposed. The same semilinearity results hold also for the augmented class of systems. In fact, given a r-reversal-bounded counter system $\mathcal{S}$ which has one free

counter, we can build a 1-reversal-bounded counter system $\mathcal{S}'$ such that the free counter behaves in the same way of $\mathcal{S}$. One free counter can be simulated by a single stack with unary alphabet. Actually, Parikh's theorem can be applied also for pushdown automaton with unary stack alphabet. Then, semilinearity (and effective Presburger definability) can be derived in the same way as Theorem 16.

We provide now some results concerning complexity analysis for reachability problem which are related to the analysis presented in Chapter 4. The analysis of Ibarra's original procedure in [3] reveals that the algorithm to decide reachability problem for a reversal-bounded counter system is not optimal. In fact, construction of Parikh image for a (nondeterministic) finite-state automaton runs in exponential time for general cases, in the size of the alphabet considered and the dimension of the automaton, and may yield results which are union of exponentially many linear sets, in the worst case. Let n and r be the number of counters and the number of reversal of a counter system; let p be the cardinal of control state set. Parikh's construction is applied in [3] to a (nondeterministic) finite-state automaton of size exponential in r and n and polynomial in the number p; the cardinal of the set of control states of $\mathcal{A}$ in Theorem 16 is $p \cdot r^n \cdot 3^{3(1+\frac{r}{2})}$. Then, the exponential construction of Parikh's image results in double exponential complexity in r and n and exponential in n. Observe that, if the number r is given in binary, then the complexity is triple exponential in r. In [36], To provides an algorithm which, given a nondeterministic finite-state automaton with n states over an alphabet of size $k \geq 1$, computes a union of linear sets with at most k periods and total size $2^{\mathcal{O}(k^2 log n)}$. When the alphabet is unary, i.e., $k = 1$, as in the case of counter systems considered in the thesis, the algorithm runs in polynomial time in the size of the automaton. Consequently, Ibarra's method deciding reachability results in exponential complexity in r and n and polynomial in n (for r in binary, complexity is double exponential in r). Gurari and Ibarra, who firstly observed the non-optimality of procedure in [3], proposed a technique to derive the upper and lower bound for nonemptiness problem (and, then, for reachability) for the class of reversal-bounded counter systems. The main result claims that if a 1-reversal-bounded counter systems accepts an input word then there exists a shorter accepting run of bounded length (which does not depend on the length of the input word). The shorter accepting run can be constructed as a sequence of subcomputation, each corresponding to a sequence of transition rules compatible with a vector mode. The analysis of the length of accepting runs is done by exploiting the property of small solutions for systems of equations which define the arithmetic behavior of counters in each subcomputation.

Lemma 18 (Theorem 3, [37]; Lemma 2, [38]). *Let $\mathcal{S} = (Q, n, \delta)$ be a 1-reversal-bounded counter system with alphabet Σ. There exists a fixed positive constant c such that $\mathscr{L}(\mathcal{S}) \neq \emptyset$ if, and only if, $\mathcal{S}$ accepts some input word within time $n|\delta|^{cn}$.*

The lemma can be used to prove the theorem:

Theorem 19 (Theorem 1, [38]). *Let n and r be fixed positive integer. Then, the nonemptiness problem for r-reversal-bounded n-counter systems is decidable in* PTIME.

The algorithm uses a Turing machine which simulates computation of the counter system given in input which accepts some words if, and only if, all its computation are of bounded length (then bounded time).

Theorem 20 (Theorem 3, [38]). *The nonemptiness problem for deterministic r-reversal-bounded 2-counter systems is* NP*-hard, even for system with unary alphabet. The result holds also for 1-reversal-bounded n-counter systems.*

Theorem 21 (Theorem 4, [38]). *The nonemptiness problem for deterministic r-reversal-bounded n-counter system is* PSPACE*-hard, even for system with unary alphabet.*

Later, these results were refined by Howell and Rosier [37] who presented an algorithm for deciding nonemptiness problem for nondeterministic r-reversal-bounded n-counter system. The algorithm exploits nondeterministic guesses to determine the accepting run of bounded length. For each subcomputation corresponding to a vector mode, the algorithm guesses in parallel a bounded number of loops of the system, i.e., by guessing how many loops each transition takes. Moreover, the algorithm requires the construction of a 1-reversal-bounded counter system with the same dimension of the original $\mathcal{S}$ which is done in polynomial time in the description of $\mathcal{S}$. The algorithm runs in NP if r is fixed or is given in unary. For binary encoding, it is polynomial in the description of $\mathcal{S}$ and EXPTIME in r. Lower bound for nonemptiness problem is given by constructing a deterministic counter systems with 3 counters which is $2^{O(l)}$-reversal-bounded, where l is the size of the input word. The intuition of the theorem is to build a deterministic system which tries a different configuration of reversal for each symbol of the input word. Since the input is of length l then there are at most $2^{O(l)}$ possible permutations of mode vectors, one for each symbol. When the number of reversal r is given in binary, we have the following result:

Theorem 22 (Theorem 3, [37]). *Nonemptiness problem for r-reversal-bounded n-counter systems is* NEXPTIME*-complete. The result holds also when systems are enriched with a free counter.*

The class of counter systems considered so far can be extended to a more general class of systems which use a richer language of constraints over transitions. In [39], authors proposed the following extensions:

a. counters are variables over $\mathbb{Z}$. By storing the signs of each counter in the control states, it is possible to simulate a counter system over Integers $\mathcal{S}$ by means of a counter system $\mathcal{S}'$ whose counters assume values over $\mathbb{N}$. If $\mathcal{S}$ is r-reversal-bounded then $\mathcal{S}'$ is r-reversal-bounded.

b. formulae ξ on transitions $q \xrightarrow{\xi} q'$ belong to language QFP($<$) of linear constraints or QFP($<_1$) of linear constraints in which only one free variable occurs, i.e., cardinal of $free(\xi)$ is one.

c. formulae ξ on transition are no longer conditional relations on counters but they can also define assignment of the form $x := y$ or $x := 0$ (reset) where x, y are two counters of the system (provided that free counter do not appear in any instruction of this form). Both the assignments can be simulated by constructing a counter system $\mathcal{S}'$ from $\mathcal{S}$ which preserves reversal-boundedness.

Extension **b.** requires a more detailed analysis. In order to preserve decidability of reachability problem (i.e., effective Presburger definability of the reachability relation) some restrictions on formulae over transitions are imposed. In fact, the unrestricted use of zero tests leads to undecidability since Minsky machines can be simulated by 0-reversal-bounded 4-counter systems in CS(QFP$<$) allowing conditionals of the form $x - y < 0$, since a zero test $x - y = 0$ can be rewritten into $x - y - 1 < 0 \wedge \neg(x - y < 0)$. Moreover, undecidability for halting problem holds even for three 0-reversal-bounded counters.

Theorem 23 (Theorem 4.2, [39]). *Let $\mathcal{S} \in$ CS(QFP$<$) be a deterministic 3-counter systems which uses only formulae of the form $x_1 = x_3$ and $x_2 = x_3$ and $\boldsymbol{d}(i) = \{-1, 0, +1\}$. Then, the halting problem is undecidable.*

Reachability problem for initialized r-reversal-bounded counter systems becomes decidable when transitions formulae belong to QFP($<_1$), since zero tests can not be represented. Note that this restriction do not affect formulae defining update vectors $\boldsymbol{d}$. Decidability (of nonemptiness)

in the case of CS(QFP($<_1$)) is provided in [39] by constructing a counter system $\mathcal{S}'$ from the given initial one $\mathcal{S}$ in which no formulae over transitions occur. Therefore, $\mathcal{S}'$ is a standard counter systems which simulates phases of $\mathcal{S}$ by guessing the truth value of constraints on transitions. Moreover, authors prove that formulae in CS(QFP($<$)) can be retained when the set of runs satisfy a stronger property characterizing reversals named *strong* reversal-boundedness. From a run $\rho = (q_0, \boldsymbol{x}_0), (q_1, \boldsymbol{x}_1), \ldots$ of $\mathcal{S}$, let $\boldsymbol{m}_0, \boldsymbol{m}_1, \ldots$ be the sequence of mode vectors such that each $\boldsymbol{m}_i$ belongs to $\{\nearrow, \searrow, \rightarrow\}^n$. According to the new definition, each value in a mode vector now records whether a counter is currently in increasing, decreasing or *no-change* phase. The sequence $\boldsymbol{m}_0, \boldsymbol{m}_1, \ldots$ is defined by the following updated rules:

- By convention, $\boldsymbol{m}_0$ is the unique vector in $\{\nearrow\}^n$.
- For $j \geq 0$ and $i \in [1, n]$ we have:
  1. $\boldsymbol{m}_{j+1}(i) \stackrel{\text{def}}{=} \rightarrow$ when $\boldsymbol{x}_j(i) = \boldsymbol{x}_{j+1}(i)$,
  2. $\boldsymbol{m}_{j+1}(i) \stackrel{\text{def}}{=} \nearrow$ when $\boldsymbol{x}_{j+1}(i) - \boldsymbol{x}_j(i) > 0$,
  3. $\boldsymbol{m}_{j+1}(i) \stackrel{\text{def}}{=} \searrow$ when $\boldsymbol{x}_{j+1}(i) - \boldsymbol{x}_j(i) < 0$.

Definition 24. *A run ρ is* strongly *r-reversal-bounded for some $r \geq 0$ $\stackrel{def}{\Leftrightarrow}$, for all $i \in [1, n]$,* $\text{card}(Rev_i) \leq r$.

The definition of reversal-boundedness over runs naturally extends to counter systems by considering the set of all initialized runs from an initial configuration. By assuming strong reversal-boundedness of a counter systems $\mathcal{S}$ using update vectors $\boldsymbol{d} = (-1, 0, +1)^n$, we have:

Theorem 25 ([39])**.** *Let $(\mathcal{S}, (q, \boldsymbol{x}))$, be an initialized r-strong-reversal-bounded counter system using* QFP($<$) *formulae on transitions. Then, for any $q \in Q$, the set R_q is semilinear. Let $\mathcal{S}$, be an initialized r-strong-reversal-bounded counter system using* QFP($<$) *formulae on transitions. Then, for any $q, q' \in Q$, the set $R_{(q,q')}$ is semilinear.*

When the assumption of strong-reversal-boundedness is relaxed, decidability is regained provided that systems involve QFP($<_1$) formulae but more general update vectors $\boldsymbol{d} \in \mathbb{Z}^n$.

Theorem 26 ([39])**.** *Let $(\mathcal{S}, (q, \boldsymbol{x}))$, be an initialized r-reversal-bounded counter system using* QFP($<_1$) *formulae on transitions. Then, for any $q \in Q$, the set R_q is semilinear. Let $\mathcal{S}$, be an initialized r-reversal-bounded counter system using* QFP($<_1$) *formulae on transitions. Then, for any $q, q' \in Q$, the set $R_{(q,q')}$ is semilinear.*

The reachability problem is naturally related to study of safety properties. Verification of safety properties is reduced to instances of reachability problem enriched by constraints which restrict values of counters at final configuration. Given a Presburger formula φ and an initialized system $(\mathcal{S}, (q, \boldsymbol{x}))$, an instance of safety problem consists of determining whether there exists a run of $(\mathcal{S}, (q, \boldsymbol{x}))$ such that values of counters at final configuration $(q', \boldsymbol{x}')$ satisfies P; i.e., $\boldsymbol{x}' \models_{\text{PA}} \varphi$. Liveness properties are investigated on infinite runs; given a Presburger formula φ, an infinite run ρ of a system $\mathcal{S}$ is φ-infinitely-often when there are infinitely many position i such that $\boldsymbol{x}_i \models_{\text{PA}} \varphi$, where $\rho(i) = (q_i, \boldsymbol{x}_i)$. An infinite run ρ is φ-almost-always when there is a position j such that $\boldsymbol{x}_i \models_{\text{PA}} \varphi$, for all position $i \geq j$.

Repeated reachability problem plays a particular and fundamental role for the decidability of model-checking problem when liveness properties are considered.

CONTROL STATE REPEATED REACHABILITY PROBLEM

Input	- counter system $\mathcal{S} = (Q, n, \delta)$ - a configurations $(q, \boldsymbol{x})$ - a control state $q' \in Q$
Problem	does there exist a *infinite* run ρ of $\mathcal{S}$ such that $\rho(0) = (q, \boldsymbol{x})$ and q' is visited infinitely often?

∃-PRESBURGER INFINITELY OFTEN PROBLEM:

Input	- a counter system $\mathcal{S} = (Q, n, \delta)$ - a configurations $(q, \boldsymbol{x})$ - a Presburger formula φ
Problem	does there exist an *infinite* run ρ of $\mathcal{S}$ such that $\rho(0) = (q, \boldsymbol{x})$ and φ is satisfied infinitely often; i.e., there are infinitely many position i such that $\boldsymbol{x}_i \models_{\text{PA}} \varphi$, where $\rho(i) = (q_i, \boldsymbol{x}_i)$?

The complement problem of a ∃-Presburger infinitely often problem is:

∀-Presburger almost always problem

Input	- a counter system $\mathcal{S} = (Q, n, \delta)$ - a configurations $(q, \boldsymbol{x})$ - a Presburger formula φ
Problem	does there exist an *infinite* run ρ of $\mathcal{S}$ such that $\rho(0) = (q, \boldsymbol{x})$ and φ is satisfied almost always; i.e., there is a position $j > 0$ such that $\boldsymbol{x}_i \models_{\text{PA}} \varphi$ for all $i \geq j$, where $\rho(i) = (q_i, \boldsymbol{x}_i)$?

Dang, Ibarra and San Pietro investigate these problems and show their decidability [40].

Theorem 27 (Lemma 8, [41]). *Let $\mathcal{S}$ be a r-reversal-bounded n-counter system and one free counter. Then, ∃-Presburger infinitely often problem and ∀-Presburger almost always problem are decidable.*

The technique suggested to prove decidability exploits reduction of ∃-Presburger infinitely often problem (∀-Presburger almost always problem) to a finite amount of reachability problem on 0-reversal-bounded counter systems built from the original system. A similar construction will be used in Chapter 4 to solve our version of reversal bounded model-checking problem. It exploits a simple but essential property characterizing the class of reversal-bounded counter systems: for each infinite run there exists a position $h \geq 0$ such that all configurations at position $i \geq h$ are compatible with a unique mode vector $\boldsymbol{m}_h$. Then, all infinite computations of $\mathcal{S}$, whose counters agree with the mode $\boldsymbol{m}_h$ after position h, can be represented by a 0-reversal-bounded counter system $\mathcal{S}^{\boldsymbol{m}_h}$. Transition relation of $\mathcal{S}^{\boldsymbol{m}_h}$ consists of only transitions compatible with $\boldsymbol{m}_h$. Moreover, transitions are refined since they are enriched by formulae which guarantee suitable value for counters throughout infinite runs. Repeated reachability and ∃-Presburger infinitely often problem (∀-Presburger almost always problem) are solved by investigating existence of runs of $\mathcal{S}^{\boldsymbol{m}_h}$ starting from a configuration $(q, \boldsymbol{x})$ reaching a configuration $(q, \boldsymbol{x} + \boldsymbol{\Delta})$ at the same control state q.

Lemma 28 (Lemma 8, [41]). *Let $\mathcal{S} = (Q, n, \delta)$ be a r-reversal-bounded counter system, $q \in Q$ be a control state, φ a Presburger formula and I be a Presburger formula defining initial configuration for $\mathcal{S}$. There exists a φ-infinitely-often (-almost-always) run of $\mathcal{S}$ starting from a configuration $(q, \boldsymbol{x})$ if, and only if, for some mode vector $\boldsymbol{m}$, there exists a φ-infinitely-often (-almost-always) run ρ' in $\mathcal{S}^{\boldsymbol{m}}$.*

By the lemma we have:

Theorem 29 (Theorems 2,3; [41]). *The $\exists$-Presburger infinitely often, $\forall$-Presburger almost-always and control state repeated reachability problems are decidable for the class of r-reversal-bounded counter system. The result holds also when one free counter is added to the system.*

Let consider the problem ($\exists$-Presburger-always) of checking whether there exists a *infinite* run ρ of $\mathcal{S}$ such that $\rho(0) = (q, \boldsymbol{x})$ and for all position i we have $\boldsymbol{x}_i \models_{\text{PA}} \varphi$, where $\rho(i) = (q_i, \boldsymbol{x}_i)$. In this case, Dang et al. in [40] prove that the reachability set is not recursive. This result is worth to be noticed if it is compared with decidability results which we have obtained in Section 4.

2.7. Temporal logic over arithmetic constraints

Modal Logics [42] were introduced by philosophers to reason about "modes" of truth of elements characterizing the universe of reasoning. In particular, when we establish a language to reason about elements in the universe, the truth value of assertions in that language is modified by the use of *modalities*. A modality is defined to be a *syntactic* element belonging to the language which characterizes the truth of a predicate; for instance, if P is a predicate which can be true or false, the sentence "possibly" P is true if it is possible to make P true. Temporal logic is a special class of Modal Logic which provides a formal system to reason about how truth of assertions changes over time. Amir Pnueli [43] was been the pioneer of temporal logic in computer science; he argued that temporal logic can be used to reason about processes and computations. In particular, this formal system is particularly appropriate when nonterminating, continuously operating systems are considered: they maintain an "infinite" computation representing interactions with the environment. This peculiarity makes temporal logic successful and effective if it is compared with formalisms like Hoare logic which are based on transformations describing semantics of programs.

A fundamental question concerns the nature of time. Various distinct alternatives exist; here, we list the main ones:

- linear time, branching time;
- instant-based or interval-based;
- discrete time or dense time.

When the course of time is linear, at each moment, there exists only one possible future instant after the current one. This nature of time is

useful when we want to reason about properties of all possible behaviors the system can execute from the current instant. On the other side, branching time allows alternative courses, representing different possible futures, to be realized from the current instant. Next section recall the definition of Linear Temporal Logic which is the temporal language adopted throughout this thesis. Some fundamental results are also summarized.

2.7.1. Linear temporal logic - LTL

Sentences, or *formulae*, of the language LTL are defined by the grammar:

$$\phi := p \mid \neg\phi \mid \phi \wedge \psi \mid \mathbf{X}\phi \mid \phi \mathbf{U} \psi$$

We assume $\top$ is the symbol representing true value, p is a symbol of the set AP of atomic propositions, $\neg$ and $\wedge$ are the usual boolean connectives for negation and conjunction and , are the temporal modalities for "next-time" and "until". Other formulae are introduced as abbreviation: $\phi \vee \psi$ instead of $\neg(\neg\phi \wedge \neg\psi)$; $p \Rightarrow q$ abbreviates $\neg p \vee q$ and $p \Leftrightarrow q$ abbreviates $(p \Rightarrow q) \wedge (q \Rightarrow p)$. The boolean constant $\top$ is a shorthand for $p \vee \neg p$ and $\bot$ for $\neg\top$.

Semantics of LTL formulae is defined with respect to a structure (S, π, L) where:

- S be a set of states.
- π is an infinite sequence of states $\pi \in S^\omega$; the structure of time is a totally ordered set which we assume to be isomorphic to $(\mathbb{N}, <)$.
- $L : S \to 2^{AP}$ is a labeling function which defines the set of true atoms for each state.

A satisfaction relation can be equivalently defined from π and w; we prefer the first one:

$$\begin{aligned}
\pi, i \models p &\stackrel{\text{def}}{\Leftrightarrow} p \in L(s_i) \text{ for } p \in AP \\
\pi, i \models \neg\phi &\stackrel{\text{def}}{\Leftrightarrow} \pi, i \not\models \phi \\
\pi, i \models \phi \wedge \psi &\stackrel{\text{def}}{\Leftrightarrow} \pi, i \models \phi \text{ and } \pi, i \models \psi \\
\pi, i \models \mathbf{X}\phi &\stackrel{\text{def}}{\Leftrightarrow} \pi, i+1 \models \phi \\
\pi, i \models \phi \mathbf{U} \psi &\stackrel{\text{def}}{\Leftrightarrow} \exists j \geq i : \pi, j \models \psi \text{ and } \forall i \leq n < j \ \pi, n \models \phi
\end{aligned}$$

A formula $\phi \in$ LTL is *(initially) satisfiable* if there exists a linear time structure (S, π, L) such that $\pi, 0 \models \phi$. In this case, we say that (S, π, L),

or simply π, is a model for ϕ. Equivalently, a model for a LTL formula is an infinite sequence of states $w \in (2^{AP})^\omega$. $\phi \in$ LTL is satisfiable if there exists a word $w \in (2^{AP})^\omega$ such that $w, 0 \models \phi$. In this case, the definition of $\models$ is the same as the one presented above except for the case of atomic proposition:

$$w, i \models p \overset{\text{def}}{\Leftrightarrow} p \in w(i) \text{ for } p \in AP.$$

Two LTL formulae ϕ and ψ are (*initially*) *equivalent*, written $\phi \equiv_i \psi$ when for all linear time strucure (S, π, L), $(\pi, 0) \models \phi$ if, and only if, $(\pi, 0) \models \psi$.

Past-time operators , "yesterday", and "since" can be added to the LTL language. Though they do not augment expressiveness, they make formulae exponentially more succinct than an equivalent formula using only future modalities: this result is proved by Markey in [44]. The extension of LTL with past-time modalities will be denoted as LTLB.

$$\pi, i \models \mathbf{Y}\phi \overset{\text{def}}{\Leftrightarrow} \pi, i-1 \models \phi \text{ and } i > 0$$

$$\pi, i \models \phi \mathbf{S} \psi \overset{\text{def}}{\Leftrightarrow} \exists 0 \leq j \leq i : \pi, j \models \psi \text{ and } \forall j < n \leq i \; \pi, n \models \phi$$

When past-time modalities are included in the language, the position of instant at which a formula is evaluated with respect to the sequence π yields different notions of equivalence and satisfiability. A LTL formula ϕ is *globally satisfiable* when there exists a linear time structure (S, π, L) such that $\pi, i \models \phi$ for some $i \geq 0$. Two LTL formulae ϕ and ψ are *globally equivalent*, written $\phi \equiv_g \psi$, when for all linear time strucure (S, π, L) and for all $i \geq 0$, $\pi, i \models \phi$ if, and only if, $\pi, i \models \psi$.

Theorem 30 (Theorem 3.1, [45]). *LTLB is strictly more expressive than LTL with respect to global equivalence $\equiv_g$. The two logic are equi expressive when initial equivalence $\equiv_i$ is considered.*

Proof. The proof is a summary of the proof of Theorem 3.1 in [45]. Let q be an atomic proposition belonging to AP and $\pi = q(AP)^\omega$, $\pi' = \neg q(AP)^\omega$ be two sequences over AP. The two sequences π and π' are identical except for the first position at 0 instant. While LTLB formula $\mathbf{Y}p$ distinguishes the two model at instant 1, i.e., $\pi, 1 \models \mathbf{Y}p$ and $\pi', 1 \not\models \mathbf{Y}p$, LTL formulae can not distinguish $\pi, 1$ from $\pi', 1$. In its PhD work [46], Kamp proved that the first-order theory of linear order with equality, order relation $<$ and monadic predicates, written $\mathbf{FO}^m(<, =)$ is equivalent to LTLB. Gabbay et al. provides in [47] the justification for the second part of the proposition. They show that for every $\mathbf{FO}^m(<, =)$ formula χ there exists a LTL formula ϕ such that $\phi \equiv_i \chi$. □

Given a LTL formulae ϕ we write $cl(\phi)$ to denote the smallest set containing all subformulae of ϕ that is also closed under negation.

2.7.2. Satisfiability problem for LTL and LTL Model Checking

Informally, satisfiability problem for a logic language is the problem of finding a model satisfying a given formula, i.e., an assignment for the element of the language defining the formula which makes the evaluation of the formula true. Along with the validity problem, satisfiability (and validity) is of fundamental interest when a logic language is defined. Various approaches can be used to show their decidability, while undecidability is typically obtained by reducing an undecidable problem to an instance of them. Hereafter, we remind some fundamental techniques to prove decidability for satisfiability problem. Techniques using tableaux build a graph which represents true elements of a formula encoding potential models for it. A formula is satisfiable when its correspondent tableau satisfies a suitable consistency property. Consistency can then be tested to check if the tableau contains proper models for the formula. Tableaux-based approaches can be distinguished between declarative and incremental methods. Declarative methods [48] [49] first generate all possible sets of subformulae of a given formula and then they eliminate some (possibly all) of them. Incremental methods [50] generate only "meaningful" sets of subformulae. A different approach based on automata is proposed by Wolper and Vardi [51] for which Büchi automata are built to recognize the language of models of LTL formulae. Satisfiability problem is then reduced to reachability of control states on automata which is decidable in NLOGSPACE in the dimension of the graph representing the automata. Main results presented in Chapters 3 and 4 are based on this construction.

SATISFIABILITY OF LTL FORMULA:

Input	a LTL formula ϕ
Problem	does there exists a model $w \in (2^{AP})^{\omega}$ such that $w, 0 \models \phi$?

LTL model-checking problem is the problem of verifying the existence of an execution of an automata-based model which satisfies a given formula of the logic language, i.e., the execution is a model for the formula. Satisfiability and model-checking problems are often strictly related since they are reciprocally reducible to each other. Though model-checking can be defined with respect to different formalisms defining models for formulae (automata, transitions systems, automatic transitions systems), we give main results when transitions systems are considered and, in particular, we focus on the class of Kripke structures. A Kripke structure

is a triple $M = (S, R, L)$ where:

- S is a non-empty set of states,
- $R \subseteq S \times S$ is a binary relation between states
- $L : S \to 2^{AP}$ is a labeling function which characterizes true element on states.

A *path* in M is a sequence $s_0 s_1$, finite or infinite, such that $(s_i, s_{i+1}) \in R$; i.e., a path is a word of the language $(2^{AP})^\omega$. We write $P(M, s)$ to denote the set of infinite paths of M starting from s.

The *existential* LTL model-checking problem for LTL is defined as:

LTL MODEL-CHECKING PROBLEM:

Input	- a Kripke structure $M = (S, R, L)$ - a state $s_0 \in S$ - a LTL formula ϕ
Problem	does there exist an *infinite* path $\pi \in s_0 S^\omega$ of M such that $\pi, 0 \models \phi$, written $M, s_0 \models \phi$?

We remind main results concerning complexity of the two problems. Most of them can be found in the work of Sistla and Clarke [52] where complexity analysis is presented also for fragments of LTL language.

Theorem 31 (Lemma 4.3, [52]; [53]). *The satisfiability problem for LTL is* PSPACE*-hard.*

An immediate reduction demonstrating PSPACE-hardness of LTL model-checking is defined by reducing satisfiability problem of LTL formulae to LTL model-checking with respect to a Kripke structure defined by $M = (\{s\}, R, L)$ with $(s, s) \in R$ and $L : \{s\} \to \top$. Then, given a LTL formula ϕ, ϕ is satisfiable if, and only if, $M, 0 \models \phi$. PSPACE-hardness of LTL model-checking problem is also proved by reducing satisfiability of Quantified Boolean Formulae to LTL model-checking. The satisfiability problem for Quantified Boolean Formulae is known to PSPACE-complete by Stockmeyer and Meyer in [54].

Theorem 32 (Lemma 4.4, [52]). *LTL model-checking problem is* PSPACE*-hard.*

Sistla and Clarke proved the same results with a different approach. LTL model-checking is first shown to be polynomial reducible to LTL satisfiability problem. PSPACE-hardness is derived from a deterministic Turing machine which is $S(n)$-space bounded by a polynomial in the

dimension n of the LTL model-checking problem. PSPACE upper bound results from constructing a nondeterministic Turing machine solving satisfiability problem in polynomial space in the dimension of the formula. Using Savitch's theorem [55], it follows that there is a polynomial space bounded deterministic Turing machine that decides satisfiability.

Theorem 33 (Theorem 4.1, [52]). *The satisfiability problem for LTL formulae is* PSPACE*-complete.*

The most popular method for solving satisfiability and model-checking problem for LTL consists of reducing the problems to problems on Büchi automata representing formulae. A fundamental result is established by Vardi and Wolper in [51] for which, for a given LTL formula ϕ, one can define a Büchi automaton $\mathcal{A}_\phi$ recognizing models of ϕ, i.e., the language $\mathscr{L}(\mathcal{A}_\phi)$ is the set of models satisfying ϕ. Essentially, the automaton keeps track of the set of subformulae which are currently satisfied. The structure of the automaton is such that all the subformulae are, eventually, satisfied.

Theorem 34 (Theorem 2.1, [51]). *For every LTL formula ϕ there exists a Büchi automaton $\mathcal{A}_\phi$ such that $\rho \models \phi$ if, and only if, $\rho \in \mathscr{L}(\mathcal{A}_\phi)$.*

The automaton $\mathcal{A}_\phi$ for a given formula ϕ can be effectively computed in polynomial space in the dimension of the formula. By combining lower and upper bound results we get the PSPACE-completeness for the satisfiability and LTL model-checking problem.

Theorem 35. *Satisfiability problem for LTL formulae and LTL model-checking problems are* PSPACE*-complete.*

2.7.3. Bounded LTL model-checking

First model-checking algorithms implement explicitly enumeration of reachable states of systems in order to represent their legal executions. When non trivial systems are considered, the number of explored states is often exponential in the number of variables defining the system. Consequently, the state space exploration process becomes rapidly unfeasible and proves to be inappropriate for handling examples of real complexity. Various techniques was developed during past decades to reduce the computational demand of resources. *Symbolic model checking* [56] is a technique introduced by Burch, Clarke and McMillan which exploits an efficient symbolic representation of states using boolean functions. Rather than enumerating explicitly all states of the system they are implicitly represented by a boolean formula. Bryant showed that boolean formulae

can be efficiently represented by Binary Decision Diagrams (BDD) [57] and manipulating BDDs can be done efficiently since boolean operations reduces to simple operations on graphs. In [56] authors show a model-checking algorithm for μ-calculus [58] and for Computational Tree Logic (CTL) [59]. Since theoretical complexity of LTL model-checking is not appealing, the effort involved in the past research for defining efficient algorithm leading to practical verification was limited. A first attempt to make LTL model-checking feasible in terms of practical implementation was proposed by Clarke, Grumberg and Hamaguchi in [60]. They showed LTL model-checking problem is reducible to CTL model-checking problem provided that suitable fairness constraints are enforced. A *fairness constraint* is a CTL formula; a run is *fair* with respect to a set of fairness constraints when each fairness constraint holds infinitely often along the run. The proposed algorithm builds first a tableau T_ϕ for the LTL formula ϕ which is, actually, a Kriple structure whose control states represents all satisfied subformulae of ϕ. The structure $M \times T_\phi$, representing runs of M satisfying ϕ, is built by composing the Kripke structure M and the tableau T_ϕ . CTL model-checking algorithm is then invoked on $M \times T_\phi$ with suitable fairness constraints requiring that all until formulae of the form $\psi \mathbf{U} \zeta$ are infinitely often satisfied. All runs satisfying fairness constraints have the property that no subformula $\psi \mathbf{U} \zeta$ holds almost always in the run while ζ remains false.

Bounded model-checking is an alternative technique which was first proposed by Biere et al. in [61]. Clearly, it does not solve the theoretical complexity of LTL model-checking but the practice has shown that it can solve many cases which can not be handled by BDD-based techniques. Bounded model-checking amounts to search counterexample in executions whose length is bounded by some nonnegative integer k. If no counterexample is found, then k is increased until either a counterexample is detected or the SAT problem becomes intractable or a special upper bound, called *completeness threshold* is reached. The model-checking problem is efficiently reduced to SAT since finite runs of a systems can be finitely represented.

BOUNDED LTL MODEL-CHECKING PROBLEM

Input	- a Kripke structure $M = (S, R, L)$; a state $s_0 \in S$ - a LTL formula ϕ a nonnegative integer k
Problem	does there exist a *finite* path $\pi \in S^k$ of M such that: • if $\pi = uv$, where $u \in s_0 S^*$, $v \in S^+$, then $uv^\omega, 0 \models \phi$; or, • if $\pi = u$, where $u \in s_0 S^{k-1}$, then $uS^\omega, 0 \models \phi$. written $M, s_0 \models \phi$

Although runs p of M are of finite length, they still represent infinite paths. However, the satisfiability relation of LTL no longer can be applied in the case of finite paths. Therefore, a new relation $\models_k$ of satisfaction will be later defined in order to deal with finite representation of infinite models. When $p = uv$, the finite path is interpreted as an ultimately periodic run $p = uv^\omega$. The satisfaction relation $\models_k$ adopted to evaluate satisfiability of ϕ with respect to p is the same as $\models$ of standard LTL. In the case of runs of the form $p = u$, the prefix u represents all possible infinite set of infinite runs $p = uS^\omega$ and all the information needed to evaluate the satisfiability of ϕ is contained in u. When $p = u$, analogously to the notion of *bad prefix* defined by Kupferman and Vardi in [62], we say that a finite path is a *good prefix*, i.e., a finite word which can be always extended to an infinite word defined by uS^ω. Intuitively, this means that the prefix u is enough to evaluate correctly the formula and $uS^\omega, 0 \models \phi$ reduces to evaluate $u, 0 \models_k \phi$. The satisfaction relation $\models$ defined for LTL is not defined over finite sequences and the path $p = u$ can not be used to evaluate $u, 0 \models \phi$. Finite runs of M entail a new semantics for LTL, called *bounded semantics* as defined in [61], which is an approximation of the standard LTL semantics defined in Section 2.7.1. Bounded semantics of LTL formulae is given with respect to a "bounded" structure (S, π, L) where $\pi = s_0 \dots s_{k-1}$ is now a finite sequence of states $s_i \in S$. Equivalently, a model for a LTL formula is a finite sequence of states $w \in (2^{AP})^k$.

$$
\begin{aligned}
\pi, i \models_k p &\overset{\text{def}}{\Leftrightarrow} p \in L(s_i) \text{ for } p \in AP \\
\pi, i \models \neg\phi &\overset{\text{def}}{\Leftrightarrow} \pi, i \not\models_k \phi \\
\pi, i \models_k \phi \wedge \psi &\overset{\text{def}}{\Leftrightarrow} \pi, i \models_k \phi \text{ and } \pi, i \models_k \psi \\
\pi, i \models_k \mathbf{X}\phi &\overset{\text{def}}{\Leftrightarrow} \pi, i+1 \models_k \phi \text{ and } i < k \\
\pi, i \models_k \phi \mathbf{U} \psi &\overset{\text{def}}{\Leftrightarrow} \exists\, i \leq j \leq k : \pi, j \models_k \psi \text{ and } \pi, n \models_k \phi \ \ \forall\, i \leq n < j
\end{aligned}
$$

A formula $\phi \in$ LTL is *bounded satisfiable* if there exists a linear time structure (S, π, L) such that $\pi, 0 \models_k \phi$ (in which case (S, π, L), or simply π, is a *model* of ϕ). Equivalently, a LTL formula ϕ is satisfiable if there exists a word $w \in (2^{AP})^k$ such that $w, 0 \models_k \phi$. The definition of "release" temporal operators introduced in Section 2.7.1 and defined as $\neg(\phi \mathbf{R} \psi) \Leftrightarrow \neg\phi \mathbf{U} \neg\psi$ is no longer admissible. In fact, according to the standard semantics for LTL, given a structure (S, π, L) where $\pi = s_0 S^\omega$ is an infinite sequence of states in S, the satisfaction relation for $\phi \mathbf{R} \psi$ is defined as:

$$\pi, i \models \phi \mathbf{R} \psi \stackrel{\text{def}}{\Leftrightarrow} \forall j \geq i \ (\pi, j \models \psi \ \text{ or } \ \exists n > i \ \pi, n \models \phi)$$

When we consider bounded models of finite length, the case for $\phi \mathbf{R} \psi$ where ψ always holds and ϕ is never satisfied has to be excluded. Therefore, the bounded semantics for $\phi \mathbf{R} \psi$ is defined equivalently to the semantics for the until operator:

$$\pi, i \models_k \phi \mathbf{R} \psi \stackrel{\text{def}}{\Leftrightarrow} \exists i \leq j \leq k : \pi, j \models_k \psi \ \text{ and } \ \pi, n \models_k \phi \ \ \forall i \leq n < j.$$

Bounded semantics and standard semantics for LTL can be proved to be equivalent.

Lemma 36 ([61]). *Let ϕ be a LTL formula and (S, π, L) be a Kripke structure where $\pi \in S^+$ a finite sequence of states. Then, $\pi, 0 \models_k \phi \Leftrightarrow \pi, 0 \models \phi$.*

Lemma 37 (Lemma 8, [61]). *Let ϕ be a LTL formula and (S, π, L) be a Kripke structure where $\pi \in S^\omega$. If $\pi, 0 \models \phi$ then there exists $k \in \mathbb{N}$ such that $\pi, 0 \models_k \phi$.*

Finally, we can claim the equivalence between LTL model-checking problem and bounded LTL model-checking problem.

Theorem 38 (Theorem 9, [61]). *Let ϕ be a LTL formula and (S, R, L) be a Kripke structure and $s_0 \in S$. Then, $M, s_0 \models \phi$ if, and only if, there exists $k \in \mathbb{N}$ such that $M, s_0 \models_k \phi$*

Next subsection provides the encoding of bounded semantics of LTL. Given a LTL formula ϕ and a positive integer k, the encoded bounded semantics is a propositional formula $[\phi]_k$ representing infinite ultimately periodic models uv^ω of ϕ such that $|uv| = k$. A detailed analysis of efficient encodings requiring quadratic number of variables in the size of LTL formulae can be found in [63].

2.7.4. Linear Encoding of LTL for SAT

The ultimately periodic semantics of LTL formulae is encoded by a boolean formula representing ultimately periodic models. The proposed encoding is based on the "eventuality encoding" defined in thw work of Biere et al. [63] which is an improved version of the one defined by Biere et al. in the seminal work [61] introducing bounded model checking.

Let ϕ be a LTLB formula, AP be the set of atomic propositions and $[\phi]_k$ be the boolean formula representing models $\pi \in S^\omega$ of ϕ such that $[\phi]_k$ is satisfiable if, and only if, $\pi \models_k \phi$. Let S be the set of states and $s \in S$ be an element an infinite sequence in S^ω as defined in Section 2.7.1.

2.7.5. Encoding periodicity

When formula ϕ has an ultimately periodic models $a(sb)^\omega$, i.e. $a(sb)^\omega \models_k \phi$ where $a, b \in S^*$, the encoding enforce the presence of a repeating state s such that finite models $asbs$ of length $k+1$ can effectively represent infinite models of the form $a(sb)^\omega$. Therefore, the word sb is the loop of the ultimately periodic model. In order to represent loop of ultimately periodic models, the encoding exploits $k+1$ fresh *loop selectors* variables $l_0, \ldots, l_k$ which determine where the model has loop. If l_j then $s_{j-1} = s_k$. When $\neg l_i$ for all $i \in [0, k]$ then the encoding represents finite prefixes in S^{k+1}. Loop constraints non-deterministically select the loop position and guarantee that at most one loop selector l_i is true. Moreover, $k+1$ variables $inLoop_i$ determine which position of the model between 0 and k are part of the loop. The base case defining loop constraint is:

$$\neg \boldsymbol{l}_0 \text{ and } \neg \boldsymbol{inLoop}_0$$

For all $1 \leq i \leq k$, loop constraints define position of the loop and the loop part:

$$\begin{aligned} \boldsymbol{l}_i &\Rightarrow \pi(i-1) = \pi(k) \\ \boldsymbol{inLoop}_i &\Leftrightarrow \boldsymbol{inLoop}_{i-1} \vee \boldsymbol{l}_i \\ \boldsymbol{inLoop}_{i-1} &\Rightarrow \neg \boldsymbol{l}_i \\ \boldsymbol{loopEx} &\Leftrightarrow \boldsymbol{inLoop}_k \end{aligned}$$

Formula $\pi(i-1) = \pi(k)$ enforces equivalence between two position of the model defining the loop; i.e., $s_{i-1} = s_k$.

2.7.6. Encoding the Propositional Terms

Boolean encoding associates to each propositional subformula φ a *predicate* $\boldsymbol{\varphi} \in \{true, false\}$.

As the length of models is fixed to $k+1$, and all models start from 0, formula predicates are subsets of $\{0,\ldots,k+1\}$. The predicate associated with φ (denoted by the same name but written in bold face), is recursively defined for all positions in $\{0,\ldots,k+1\}$ as:

φ	$0 \leq i \leq k+1$
p	$\boldsymbol{\varphi}_i \Leftrightarrow p \in L(s_i)$
$\neg\psi$	$\boldsymbol{\varphi}_i \Leftrightarrow \neg\boldsymbol{\psi}_i$
$\zeta \wedge \psi$	$\boldsymbol{\varphi}_i \Leftrightarrow \boldsymbol{\zeta}_i \wedge \boldsymbol{\psi}_i$

The conjunction of all the constraints for all the subformulae φ of ϕ constitutes the formula $|PropConstraints|_k$.

2.7.7. Encoding Temporal Operators

Temporal subformulae constraints ($|TempConstraints|_k$) define the basic temporal behavior of future and past operators, by using their traditional fixpoint characterizations. Let ζ and ψ be propositional subformulae of ϕ, then:

φ	$0 \leq i \leq k$
$\mathbf{X}\psi$	$\boldsymbol{\varphi}_i \Leftrightarrow \boldsymbol{\psi}_{i+1}$
$\zeta\mathbf{U}\psi$	$\boldsymbol{\varphi}_i \Leftrightarrow \boldsymbol{\psi}_i \vee (\boldsymbol{\zeta}_i \wedge \boldsymbol{\varphi}_{i+1})$
$\zeta\mathbf{R}\psi$	$\boldsymbol{\varphi}_i \Leftrightarrow \boldsymbol{\psi}_i \wedge (\boldsymbol{\zeta}_i \vee \boldsymbol{\varphi}_{i+1})$

To correctly define the semantics of past operators, the initial instant $i=0$ has to be treated separately.

φ	$0 < i \leq k+1$	$i = 0$
$\mathbf{Y}\psi$	$\boldsymbol{\varphi}_i \Leftrightarrow \boldsymbol{\psi}_{i-1}$	$\neg(\boldsymbol{\varphi})_0$
$\mathbf{Z}\psi$	$\boldsymbol{\varphi}_i \Leftrightarrow \boldsymbol{\psi}_{i-1}$	$\boldsymbol{\varphi}_0$
$\zeta\mathbf{S}\psi$	$\boldsymbol{\varphi}_i \Leftrightarrow \boldsymbol{\psi}_i \vee (\boldsymbol{\zeta}_i \wedge \boldsymbol{\varphi}_{i-1})$	$\boldsymbol{\varphi}_0 \Leftrightarrow \boldsymbol{\psi}_0$
$\zeta\mathbf{T}\psi$	$\boldsymbol{\varphi}_i \Leftrightarrow \boldsymbol{\psi}_i \wedge (\boldsymbol{\zeta}_i \vee \boldsymbol{\varphi}_{i-1})$	$\boldsymbol{\varphi}_0 \Leftrightarrow \boldsymbol{\psi}_0$

Last state constraints ($|LastStateConstraints|_k$) define an equivalence between (sub)formulae in $k+1$ and (sub)formulae at position of loop l_i, because the instant $k+1$ is representative of the instant defined by l_i along periodic paths. Otherwise, truth values in $k+1$ are trivially false.

$$loopExists \Leftarrow \boldsymbol{\zeta}_{k+1}$$
$$l_i \Rightarrow (\boldsymbol{\zeta}_{k+1} \Leftrightarrow \boldsymbol{\zeta}_i)$$

for all subformulae ζ of ϕ.

Let us observe that if a loop does not exists then the fixpoint semantics of $\mathbf{R}$ is exactly the bounded semantics defined over finite acyclic path

in Section 2.7.3. Finally, to correctly define the semantic of **U** and **R**, their *eventuality* have to be enforced. Briefly, if $\zeta \mathbf{U} \psi$ holds at i, then ψ eventually holds in $j \geq i$. When $\zeta \mathbf{R} \psi$ does not hold at i, then ψ must eventually does not hold in $j \geq i$, i.e., no formula $\zeta \mathbf{R} \psi$ holds almost always while ψ remains false. However, according to the temporal encoding of until and release, this may happen. Formula $\varphi = \zeta \mathbf{U} \psi$ may hold even if ψ is never satisfied: in fact, $\boldsymbol{\varphi}_i \Leftrightarrow (\boldsymbol{\zeta}_i \wedge \boldsymbol{\varphi}_{i+1})$ for all $i \in [0, k]$. Also, formula $\varphi = \zeta \mathbf{R} \psi$ may not hold even if $\neg\psi$ is never satisfied since $\boldsymbol{\varphi}_i \Leftrightarrow (\boldsymbol{\zeta}_i \vee \boldsymbol{\varphi}_{i+1})$ for all $i \in [0, k]$. Formula ψ holds at all indices of loop but $\zeta \mathbf{R} \psi$ does not hold at any position. Along finite models, eventualities must hold between 0 and k. If a loop exists, an eventuality may holds within the loop. Boolean encoding introduces k propositional variables for each $\zeta \mathbf{U} \psi$ and $\zeta \mathbf{R} \psi$ subformula of ϕ, for all $1 \leq i \leq k$, which represent the eventuality of ψ implicit in the formula. To enforce assignments described above such that eventualities of until and release always occur, we introduce new auxiliary formulae $\langle \mathbf{F} \psi \rangle$ and $\langle \mathbf{G} \psi \rangle$ for each position $i \in [0, k]$. When $\langle \mathbf{F} \psi \rangle$ holds at position i, then all formulae in $[i, k]$ requiring eventuality for ψ may be satisfied because ψ occurs inside the loop. When $\langle \mathbf{G} \psi \rangle$ does not hold at position i, then all formulae in $[j, k]$, such that l_j holds, requiring eventuality for ψ can not be satisfied because $\neg\psi$ occurs inside the loop. When loop can be realized, i.e., *loopExists* holds, then:

- if $\zeta \mathbf{U} \psi$ holds at position k then we have to force the eventuality for ψ by imposing that $\langle \mathbf{F} \psi \rangle$ holds at k.

- if $\neg(\zeta \mathbf{R} \psi)$ holds at position k then we have to force the eventuality for $\neg\psi$ by imposing that $\langle \mathbf{G} \psi \rangle$ does not hold at k.

φ	
$\zeta \mathbf{U} \psi$	$loopExists \Rightarrow (\boldsymbol{\varphi}_k \Rightarrow \langle \mathbf{F} \psi \rangle_k)$
$\zeta \mathbf{R} \psi$	$loopExists \Rightarrow (\boldsymbol{\varphi}_k \Leftarrow \langle \mathbf{G} \psi \rangle_k)$

Semantics for $\langle \mathbf{F} \psi \rangle$ and for $\langle \mathbf{G} \psi \rangle$ are defined recursively along the finite model. Let us consider the auxiliary formula $\langle \mathbf{F} \psi \rangle_i$. If ψ holds at position i inside the loop, then the eventuality of ψ occurs and $\langle \mathbf{F} \psi \rangle_i$ holds. When $\langle \mathbf{F} \psi \rangle_i$ holds, then from i to k eventuality of ψ is enforced, recursively. Since loop index can not be at 0, i.e., $\neg l_0$, then formula $\langle \mathbf{F} \psi \rangle_0$ is always false and ψ must occur in $[1, k]$. When $\neg\psi$ holds at position i inside the loop, then formula $\langle \mathbf{G} \psi \rangle_i$ is falsified. Analogously with **U**, if $\langle \mathbf{G} \psi \rangle_i$ does not hold, then from i to k eventuality of $\neg\psi$ is enforced. Since loop index can not be at 0, i.e., $\neg l_0$, then formula $\langle \mathbf{G} \psi \rangle_0$ is always true and

$\neg\psi$ must occur in $[1, k]$.

φ	$1 \leq i \leq k$	
$\zeta \mathbf{U} \psi$	$\langle \mathbf{F}\psi \rangle_i \Leftrightarrow \langle \mathbf{F}\psi \rangle_{i-1} \vee (inLoop_i \wedge \boldsymbol{\psi}_i)$	$\neg\langle \mathbf{F}\psi \rangle_0$
$\zeta \mathbf{R} \psi$	$\langle \mathbf{G}\psi \rangle_i \Leftrightarrow \langle \mathbf{G}\psi \rangle_{i-1} \wedge (\neg inLoop_i \vee \boldsymbol{\psi}_i)$	$\langle \mathbf{G}\psi \rangle_0$

The conjunction of all the constraints for all the subformulae φ of ϕ constitutes the formula $|Eventually|_k$.

The complete encoding $[\phi]_k$ consists of the logical conjunction of all above components, together with ϕ evaluated at the first instant along the time structure.

Complexity analysis

Let ϕ be the PLTLB formula and $|\phi|$ be its dimension. If $m = \mathcal{O}(|\phi|)$ is the total number of subformulae and n is the total number of temporal operators $\mathbf{U}$ and $\mathbf{R}$ occurring in ϕ, then the boolean encoding requires $(2k+3)+(k+2)m+(k+1)n = O(k(m+n))$ fresh propositional variables.

Given an LTL formula ϕ a Kripke structure $M = (S, R, L)$ and a bound $k \geq 0$ bounded model-checking is performed by solving the propositional formula:

$$\boldsymbol{s}_0 \wedge \bigwedge_{i=0}^{k-1} R(\boldsymbol{s}_i, \boldsymbol{s}_{i+1}) \wedge [\phi]_k$$

where $\boldsymbol{s}_i$ are propositional variables representing states belonging to S. The formula $\bigwedge_{i=0}^{k} R(\boldsymbol{s}_i, \boldsymbol{s}_{i+1})$ represents finite paths of M of length k and $[\phi]_k$ is the encoding of ultimately periodic runs of $\mathcal{A}_\phi$ recognizing models for ϕ (see Section 2.7.4 for details).

Clarke et al. [64] show an alternative method to perform bounded model-checking. It exploits the Vardi-Wolper LTL model checking method testing the emptiness of the product $M \times \mathcal{A}_\phi$ of the given Kripke structure M and the Büchi automaton $\mathcal{A}_\phi$ (in this case, $\neg\phi$ is the property to be verified). Nonemptiness for a Büchi automaton can be shown by exhibiting a path from an initial state which defines a fair loop, i.e., a loop involving a final control state. LTL model-checking is then reduced to finding fair loops of $M \times \mathcal{A}_\phi$ whose length is bounded by the size of the automaton $M \times \mathcal{A}_\phi$. The translation adopted is inspired by the work of deMoura et al. [65] who suggested an analogous translation in the context of bounded model-checking of infinite state systems. Let F be the set of final states of the product automaton $M \times \mathcal{A}_\phi$; finding paths of length k with fair loops on states of F is reduced to satisfiability of the following formula. Let $M \times \mathcal{A}_\phi$ be a tuple (Q, δ, Q_0, F) where Q is the

set of control states δ be the transition relation Q_0 be the set of initial states and F be the set of accepting states:

$$\boldsymbol{q}_0 \wedge \bigwedge_{i=0}^{k-1} \delta(\boldsymbol{q}_i, \boldsymbol{q}_{i+1}) \wedge \bigwedge_{l=0}^{k-1} \left(\boldsymbol{q}_l = \boldsymbol{q}_k \wedge \bigvee_{j=l}^{k} \bigvee_{q \in F} \boldsymbol{q}_j \right) \tag{2.1}$$

where $\boldsymbol{q}_i$ represents instances of boolean variables corresponding to control state $q \in Q$ at position i along the path $q_0, \ldots, q_{k-1}$. Formula 2.1 is satisfiable if there exists a path of M of length k satisfying the formula ϕ. The construction of $\mathcal{A}_\phi$, which is known to be PSPACE in the size of the formulae ϕ, represents the major limitation of this approach.

Completeness for bounded LTL model-checking

A typical application of bounded model-checking starts at $k = 1$ and increments the value for k until $M, s_0 \models_k \phi$. This represents a partial decision procedure since termination is not guaranteed. In fact, when $M, s_0 \not\models \phi$, there does not exist a value $k \in \mathbb{N}$ such that $M, s_0 \models_k \phi$. Still, the existence of a finite completeness threshold on length of runs satisfying a LTL formula makes the procedure complete since it bounds the number of tests $M, s_0 \models_k \phi$ which shall be performed. Given a Kripke structure M and a LTL formula ϕ, if $M, s_0 \not\models_k \phi$ does not hold for all between 0 and the completeness threshold then there does not exist any finite sequences $p = uv$ or $p = u$ such that $uv^\omega \models \phi$ or $uS^\omega \models \phi$. In this case, we claim that $M, s_0 \not\models \phi$ and we conclude that ϕ does not hold in M.

We briefly recall some main results on completeness for LTL model-checking. The first result can be found in the work of Biere et al. [61] which provides a fundamental results about completeness for full LTL formulae.

Theorem 39 (Theorem 25, [61]). *Given an LTL formula ϕ and a Kripke structure $M = (S, R, L)$. Then, $M \models \phi$ if, and only if, there exists k in $|S| \cdot 2^{O(|\phi|)}$ such that $M \models_k \phi$.*

The upper bound is obtained from the dimension of automaton derived by synchronizing the Kripke structure M and Büchi automaton $\mathcal{A}_\phi$ which is of exponential size in the dimension of the formula ϕ. It is worth noticing that given a set of n boolean variables, the number of different states, defining the set S of a Kripke structure $M = (S, R, L)$, is bounded by 2^n; i.e., $|S| \leq 2^n$. From Theorem 39, the value for the completeness threshold is still exponential in the number of variables and in the dimension of the LTL formula. Although the bound is often too

large for practical model-checking problems, Theorem 39 establishes the completeness property of the procedure when full LTL is considered.

Definition 40. *Let $M = (S, R, L)$ be a Kripke structure. Let $\boldsymbol{s}_i, \boldsymbol{q}_i$ be propositional variables representing states of S at instant i. The* reachability diameter $d(M)$ *is the minimal d satisfying:*

$$\forall \boldsymbol{s}_0 \ldots \boldsymbol{s}_{d+1}\ \exists \boldsymbol{q}_0 \ldots \boldsymbol{q}_d$$
$$\bigwedge_{i=0}^{d} R(\boldsymbol{s}_i, \boldsymbol{s}_{i+1}) \Rightarrow (\boldsymbol{q}_0 = \boldsymbol{s}_0 \wedge \bigwedge_{i=0}^{d-1} R(\boldsymbol{q}_i, \boldsymbol{q}_{i+1}) \wedge \bigvee_{i=0}^{d-1} \boldsymbol{q}_i = \boldsymbol{s}_{d+1}).$$

Intuitively, the value d is the longest shortest path starting from any initial state s_0 to any reachable state. The diameter problem can be reduced to the "all pair shortest path" problem and therefore be solved in time polynomial in the size of the graph representing M. The result of Theorem 39 can be refined when formulae defining invariant properties $\mathbf{G}p$ are considered, where $p \in AP$. However, solving quantified boolean formulae is hard, since the problem is known to be NP-complete. Hardness of computation can be eased by removing alternation of quantifiers in the formula provided we admit an approximation of the value $d(M)$.

Definition 41. *Let $M = (S, R, L)$ be a Kripke structure. Let $\boldsymbol{s}_i, \boldsymbol{q}_i$ be propositional variables representing states of S at instant i. The* recurrence diameter $r(M)$ *is the minimal r satisfying:*

$$\forall \boldsymbol{s}_0 \ldots \boldsymbol{s}_{r+1} \in S\ \bigwedge_{i=0}^{d} R(\boldsymbol{s}_i, \boldsymbol{s}_{i+1}) \Rightarrow (\boldsymbol{q}_0 = \boldsymbol{s}_0 \wedge \bigwedge_{i=0}^{r-1} R(\boldsymbol{q}_i, \boldsymbol{q}_{i+1})).$$

Recurrence diameter $r(M)$ is the longest loop-free path between two states. Since every shortest-path is a loop-free path, then $d(M) \leq r(M)$. The recurrence diameter can be computed by solving a series of SAT instances, rather than QBF instances which computation is typically harder. The recurrence diameter is important because $r(M)$ characterizes the completeness threshold for $\mathbf{F}p$ properties. Both definitions can be refined by specifying explicitly the initial states: $d(M, I)$ and $r(M, I)$ denote reachability diameter for M when paths are required to start from an initial states belonging to $I \subseteq S$.

For reachability formulae of the form $\mathbf{F}p$, where $p \in AP$ is an atomic proposition, Biere et al. proved:

Theorem 42 (Theorem 23, [61])**.** *Given a LTL formula $\phi = \mathbf{F}p$, a Kripke structure $M = (S, R, L)$ and $s_0 \in S$, $M, s_0 \models \phi$ if, and only if, there exists $k \leq d(M, I)$ such that $M, s_0 \models_k \phi$.*

When invariant constraints of the form $\mathbf{G}p$ are considered, Kroening and Strichman refine the result of Biere et al. in [61] and show that the completeness bound is $r(M, I)$:

Theorem 43 ([66]). *Given a LTL formula $\phi = \mathbf{G}p$, a Kripke structure $M = (S, R, L)$ and $s_0 \in S$, $M, s_0 \models \phi$ if, and only if, there exists $k \leq r(M, I)$ such that $M, s_0 \models_k \phi$.*

Reachability diameter is sufficient for $\mathbf{G}p$ formulas. In fact, a counterexample to $\mathbf{G}p$ is a path of M leading to a state which contradicts p. Since all states can be reached through paths of length $r(M)$ or less, checking paths whose length is bounded by $r(M)$ is sufficient for finding all reachable states that contradict p. Reachability diameter is not sufficient for finding all counterexamples to $\mathbf{F}p$ formulas. A counterexample for such a formula is a path ending in a back-loop, where all the states on the path satisfy $\neg p$. However, in the Kripke structure may exists a unique path of length greater that $r(M)$ which is the only counterexample to $\mathbf{F}p$. Since the bound is $r(M)$ it can not be discovered.

When Formula (2.1) is adopted to solve BMC, Clarke et al. showed that the completeness threshold is derived by the product automaton $M \times \mathcal{A}_\phi$.

Theorem 44 (Theorem 1, [64]). *Given a LTL formula ϕ, a Kripke structure $M = (S, R, L)$ and $s_0 \in S$. Let A be the product $M \times \mathcal{A}_\phi$. $M, s_0 \models \phi$ if, and only if, there exists $k \leq min(r(A, Q_0) + 1, d(A, Q_0) + d(A))$ such that $M, s_0 \models_k \phi$.*

The reduction of BMC problem to SAT proposed by Clarke et al. in [64] uses an explicit representation of Büchi automata which can be defined by an exponential number of control states in the dimension of the LTL formula. Moreover, the product automaton is defined when $\mathcal{A}_\phi$ is a non-generalized Büchi automaton which is derived from the Vardi-Wolper automaton representing the LTL formulae. Since the conversion of generalized Büchi automaton to a non-generalized one expands the size of the automaton by a factor related to the number of fair sets, the length of path satisfying the LTL formula ϕ is not minimal.

Completeness for full LTL is considered also in [63], where authors adapt the ideas of Sheeran et al. [67] to incremental bounded model-checking. Sheeran et al. proposed a SAT-based method [67] to check safety properties in finite-state systems which is later refined and extended to infinite-state systems by de Moura et al. in [68]. Techniques based on *induction* can be used to to prove invariant property on systems, i.e., a property which hold in all reachable states of the system.

In general, proving an invariant property typically involves finding an *inductive invariant*. In this context, we slightly simplify the satisfaction relation $\models$ of LTL formulae since we will deal only with formulae which do not involve temporal operators.

$$\begin{aligned} s[p] &\overset{\text{def}}{\Leftrightarrow} p \in L(s) \text{ for } p \in AP \\ s[\neg\phi] &\overset{\text{def}}{\Leftrightarrow} \neg s[\phi] \\ s[\phi \wedge \psi] &\overset{\text{def}}{\Leftrightarrow} s[\phi] \text{ and } s[\psi] \end{aligned}$$

A property υ is an inductive invariant for a property ϕ when it satisfies some suitable requirements. First, υ does not involve temporal operators; therefore, it is only defined by means of atomic proposition $p \in AP$ and boolean connectives. Other requirements are the following:

1. υ strengthens ϕ, i.e., υ implies ϕ in all states:

$$\forall s \in S \; s[\upsilon] \Rightarrow s[\phi]$$

2. υ is invariant over all paths of length n, i.e., the following formula is unsatisfiable:

$$\exists \boldsymbol{s}_0 \ldots \boldsymbol{s}_n \; \boldsymbol{s}_0 \wedge \bigwedge_{i=0}^{n-1} R(\boldsymbol{s}_i \boldsymbol{s}_{i+1}) \wedge \bigvee_{i=0}^{n} \neg \boldsymbol{s}_i[\upsilon]$$

where each $\boldsymbol{s}_i$ is a propositional variable representing $s \in S$ occurring in the sequence $s_0, \ldots s_n$ at position i. Informally, the unsatisfiability of this formula guarantees that for all paths of length n the invariant υ holds.

3. υ is inductive, i.e., validity is preserved under each transition of the system. The property υ is inductive when the following formula is unsatisfiable:

$$\exists \boldsymbol{s}_0 \ldots \boldsymbol{s}_{n+1}$$
$$loopfree(\boldsymbol{s}_0, \ldots, \boldsymbol{s}_{n+1}) \; \wedge \; \bigwedge_{i=0}^{n} (\boldsymbol{s}_i[\upsilon] \; \wedge \; R(\boldsymbol{s}_i \boldsymbol{s}_{i+1})) \wedge \neg \boldsymbol{s}_{n+1}[\upsilon]$$

where each $\boldsymbol{s}_i$ is a propositional variable representing $s \in S$ occurring in the sequence $s_0, \ldots s_n$ at position i. The unsatisfiability of this formula guarantees that for all loop-free paths of length $n+1$ the invariant υ can not be falsified.

Completeness of the induction rule is proved by Sheeran in [67] and results by imposing that sequences $s_0 \dots s_n$ satisfying formulae are loop-free. In fact, let (S, R, L) be a Kripke structure satisfying the invariant ϕ, i.e., $\forall s \in S\ s[\phi]$. In this case, there are no finite paths leading to a state violating the invariant from an initial one; i.e., there are no loop-free paths reaching a state $s \in S$ such that $s[\neg\phi]$. Otherwise, the system is not ϕ invariant and there exists a finite loop-free path starting from an initial state which ends in $s[\neg\phi]$. By induction, every loop-free path (of any length) starting from an initial state contains only states satisfying the property ϕ. In practice, steps **1.** and **2.** are iterated from $n = 0$ until the value n reaches the length of the longest loop-free path starting from an initial state or the length of the longest loop-free path consisting in all states satisfying ϕ followed by a state in which ϕ does not hold (**2.** falsified). In the first case, if both the formulae are unsatisfiable then the property ϕ holds; otherwise, ϕ is not an invariant for the system. In this second case, if the first state s_0 is an initial state then **1.** will be unsatisfiable at next step for path of $n + 1$. The induction rule can be refined in order to reduce the length of paths between pairs of states. In fact, the longest loop-free path between two states can be longer than the shortest path between them which is sufficient to detect a violation if one of the two states does not satisfy the invariant, i.e., $s[\neg\phi]$. Still, soundness and completeness of induction schema is guaranteed [67].

de Moura et al. refine in [68] the previous definition of induction rule and extend it to infinite-state systems. Induction rule is parameterized with respect to a suitable notion of simulation $\preceq$ on states. It is tailored to infinite-state systems for which equality relation between states is no longer effective as for finite-state systems. In fact, simulation relation over states can be used to restrict the set of paths of transition systems on which properties are verified. As in the case of finite-state induction, runs which do not contain "similar" states, called *compressed path*, are considered when invariant properties are verified. Compressed path are used to prove the invariance of a property by induction with respect to $\preceq$. Completeness of the induction schema results from definition of suitable simulation $\preceq$. For instance, induction for infinite-state system is not complete when the simulation relation is the equality. In order to handle finite runs of length n, authors show that the simulation relation which they adopt (*direct/inverse simulation*) is closed with respect to transitive closure $\preceq^n$ for $n \in \mathbb{N}$. Therefore, when $\preceq$ is a direct/inverse simulation, they are allowed to consider the finite transitive closure $\preceq^n$ to abstract computation of the systems when they prove invariant over finite runs of length n. Moreover, whenever formula **2.** of the induction schema does not hold, the sequence $s_0 \dots s_{n+1}$ is a counterexample such

that the first n states satisfy ϕ and the last state does not. If the first state is reachable, then ϕ is refuted. Otherwise, the sequence is labeled *spurious* and it is used to strengthen the invariant ϕ.

3. Bounded Satisfiability Problem

The family of linear-time logics, such as LTL, are quite attractive since they have a natural connection to automata on infinite words. Decidability of most of the model-checking algorithms for linear-time logics is proved by the automata-based approach proposed by Vardi and Wolper [51]. Formulae are translated into Büchi automata and Model-checking problem is then reduced to emptiness problem. Although satisfiability and LTL model-checking problem for finite-state systems are decidable and their complexity is known to be PSPACE-complete, even the most basic properties such as safety properties, in the general case, become undecidable over infinite-state systems. Therefore, various symbolic representations regaining decidability have been proposed to deal with model-checking problem over infinite-state system. Examples of formalisms admitting decidable LTL model-checking are pushdown automata, Petri Nets (equivalently VASS), Process Algebra and Parallel Process [69]:

Table 3.1.: Decidability results of model-checking over infinite models.

Finite-state automata	PSPACE-complete
Basic Process Algebra	EXPTIME-complete
Pushdown automata	EXPTIME-complete
Basic Parallel Process	EXPSPACE-complete
Petri Nets	EXPSPACE-complete

Timed automata and some classes of counter systems, like reversal-bounded counter systems, presented in Section 2.6, are other examples.

Propositional variables involved in LTL language can represent properties on current configurations of the systems. In particular, when counter systems are considered, propositional atoms can abstract Presburger formulae like, for instance, $p := x > y + 1$. Extending LTL with Presburger constraints allow us to specify quantitative properties over counter systems which go beyond reachability. Rather than defining a syntactic replacement of propositional variables, Presburger formulae can be included in the logical language. If the language admits quantifiers, values

of variables at distinct positions over time can be compared: for instance, let x be a variable representing a counter of the system and $\mathtt{y}$ be a fresh variable of the logic language. A formula like $\exists \mathtt{y} x > \mathtt{y} \wedge \mathbf{XXX}(\mathtt{y} = x)$ defines a comparison between the value of the counter x three positions away in the future. A first attempt to define a temporal logic with Presburger constraints is proposed by Bouajjani et al. in [70]. Constrained LTL is an extension of LTL which is able to define constraints on pattern of computations (order of appearance of states) or constraining the number of occurrences of states in computations. Constraints on pattern are defined by means of automata whereas constraints on numbers of occurrences are expressed by Presburger formulae. If AP is the set of propositional atoms, Constrained LTL formulae are interpreted over infinite sequences $(2^{AP})^{\omega}$. Although different versions of LTL with Presburger formuale can be defined, see Comon and Cortier [5] for instance, we will adopt fragment of Presburger LTL defined in [11]. In the original definition, Presburger LTL is dedicated to reason about computations of counter systems. Therefore, models of formulae are infinite sequences of configuration of counter systems of the form $(Q \times D^n)^{\omega}$ where Q is the set of control states, D is the domain defining values for counters and n is the dimension of the counter system. Formulae of Presburger LTL define a fragment of a more general extension of LTL where atoms belong to first-order language. First-order LTL, denoted LTL(**FO**), is the extension of LTL where, in addition to atomic propositions, boolean connectives and temporal modalities, there are also predicates, functions, constants and individual variables, each interpreted over an appropriate domain. Following definition are taken from [45].

Let $(\mathcal{F}, \mathcal{R})$ be the signature of the language defining symbols functions and symbols predicate. A *term* (or *Arithmetic Temporal Term*, a.t.t.) of LTL(**FO**) is defined by the grammar:

$$\tau := c \mid x \mid f(\tau_1, \ldots, \tau_n) \mid \mathrm{X}\tau$$

where $c \in D$, $x \in \mathcal{F}$ is a 0-ary function and $f \in \mathcal{F}$ is a n-ary function symbol and X is a temporal modalities over terms. Equality $=$ between terms is understood. Let V be the set of 0-ary functions which defines the set of variables occurring in a formula. The *depth* $|\tau|$ of an a.t.t. is the total amount of temporal shift needed in evaluating τ. The depth $|\tau|$ of the term τ is recursively defined as:

$$|x| = 0, |\tau| = |\tau| + 1$$

and $|f(\tau_1, \ldots, \tau_n)| = max\{|\tau_1|, \ldots, |\tau_n|\}$. *Atomic formulae*, or *atoms*, of LTL(**FO**) are atomic propositions or predicates over terms according to

the following grammar:

$$\alpha := p \mid R(\tau_1, \ldots, \tau_n)$$

where $p \in \mathcal{R}$ is a 0-ary predicate symbol and $R \in \mathcal{R}$ is a n-ary predicate symbol. We denote the set of atomic proposition AP to be the set of 0-ary predicate symbol. *Formulae* of LTL(**FO**) are defined by the following grammar:

$$\phi := \alpha \mid \neg\phi \mid \phi \wedge \phi \mid \mathbf{X}\phi \mid \phi\mathbf{U}\phi \mid \exists y\ \phi$$

where $y \in V'$ is a free variable occurring in ϕ, and $\mathbf{X}$ and $\mathbf{U}$ are classical LTL temporal modalities for "next"-time and "until".

To correctly define semantics of quantified formulae we will assume that the set of atomic propositions and first-order variables are divided into two subset: $AP = AP_l \cup AP_g$ is the set of atomic propositions and $V = V_l \cup V_g$ is the set of variables. This distinction is useful when we want to distinguish variables in V_l taking values from the underlying linear-time structure and auxiliary variable V_g from the language (quantified variables), as adopted, for instance, by Thomas in [71] and Demri et al. in [72]. We call V_l the set of *local variable* and V_g the set of *global variables*. This choice is motivated by the need of interpreting local variables with the values of counters on configurations when the language is used in model-checking of counter systems. On the other hand, global variables can be seen as parameters of the systems or quantitative values whose values are determined within the context encompassing the system. Therefore, in this case, we are interested in dealing with LTL(**FO**) formulae where all occurrence of each variable in V_g is in the scope of some quantifiers, i.e., there are no free variables belonging to V_g; whereas all local variables are free.

When all function symbols in $\mathcal{F}$ and predicates symbols in $\mathcal{R}$ have the same interpretation over all states of S, interpretation of symbols of LTL(**FO**) language is inherited by the underlying interpretation $\mathcal{I}$ of symbols of the first-order language **FO**. Semantics of LTL(**FO**) is defined with respect to an interpretation (model) $(\mathcal{M}, \sigma)$ for each position of time, where $\mathcal{M}$ is a structure $\mathcal{M} = (D, \mathcal{I})$, and an environment $\varepsilon : V_g \to D$ which defines the value of free global variables belonging to V_g. The structure (S, π, L) determines the interpretation of every local object, i.e., free local variables of V_l and local atomic propositions.

- S be a set of states;
- $\pi \in S^\omega$ is an infinite sequence of states.
- $L : S \to (\mathcal{M}, \sigma)$.

Variables of V_l, which occurs free, as well as values for terms are defined from σ with respect to the structure $\mathcal{M} = (D, \mathcal{I})$. In particular, we have: $\sigma = (\sigma', \sigma'')$ such that:

- $\sigma' : V_l \to D$ is a function associating a value for each free local variable;
- $\sigma'' : AP_l \to \{true, false\}$ is a function associating a value for each free local atomic proposition.

When the set of atomic proposition is empty, we will use simply σ instead of σ'. If non-logical symbols are interpreted locally, the definition of elements of $\mathcal{F}$ and $\mathcal{R}$ may vary at different position. Therefore, the interpretation $\mathcal{I}$ at position s_i may differ with respect to others. Global semantics is realized when $\mathcal{I}_i = \mathcal{I}$ for all $i \geq 0$.

Value of terms is defined by the function $[]_{(\pi,i)}$ with respect to the first-order structure $(\mathcal{M}, \sigma)$ at each position $\pi(i) = s_i$, such that $L(s_i) = (\mathcal{M}, \sigma)$, and the environment ε for global variables:

$$
\begin{aligned}
[x]_{(\pi,i)} &= \begin{cases} \sigma'(x) \text{ for } x \in V_l \\ \varepsilon(x) \text{ for } x \in V_g \end{cases} \\
[\mathrm{X}\tau]_{(\pi,i)} &= [\tau]_{(\pi,i+1)} \\
[f(\tau_1, \ldots \tau_n)]_{(\pi,i)} &= f_{\mathcal{I}_i}\left([\tau_1]_{(\pi,i)}, \ldots, [\tau_n]_{(\pi,i)}\right)
\end{aligned}
$$

where $f_{\mathcal{I}_i}$ is the n-ary function f with respect to the interpretation $\mathcal{I}_i$. The satisfaction relation $\models_\varepsilon$, extending the semantic relation of LTL, is defined recursively with respect to the assignment ε of global elements:

$$
\begin{aligned}
\pi, i \models_\varepsilon p &\overset{\text{def}}{\Leftrightarrow} \sigma''(p) \text{ for } p \in AP \\
\pi, i \models_\varepsilon R(\tau_1, \ldots, \tau_n) &\overset{\text{def}}{\Leftrightarrow} ([\tau_1]_{(\pi,i)}, \ldots, [\tau_n]_{(\pi,i)}) \in R_{\mathcal{I}_i} \\
\pi, i \models_\varepsilon \neg\phi &\overset{\text{def}}{\Leftrightarrow} \pi, i \not\models_\varepsilon \phi \\
\pi, i \models_\varepsilon \phi \wedge \psi &\overset{\text{def}}{\Leftrightarrow} \pi, i \models_\varepsilon \phi \,\text{and}\, \pi, i \models_\varepsilon \psi \\
\pi, i \models_\varepsilon \mathbf{X}\phi &\overset{\text{def}}{\Leftrightarrow} \pi, i+1 \models_\varepsilon \phi \\
\pi, i \models_\varepsilon \phi \mathbf{U} \psi &\overset{\text{def}}{\Leftrightarrow} \exists j \geq i \; \pi, j \models_\varepsilon \psi \text{ and } \forall i \leq n < j \; \pi, n \models_\varepsilon \phi \\
\pi, i \models_\varepsilon \exists y \phi &\overset{\text{def}}{\Leftrightarrow} \text{there is } m \in D \; \pi, i \models_{\varepsilon(y)=m} \phi \text{ such that } y \in V_g \text{ is free in } \phi.
\end{aligned}
$$

It is worth noticing that, whenever the set of local variables is empty $V_l = \emptyset$, the language is the similar to the language of Hodkinson et al. in [73], but enriched with temporal modality X over terms. A formula $\phi \in$ LTL(**FO**) is *satisfiable* with respect to a linear time structure (S, π, L) if there exists an environment ε such that $\pi, 0 \models_\varepsilon \phi$. In this case, we say

that (S, π, L), or simply π, is a model for ϕ. Equivalently, a model for a $\phi \in \text{LTL}(\mathbf{FO})$ formula is an infinite sequence of propositional atoms and valuations of local variables $w \in (2^{AP} \times D^{|V_l|})^\omega$ and an environment ε for quantified global variables. When no local variable occurs, as in the case of the language adopted by Hodkinson et al. in [73], then, $\phi \in \text{LTL}(\mathbf{FO})$ is satisfiable if there exists a word $w \in (2^{AP})^\omega$, such that $w, 0 \models \phi$, and an environment ε. When a linear time structure is not provided, i.e., we are not solving a model-checking problem, we say that a formula $\phi \in \text{LTL}(\mathbf{FO})$ is *satisfiable* if there exists a linear time structure (S, π, L) such that ϕ is satisfiable with respect to (S, π, L).

Past-time temporal modalities $\mathbf{Y}$ and $\mathbf{S}$, with the same semantics of LTL, can be added to the language LTL($\mathbf{FO}$) defined so far.

Fragments of LTL($\mathbf{FO}$)

The first fragment we consider is the *existential* fragment of LTL($\mathbf{FO}$) which is the set of LTL($\mathbf{FO}$) formulae with no occurrences of subformulae of the form $\neg\exists x \neg\phi$. No restriction is required at the level of atomic formulae α when the underlying first-order language admits quantifiers elimination procedure. In this case, atomic formulae $\alpha \in L$ can always be replaced be equivalent formulae $\alpha' \in L$ without quantifiers.

Presburger LTL (LTL(PA $\cup$ $\mathbf{AP}$)) is the fragment of LTL($\mathbf{FO}$) where the signature of the first-order language is $\mathcal{F} = \{+\}$ where $+$ is the usual sum operation and $\mathcal{R} = \{<, =\}$ (or, simply, $\mathcal{R} = \{<\}$ if the equality is understood) with $D = \{\mathbb{N}, \mathbb{Z}\}$. The temporal modality X is usually represented by using quantifiers and auxiliary global variables. All constants, function and predicate symbols $(<, =, +)$ have global interpretation, i.e., they do not vary over time. Models of formulae are structure (S, π, L) such that $L : S \to \sigma$ where structure $\mathcal{M}$ is understood. Therefore, they can be represented as infinite sequences of valuation of local variables and atomic propositions $(2^{AP} \times D^{|V_l|})^\omega$. When the context is clear, we will represent models of LTL(PA $\cup$ $\mathbf{AP}$) as a pair $\sigma = (\sigma', \sigma'')$ such that $\sigma' : S \times V \to D$ maps the set of variables to elements of the domain and $\sigma'' : S \to 2^{AP}$. Valuation of terms is adapted to sequences σ':

$$[x]_{(\sigma,i)} = \begin{cases} \sigma'(i, x) \text{ for } x \in V_l \\ \varepsilon(x) \text{ for } x \in V_g \end{cases}$$

Satisfaction relation $\models_\varepsilon$ naturally extends to σ.

CLTL(L) is the quantifier-free fragment of LTL($\mathbf{FO}$) where arithmetic modality X is applied (possibly recursively) only on atomic variables $x \in V$ and non-logical symbols are globally interpreted according to the language L. Since quantifiers are no longer included in the language,

satisfaction relation $\models_\varepsilon$ is always defined with respect to an empty environment $\varepsilon = \emptyset$. For this reason, we write simply $\models$ instead of $\models_\varepsilon$. Depending on which fragments L of Presburger arithmetic is considered, CLTL(L) represents the restriction of CLTL when atomic formulae α belong to L. Languages considered are quantifier-free Presburger arithmetic, Difference Logic and Integer periodic constraints as they are defined in Section 2.2. Also, L can include propositional language **AP**; in this case, terms α can be also propositional atoms belonging to the set AP (as already defined for LTL(**FO**)). We write $\text{CLTL}^{\text{b}}_{\text{a}}$ to denote the class of CLTL formulae such that the cardinal of V is a and depth of terms is bounded by b. Symbol ω denotes an unbound number of variables or terms.

3.1. Satisfiability problem for CLTL and CLTL Model Checking

Analogously to LTL, the *satisfiability* problem for CLTL formula ϕ consists in determining whether there exists a model σ for ϕ which is an assignment of values to all of its atomic elements satisfying the formula. The satisfiability and model-checking problem for structure $(D, <, =)$ with $D \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$ are analyzed in depth in [6], for IPC* in [74] and [75] and for DL (or DL$^+$) in [74]. Decidability of the satisfiability problem for the above cases is shown by means of automata-based approach similar to the standard case for LTL. Given a CLTL formula ϕ, it is possible to define an automaton $\mathcal{A}_\phi$ such that ϕ is satisfiable if, and only if, $\mathscr{L}(\mathcal{A}_\phi)$ is not empty. Since the emptiness of $\mathscr{L}(\mathcal{A}_\phi)$ in the considered structures is decidable with PSPACE upper bound (polynomial space in the dimension of ϕ), then the satisfiability problem is also decidable with the same complexity.

Satisfiability and model-checking problem are defined hereafter.

SATISFIABILITY OF CLTL(L) FORMULA:

Input	a CLTL formula ϕ
Problem	does there exist a model such that $w \in (2^{AP} \times D^n)^\omega$ such that $w, 0 \models \phi$?

As for LTL model-checking problem, CLTL(L) model-checking problem can be defined with respect to different formalisms defining models for formulae. Demri and Gascon in [74] deal with CLTL model-checking for class of counter systems in CS(DL) and CS(QFP) counter systems and CLTL formulae are restricted to CLTL(DL) and CLTL(QFP). Au-

thors consider also a class of counter systems (Q, n, δ) in CS(PA) which is the same as the class considered by Ibarra in [3] such that transition relation is defined by tuples of the form $(q, (zero(\boldsymbol{b}), \boldsymbol{d}), q')$ which are a shorthand for (q, ξ, q') where:

$$\xi := \bigwedge_{i\in[1,n]:\vec{b}(i)=1} x_i = 0 \ \wedge \bigwedge_{i\in[1,n]:\vec{b}(i)=0} x_i \neq 0 \ \wedge \bigwedge_{i\in[1,n]} x_i' = x + \boldsymbol{d}(i).$$

Such automata constitute the class of n-D counter automata (n-D-CA).

CLTL model-checking problem is usually defined with respect to runs of counter systems which are infinite sequences of the form $(Q \times D^n)^\omega$ where n is the dimension of counter system and D is the domain of interpretation of variables.

CLTL MODEL-CHECKING PROBLEM:

Input	- a counter systems $\mathcal{S} = (Q, n, \delta) \in \mathrm{CS}(L)$, with $L \subseteq$ PA - a CLTL(QFP) formula ϕ
Problem	does there exists a *infinite* run $\rho \in (Q \times D^n)^\omega$ such that $\rho, 0 \models \phi$, written $\mathcal{S} \models \phi$

Initial configuration can be included in the formula ϕ.

The first undecidability result for satisfiability of CLTL^1_3(DL) is given by Comon and Cortier [5] by showing that halting runs of a Minsky machine can be encoded into a CLTL formula. One auxiliary counter encodes the control state of the system.

Theorem 45 (Theorem 3, [5]). *The satisfiability problem for* CLTL^1_3(DL) *is* Σ^1_1*-hard.*

Authors of [5] suggest a way to regain decidability by means of a syntactic restriction on formulae including **U** temporal operator. The "flat" fragment of CLTL^1_ω(DL) consists of CLTL formulae such that subformula ϕ of $\phi \mathbf{U} \psi$ is $\top$, $\bot$ or a conjunction $\zeta_1 \wedge \cdots \wedge \zeta_m$ where $\zeta_i \in$ DL. The fragment of "flat" CLTL^1_ω(DL) benefits of nice correspondence with a special class of counter system (flat relational counter system) with Büchi acceptance condition for which nonemptiness problem is decidable. Comon and Jurski analyze this class of counter system in [76].

Analogously to CLTL^1_ω(DL) the language CLTL^2_1(DL) is enough expressive to encode accepting runs of Minsky machines. In fact, even though CLTL language has less than three variables expressiveness of the logic does not necessarily decrease. The existence of accepting runs for counter systems $(Q, 2, \delta)$ in CS(DL) can be reduced to satisfiability

problem for $\mathrm{CLTL}_1^2(\mathrm{DL})$ formulae. Moreover, since CS(DL) with counter over $\mathbb{N}$ can simulate nondeterministic Minsky machines whose recurrence problem is known to be Σ_1^1-hard by Alur and Henzinger [77], the same reduction can be used also to prove Σ_1^1-hardness of satisfiability problem for $\mathrm{CLTL}_1^2(\mathrm{DL})$.

Theorem 46 (Theorem 1, [74]). *The satisfiability problem for* $\mathrm{CLTL}_1^2(\mathrm{DL})$ *is* Σ_1^1*-complete.*

From previous undecidability result, satisfiability problem for $\mathrm{CLTL}_2^1(\mathrm{DL})$ can be proved to be Σ_1^1-complete.

Theorem 47 (Theorem 2, [74]). *The satisfiability problem for* $\mathrm{CLTL}_2^1(\mathrm{DL})$ *is* Σ_1^1*-complete.*

Satisfiability problem for CLTL(DL) can be reduced to CLTL(DL) model-checking problem by considering $\mathcal{S} = (\{q\}, n, (q, \top, q))$ as model. Then, a CLTL(DL) formula ϕ is satisfiable if, and only if, $\mathcal{S} \models \phi$.

Corollary 48 (Corollary 1, [74]). *The model-checking problem for* $\mathrm{CLTL}_1^2(\mathrm{DL})$ *and for* $\mathrm{CLTL}_1^1(\mathrm{DL})$ *are* Σ_1^1*-complete.*

When CLTL admits atomic formulae in QFP, undecidability of satisfiability problem for $\mathrm{CLTL}_1^1(\mathrm{QFP})$ and $\mathrm{CLTL}_1^1(\mathrm{QFP})$ model-checking follows directly from undecidability results for halting and recurrence problem of one counter machines with multiplication and division by constants [78].

Theorem 49 (Lemma 6, [74]). *The satisfiability problem for* $\mathrm{CLTL}_1^1(\mathrm{QFP})$ *and* $\mathrm{CLTL}_1^1(\mathrm{QFP})$ *model-checking over* 1-QFP *counter system are* Σ_1^1*-complete.*

Satisfiability for $\mathrm{CLTL}_\omega^\omega(\mathrm{IPC}^*)$ and $\mathrm{CLTL}_\omega^\omega(<,=)$ over $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$ is obtained by Demri and Gascon in [18] by reducing the problem to emptiness problem for Büchi automata. Notions presented hereafter are taken from [74] and [6] and they are presented as fundamental background for next chapters. We consider L to be the structure defined by $\{\mathrm{IPC}^*, (D, <, =)\}$, where $D = \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$, until a new signature will be specified. Moreover, we will simply write CLTL to denote $\mathrm{CLTL}_\omega^\omega$. In order to represent models of a $\mathrm{CLTL}(L)$ formula ϕ by means of automata, authors represent symbolically all sequences σ such that $(\sigma, 0) \models \phi$. It is worth noticing that the considered language does not include explicit set of atomic proposition, which can be represented by binary variables of domain $\{0, 1\}$. Let ϕ be a $\mathrm{CLTL}(L)$ formula, $terms(\phi)$ be the set of arithmetic terms of the form $\mathrm{X}^i x$ for all $0 \leq i \leq \lceil \phi \rceil$ and for all $x \in V$

and $c(\phi)$ be the set of constants occurring in ϕ. A set of atomic formulae over $terms(\phi)$ is *maximally consistent* if, for every atomic formula θ over $terms(\phi)$ and $c(\phi)$, either θ or $\neg\theta$ is in the set. Symbolic models are used by Demri et al. [6], [74], [18] to represent non arithmetic model of formulae.

Definition 50. *A* symbolic valuation *sv for ϕ is a maximally consistent set of atomic formulae over $terms(\phi)$ and $c(\phi)$; the set of all symbolic valuations for ϕ is denoted by $SV(\phi)$.*

A valuation $\mathbf{val} : V \to D$ naturally extends to a valuation $\mathbf{val}' : terms(\phi) \to D$, such that $\mathbf{val}' \models_{\mathcal{D}} R(\alpha_1, \alpha_2)$ if, and only if, $R(\mathbf{val}'(\alpha_1), \mathbf{val}'(\alpha_2))$. Then, a symbolic valuation sv for ϕ is *satisfiable* if there exists a valuation $\mathbf{val}' : terms(\phi) \to D$ such that $\mathbf{val}' \models_{\mathrm{PA}} \xi$, for all ξ belonging to sv. We write $\mathbf{val}' \models_{\mathrm{PA}} sv$ when sv is satisfied by $\mathbf{val}'$. Given a symbolic valuation sv and an atomic formula ξ (over a.t.t.'s), we write $sv \models_s \xi$ if for every valuation $\mathbf{val}'$ such that $\mathbf{val}' \models_{\mathrm{PA}} sv$ then $\mathbf{val}' \models \xi$. Observe that in the considered constraint systems, the problem of checking whether $sv \models_s \xi$ is decidable, since both the language in L are subset of Presburger arithmetic. All symbolic valuations may be defined by means of a syntactic construction on formula ϕ by using the procedure in [6]. In order to correctly represent the arithmetic model of formulae, authors introduce the notion of local consistency defining a rule to determine which pair of symbolic valuations can be adjacent, i.e. their relations do not contradict themselves. A pair of symbolic valuations (sv_1, sv_2) is *locally consistent* if, for all R in L:

$$R(\mathrm{X}^{i_1}x_1, \mathrm{X}^{i_2}x_2) \in sv_1 \text{ implies } R(\mathrm{X}^{i_1-1}x_1, \mathrm{X}^{i_2-1}x_2) \in sv_2.$$

A sequence of symbolic valuations $sv_0, sv_1, \ldots$ is *locally consistent* if all pairs (sv_i, sv_{i+1}), $i \geq 0$, are locally consistent. A locally consistent infinite sequence $\rho = (SV(\phi))^\omega$ of symbolic valuations *admits a model*, written $\sigma \models \rho$, if there exists a model $\sigma : \mathbb{N} \times V \to D$ of ϕ such that for every $i \geq 0$, $\sigma, i \models \rho(i)$. In this case, ρ is called a *symbolic model* for ϕ. The satisfaction relation $\models_s$ can also be extended to sequences of symbolic valuations; it is the same as $\models$ for all temporal operators except for atomic formulae:

$$\rho, i \models_s \xi \Leftrightarrow \rho(i) \models_s \xi.$$

The following fundamental proposition draws a link between the satisfiability by sequences of symbolic valuations and by sequences of valuations.

Proposition 51 (Lemma 3.1, [6])**.** *A CLTL(L) formula ϕ is satisfiable iff there exists a symbolic model for ϕ.*

Given a CLTL(L) formula ϕ, it is possible [6] to define an automaton $\mathcal{A}_\phi$ recognizing symbolic models of ϕ, which reduces satisfiability of CLTL(L) to emptiness of Buchi automata. The idea is that automaton $\mathcal{A}_\phi$ should accept the intersection of the following languages:

(i) the language of LTL models ρ;

(ii) the language of sequences of locally consistent symbolic valuations;

(iii) the language of sequences of symbolic valuations for ϕ which admit an arithmetic model.

Languages (i) and (ii) can be accepted by Büchi automata, called respectively $\mathcal{A}_s$ and $\mathcal{A}_\ell$. In general, however, the language (iii) may *not* be ω-regular. Nonetheless, automaton $\mathcal{A}_\phi$ can be defined to accept a superset of the language of the sequences of locally consistent symbolic valuations that are models for ϕ, such that the *ultimately periodic* models of $\mathcal{A}_\phi$ are all *ultimately periodic* models of ϕ. Then, from Lemma (52) below, it follows that ϕ is satisfiable iff $\mathcal{A}_\phi$ recognizes an *ultimately periodic* word.

$\mathcal{A}_\phi$ is defined as the product of automata $\mathcal{A}_\ell$, $\mathcal{A}_s$, and $\mathcal{A}_C$, where $\mathcal{A}_C$ defines a condition C guaranteeing the existence of a sequence σ such that $\sigma \models \rho$. In particular, for IPC* and the structure $(D, <, =)$ over $\{\mathbb{N}, \mathbb{Z}\}$, $\mathcal{A}_C$ can effectively be built. Condition C is given by considering the graph representation G_ρ of sequences of symbolic valuations ρ. It enforces the absence of infinite $<$-strict paths in graph G_ρ, i.e., that between any two nodes of G_ρ there are no paths of infinite length in which relation $<$ occurs (see details in [6]). When the condition C is sufficient and necessary for the existence of models σ such that $\sigma \models \rho$, then automaton $\mathcal{A}_\phi$ represents all the sequences of symbolic valuations which admit a model. A fundamental lemma, on which Proposition 53 below relies on, draws a sufficient and necessary condition for the existence of models of sequences of symbolic valuations.

Lemma 52 (Lemma 6.2, [6]). *Let ρ be an ω-periodic sequence of symbolic valuations of the form $\rho = \alpha(\beta)^\omega$ that is locally consistent. Then ρ admits a model σ iff ρ satisfies C.*

Proposition 53 (Lemma 11, [18]; Lemma 6.3, [74]). *A* CLTL(L) *formula, where* IPC* *and* $(\mathbb{Z}, <, =)$ *over* $\{\mathbb{N}, \mathbb{Z}\}$*, is satisfiable iff the language* $\mathscr{L}(\mathcal{A}_\phi)$ *is not empty.*

Theorem 54 (Theorem 1, [18]; Theorem 6.6, 7.3, [74]). *The satisfiability problem for* CLTL(IPC*) *and* CLTL($<, =$) *over* $\{\mathbb{N}, \mathbb{Z}\}$ *is* PSPACE*-complete.*

Hardness can be proved by reducing satisfiability problem for LTL to satisfiability of CLTL for both language IPC* and $(<,=)$ considered. Upper bound derives form construction of Büchi automaton $\mathcal{A}_s$. It is worth noticing that this complexity result is valid when the complexity of the problem of checking whether $sv \models_s \xi$ is decidable in PSPACE.

The construction of $\mathcal{A}_\phi$ can be made simpler when L benefits of a property of *completion* which is introduced by Balbiani and Condotta in [79].

Completion property

Each automaton involved in the definition of $\mathcal{A}_\phi$ has the function of "filtering" sequences of symbolic valuations so that 1) they are locally consistent, 2) they satisfy an LTL property and 3) they admit a (arithmetic) model. For some constraint systems, admitting a model is a consequence of local consistency. A set of relation over D has the *completion* property if, given:

(i) a symbolic valuation sv over a finite set of variables $H \subseteq V$,

(ii) a subset $H' \subseteq H$,

(iii) a valuation $\mathbf{val}'$ over H' such that $\mathbf{val}' \models sv'$, where sv' is the subset of atomic formulae in sv which uses only variables in H'

then there exists a valuation $\mathbf{val}$ over V extending $\mathbf{val}'$ such that $\mathbf{val} \models sv$. An example of such a relational structure is $(\mathbb{R}, <, =)$.

Lemma 55 (Lemma 5.3, [6]). *Let $(D, <, =)$ a relational structure where D is infinite and $<$ is a total order. Then, it satisfies the completion property iff D is dense and open.*

The following result relies on the fact that every locally consistent sequence of symbolic valuations, with respect to the relational structure $\mathcal{D}$ which have completion, admits a model.

Proposition 56. *Let $\mathcal{D}$ be a relational structure satisfying the completion property and ϕ be a CLTL($\mathcal{D}$) formula. Then, the language of sequences of symbolic valuations which admit a model is ω-regular.*

In this case, automaton $\mathcal{A}_\phi$ recognizing the sequence of symbolic valuations may be defined by $\mathcal{A}_\phi = \mathcal{A}_s \times \mathcal{A}_\ell$.

Finally, automata-based approach is exploited by Demri and Gascon in [74] to obtain decidability of satisfiability problem for $\mathrm{CLTL}^1_1(\mathrm{DL}^+ \cup \mathbf{AP})$ where $\mathbf{AP}$ is a language of atomic proposition. Models of $\mathrm{CLTL}^1_1(\mathrm{DL}^+ \cup$

AP) formulae are pair (σ', σ'') as defined for LTL(**FO**). In this case, the automaton $\mathcal{A}_\phi$ can be effectively built in polynomial space and accepts the intersection of the following two languages:

1. the language of symbolic models which symbolically satisfy the $\text{CLTL}_1^1(\text{DL}^+ \cup$ **AP**) formula ϕ, i.e. $\rho \models_s \phi$, and

2. the language of symbolic models such that ρ is satisfiable.

The automaton for the language **1.** is an extension of the Vardi-Wolper automata recognizing model for LTL formulae similar to the one defined also for CLTL(IPC*) or CLTL$(<,=)$. The construction of $\mathcal{A}_{sat}$ which recognize the language **2.**, is quite involved and its definition can be found in [74]. $\mathcal{A}_{sat}$ is a 1-$\mathbb{Z}$ counter system of alphabet $2^{AP} \times SV(\phi)$ which can perform ε-transitions. Intuitively, the automaton consists of a set of components which has the function either to check a formulae over the counter or to update the value of the counter. A component $\mathcal{A}_{\alpha,sv}$ is a 1-$\mathbb{Z}$ counter system which checks if the formulae α belonging to the symbolic valuation sv is satisfied. For instance, the component $\mathcal{A}_{x=d,sv}$, where d is a constant defined in the formula ϕ, decrements x by 1, d times and then it checks if $x = 0$; after it restore the original value of x by performing d increment by 1. Each component $\mathcal{A}_{\alpha,sv}$ contribute to enforce that the next symbolic valuation is exactly sv.

Proposition 57 (Theorem 3, [74]). *A* $\text{CLTL}_1^1(\text{DL}^+ \cup \textbf{AP})$ *formulae is satisfiable if, and only if,* $\mathscr{L}(\mathcal{A}_\phi)$ *is not empty. The satisfiability problem for* $\text{CLTL}_1^1(\text{DL}^+ \cup \textbf{AP})$ *is* PSPACE*-complete.*

The presence of propositional variables in $\text{CLTL}(\text{DL}^+ \cup \textbf{AP})$ makes the reduction from $\text{CLTL}(\text{DL}^+)$ model-checking problem for for CS(DL) counter system to satisfiability of $\text{CLTL}(\text{DL}^+ \cup \textbf{AP})$ straightforward. It is similar to the reduction from LTL model-checking problem to satisfiability of LTL which is proposed by Sistla and Clarke in [52]. $\text{CLTL}_1^\omega(\text{QFP})$ model-checking problem for 1-$\mathbb{Z}$ counter systems $\mathcal{A}$ is solved by directly constructing a 1-$\mathbb{Z}$ counter systems $\mathcal{A}_{sat}$ over the alphabet $2^{AP} \times SV(\mathcal{A}, \phi)$ where the set $SV(\mathcal{A}, \phi)$ is the set of symbolic valuation obtained from both $\mathcal{A}$ and ϕ. Therefore, $\mathscr{L}(\mathcal{A}_{sat})$ is the language of symbolic models ρ such that $\rho, 0 \models_s \phi$ and $\mathcal{A} \models \phi$.

Proposition 58 (Lemma 9, [74]). *Let* ϕ *be a* $\text{CLTL}_1^\omega(\text{QFP})$ *formula and* $\mathcal{A}$ *be a* 1-$\mathbb{Z}$ *counter systems. Then it is possible to build the automaton* $\mathcal{A}_\phi$ *from the intersection of* $\mathcal{A}_s$ *and* $\mathcal{A}_{sat}$ *such that* $\mathscr{L}(\mathcal{A}_\phi)$ *is empty if, and only if,* $\mathcal{A} \models \phi$.

Both Propositions 57 and 58 rely on the decidability of nonemptiness problem for 1-$\mathbb{Z}$ counter systems:

Theorem 59 (Theorem 6, [74]). *Let $\mathcal{A}$ be a 1-$\mathbb{Z}$ counter systems. Checking whether $\mathscr{L}(\mathcal{A})$ is nonempty is* NLOGSPACE*-complete.*

Decidability of nonemptiness for 1-$\mathbb{Z}$ counter systems allow us to obtain decidability of satisfiability and model-checking problem for some CLTL fragment considered.

Theorem 60 (Theorem 3, Corollary 3, Theorem 4, [74]). *Following problems are* PSPACE*-complete.*

- *satisfiability of* $\mathrm{CLTL}_1^1(\mathrm{DL}^+ \cup \mathbf{AP})$.
- $\mathrm{CLTL}_1^1(\mathrm{DL})$ *model-checking problem for* CS(DL) *counter system and* 1-$\mathbb{Z}$ *counter systems.*
- $\mathrm{CLTL}_1^\omega(\mathrm{QFP})$ *model-checking problem for* 1-$\mathbb{Z}$ *counter systems.*

Given a logical formalism, different notions of satisfiability can be defined. Bounded satisfiability is weaker than standard satisfiability since it looks for bounded models of formulae which may be not enough to deduce immediately satisfiability (or unsatisfiability) with respect to the standard notion. However, in some cases, bounded satisfiability can still be used to solve satisfiability and model-checking problems. For instance, this is the case of bounded LTL model-checking and satisfiability. Informally, let P be a problem and P_b the bounded version of P. Whenever there exists a finite bound on the number of bounded instances P_b which one have to solve to answer to a P problem, we say that the bounded approach is complete. In this chapter, we extend the notion of bounded satisfiability of LTL to logical formalisms involving variables over infinite domains. In particular, when fragments of Presburger arithmetic like IPC* or $(D, <, =)$ characterizes language CLTLB (CLTL with past-time modalities), we prove that bounded satisfiability problem is complete with respect to satisfiability for CLTLB. The strict relation between model-checking and satisfiability allow us to solve model-checking by reducing an instance of model-checking problem to satisfiability over bounded models. Completeness for bounded model-checking problem is a direct consequence of completeness of bounded satisfiability.

Beside symbolic valuations as defined in Section 3.1, we will use a weaker definition of symbolic valuation than the one introduced by Demri et al. [6], [74], [18]. Let ϕ be a CLTLB(L) and $atoms(\phi)$ be the set of atomic formulae of L in the formula ϕ. A *weak* symbolic valuation sv

for ϕ is a maximally consistent set of formulae belonging to $atoms(\phi)$ and the set of all weak symbolic valuations for ϕ is denoted by $SV_w(\phi)$. The notion of local consistency is refined and it is adapted to weak symbolic valuations. A pair of symbolic valuations (sv_1, sv_2) is *weak locally consistent* if do **not** happen:

$$R(\mathrm{X}^{i_1}x_1, \ldots, \mathrm{X}^{i_n}x_n) \in sv_1 \text{ and } \neg R(\mathrm{X}^{i_1-1}x_1, \ldots, \mathrm{X}^{i_n-1}x_n) \in sv_2.$$

where $R \in \mathcal{R}$ is an n-ary relation in $\mathcal{R}$. A sequence of symbolic valuations $sv_0, sv_1, \ldots$ is *weak locally consistent* if all pairs (sv_i, sv_{i+1}), $i \geq 0$, are weak locally consistent.

3.2. Bounded satisfiability for LTL(**FO**) over uninterpreted functions

Bounded Satisfiability Problem for LTL(**FO**) with uninterpreted functions and predicates is defined by considering bounded symbolic models of LTL(**FO**) formulae. Let ϕ be a LTL(**FO**) and $V = V_l \cup V_g$ be the finite set of local and global variables in ϕ. Then, a bounded symbolic model is, informally, a finite representation of

- a possibly infinite LTL(**FO**) model (S, π, L)
- an infinite LTL(**FO**) symbolic models over the alphabet of (weak) symbolic valuations $SV(\phi)$ (or $SV_w(\phi)$).

Bounded satisfiability is defined with respect to a k-bounded sequences of assignment to variables and atomic propositions:

- $\widehat{\sigma}'_k : \{0, \ldots, k + \lceil\phi\rceil_x\} \times \{x\} \to D$, for every $x \in V_l$,
- $\widehat{\sigma}''_k : \{0, \ldots, k\} \times p \to \{true, false\}$, for every $p \in AP$.

Moreover, we consider a finite sequence ρ, $|\rho| = k$, of weak symbolic valuations and a bounded satisfaction relation $\models_k$ defined as follows:

$$\widehat{\sigma}_k \models_k \rho \overset{\text{def}}{\Leftrightarrow} \widehat{\sigma}_k, i \models_{\mathcal{D}} \rho(i) \text{ for all } 0 \leq i \leq k.$$

In this case, we say that ρ admits a k-bounded model $\widehat{\sigma}_k$. A k-bounded model for a LTL(**FO**) is a structure (S, π, L) such that π is a finite sequence of states of of length $\pi \in S^k$ and:

- $\sigma'(x) = \widehat{\sigma}'_k(x)$, for every $x \in V_l$ and $0 \leq i \leq k$,
- $\sigma''(p) = \widehat{\sigma}''_k(p)$, for every $p \in AP$ and $0 \leq i \leq k$.

Value of terms is defined by the function $[]_{(\pi,i)}$ with respect to the first-order structure $(\mathcal{M},\sigma)$ at each position $\pi(i) = s_i$, such that $L(s_i) = (\mathcal{M},\sigma)$, and the environment ε for global variables:

$$[x]_{(\pi,i)} = \begin{cases} \sigma'(x) \text{ for } x \in V_l \text{ for } 0 \leq i \leq k \\ \widehat{\sigma}_k(x) \text{ for } x \in V_l \text{ for } i \geq k \\ \varepsilon(x) \text{ for } x \in V_g \end{cases}$$

$$[f(\tau_1, \ldots \tau_n)]_{(\pi,i)} = \begin{cases} f_{\mathcal{I}_i}\left([\tau_1]_{(\pi,i)}, \ldots, [\tau_n]_{(\pi,i)}\right) \text{ for } 0 \leq i \leq k \\ c \in D \text{ for } i > k \end{cases}$$

$$[\mathrm{X}\tau]_{(\pi,i)} = [\tau]_{(\pi,i+1)}$$

where $f_{\mathcal{I}_i}$ is the n-ary function f with respect to the interpretation $\mathcal{I}_i$.

Bounded symbolic models for a formula are introduced similarly as for satisfiability of CLTLB. Let β be the conjunction of $R(\tau_1, \ldots, \tau_n)$ and all atomic formulae belonging to sv. Let v be an assignment which maps $terms(\beta)$ to element of D. Given a symbolic valuation sv, when $R \in \mathcal{F}$ then $sv \models_s R(\tau_1, \ldots, \tau_n)$ amounts to check whether $v \models_{\mathcal{T}} sv$ then $v \models_{\mathcal{T}} R(\tau_1, \ldots, \tau_n)$ for all v. We suppose that the satisfiability problem for the theory $\mathcal{T}$ of uninterpreted function combined with the first-order theory is decidable. Let β be the conjunction of all atomic formulae belonging to sv and $R(\tau_1, \ldots, \tau_n)$. Then, symbolic satisfaction relation for $R \in \mathcal{R}$ is:

$$\rho, i \models_s R(\tau_1, \ldots, \tau_n) \overset{\text{def}}{\Leftrightarrow} \rho(i) \models_s R(\tau_1, \ldots, \tau_n)$$

When ρ is an ultimately periodic model of the form uv^ω, where $u \in SV_w(\phi)^*$ and $v \in SV(\phi)^+$ then symbolic satisfaction relation $\models_s^k$ is the same as $\models_s$. In the case of finite prefixes $\rho = u \in SV(\phi)^k$, we adapt $\models_s^k$ to finite sequences of symbolic valuations:

$$\begin{aligned} \rho, i \models_s^k p &\overset{\text{def}}{\Leftrightarrow} p \in L(\pi(i)) \\ \rho, i \models_s^k R(\tau_1, \ldots, \tau_n) &\overset{\text{def}}{\Leftrightarrow} \rho(i) \models_s R(\tau_1, \ldots, \tau_n) \\ \rho, i \models_s^k \mathbf{X}\phi &\overset{\text{def}}{\Leftrightarrow} \rho, i+1 \models_s^k \phi \text{ and } 0 \leq i+1 \leq k \\ \rho, i \models_s^k \phi \mathbf{U} \psi &\overset{\text{def}}{\Leftrightarrow} \exists i \leq j \leq k \ \rho, j \models_s^k \psi \text{ and } \forall i \leq n < j \ \rho, n \models_s^k \phi \end{aligned}$$

Finally, we say that a LTL(**FO**) formula ϕ is *k-bounded satisfiable* there exists k-bounded model (S, π, L), such that $\pi \in S^k$, an environment ε and $\widehat{\sigma}_k$ such that, for a finite (locally consistent) sequence of symbolic valuations $\rho \in SV(\phi)^k$:

$$\pi, 0 \models_{\varepsilon}^{\widehat{\sigma}_k} \phi \overset{\text{def}}{\Leftrightarrow} \widehat{\sigma}_k \models_k \rho \text{ and } \rho, 0 \models_s^k \phi.$$

According to definition given so far, satisfiability problem over k-bounded arithmetic models $\widehat{\sigma}_k$ is:

k-BOUNDED SATISFIABILITY PROBLEM WITH UF_ε

Input	a CLTLB(L) formula ϕ, a constant $k \in \mathbb{N}$
Problem	there exist a k-bounded model (S, π, L), an environment ε, a sequence $\widehat{\sigma}_k$ and a bounded symbolic model $\rho \in SV(\phi)^k$ (or $SV_w(\phi)^k$) such that: • if $\rho = uv$, then $uv^\omega, 0 \models_s^k \phi$; or, • if $\rho = u$, then $u(SV_w(\phi))^\omega, 0 \models_s^k \phi$ and $\widehat{\sigma}_k \models_k \rho$.

Theorem 61. *k-bounded Satisfiability for* LTL(**FO**) *with uninterpreted functions and predicates is decidable.*

Proof. The problem can be reduced in polynomial time to the satisfiability of a formula in the combined theory of equality and uninterpreted functions. The reduction is defined in Section 3.3.5. In particular, let ϕ be a LTL(**FO**) formula. Then we check satisfiability of

$$[\phi \wedge \mathbf{G}(\bigvee_{i=1}^{m} sv)]_k$$

where $[\zeta]_k$ denotes the encoding of ζ which is provided in next Section 3.3. □

It is worth noticing that weak symbolic valuations are implicitely represented in the encoding of Section 3.3.5. In fact, a formula $\neg\alpha \in sv$ holds when its negated form α holds, and viceversa.

An example of k-bounded satisfiability with uninterpreted functions and free variables is proposed in Section 5.1.

3.3. Encoding for LTL(**FO**)

In this section, the SAT-based encoding for LTLB proposed by Biere et al. in [63] is compared with a new version which exploits satisfiability of the theory QF-EU$\mathcal{D}$ where $\mathcal{D} = (\mathbb{N}, <)$. QF-EU$\mathcal{D}$ encoding is proved to be more concise than the propositional encoding of the same formula and it is tailored to be implemented on SMT-solvers. After, we define

encodings for CLTLB(L) and for LTL(**FO**) by enriching the encoding of LTLB formulae. Satisfiability of formulae of the theory QF-EUL, provided that the union of theories of equality and uninterpreted functions with L is consistent, is used to reduce (k-bounded) satisfiability problem for CLTLB(L) and for LTL(**FO**) and to solve bounded CLTLB model-checking. The extension is quite natural since temporal structure of CLTLB and LTL(**FO**) formulae is the same as LTL.

When the satisfiability problem for QF-EUL or EUL is decidable and has effective implemented decision procedure, then k-bounded satisfiability problem for LTL(L), where L is a first-order definable language, can be decided. In particular, the same problem for LTL(**FO**) with uninterpreted function and CLTLB(L) where $L = \{\mathrm{PA}, \mathrm{DL}^+, \mathrm{IPC}^*\}$ and for structure like $(D, <, =)$ over $\mathbb{N}, \mathbb{Z}, \mathbb{R}$, is decidable.

3.3.1. Linear Encoding of LTL for SMT

Alternatively to encoding presented in Section 2.7.4, the ultimately periodic semantics of LTL formulae is encoded as a quantifier-free formula in the theory EUF$\cup\mathcal{D}$ (QF-EU$\mathcal{D}$), where EUF is the theory of Equality and Uninterpreted Functions, and $\mathcal{D} = (\mathbb{N}, <)$. The resulting union of theories EUF $\cup\, \mathcal{D}$ is consistent because EUF $\cup\, \mathcal{D}$ is union of two consistent, disjoint, stably infinite theories (as is the case for EUF and arithmetic). The proposed encoding is based on the "eventuality encoding" explained in Section 3.3.1. Not all the parts of SAT encoding are modified but formulae defining loop constraints, last state formulae and eventuality are strongly revised and adapted to be encoded as QF-EU$\mathcal{D}$ formulae.

Let ϕ be a LTLB formula, AP be the set of atomic propositions and $[\phi]_k$ be the QF-EUL formula representing models $\pi \in S^\omega$ of ϕ such that $[\phi]_k$ is satisfiable if, and only if, $\pi \models_k \phi$. Following set of QF-EU$\mathcal{D}$ formulae constitutes the encoding $[\phi]_k$ representing infinite ultimately periodic models of ϕ of the form uv^ω such that $|uv| = k$ or uS^ω such that $|u| = k$.

3.3.2. Encoding periodicity

Differently from Boolean encoding, ultimately periodic models of the form $a(sb)^\omega$ are represented by a QF-EU$\mathcal{D}$ formula involving one non-negative integer *loop-selecting* variable $\boldsymbol{\ell} \in \mathbb{N}$:

$$\bigwedge_{i=1}^{k} (\boldsymbol{\ell} = i \Rightarrow L(\pi(i-1)) = L(\pi(k)))\,.$$

There is an immediate equivalence between the above formula and propositional representation in Section 2.7.5. Subformula $\ell = i$ holds if, and only if, l_i is satisfied and all positions $\boldsymbol{\ell} \leq i \leq k$ are such that $inLoop_i$ holds. Since indeces $i \in [1, k]$ then $\neg l_0$ and $\neg inLoop_0$ are satisfied. It is worth noticing that subformula $\bigvee_{i=1}^{k} (\boldsymbol{\ell} = i)$, which is used later, is equivalent to $loopEx$.

Previous formula is extended to LTL(**FO**) by considering periodicity over relations $R \in \mathcal{R}$ occurring into ϕ, both for interpreted (like $<$) and uninterpreted relations.

$$\bigwedge_{i=1}^{k} \left((\boldsymbol{loop} = i) \Rightarrow \bigwedge_{\theta \in \mathcal{R}} \bigwedge_{\alpha_1, \ldots, \alpha_n \in terms(\phi)} \boldsymbol{\theta}_{i-1} \Leftrightarrow \boldsymbol{\theta}_k \right).$$

3.3.3. Encoding the Propositional Terms

The QF-EUL encoding associates to each propositional subformula a *formula predicate* that is a unary uninterpreted predicate $\boldsymbol{\varphi} : \mathbb{N} \rightarrow \{true, false\}$.

Variables defining the QF-EUL formula are written in boldface. Improperly, in order to be consistent with the notation of pedices used in proof of Theorem 68, and for reasons of ease of writing, we shall write $\boldsymbol{\phi}_i$ instead of $\boldsymbol{\phi}(i)$. When the subformula φ holds at instant i then $\boldsymbol{\varphi}(i)$ holds (written $\boldsymbol{\varphi}_i$).

As the length of paths is fixed to $k + 1$, and all paths start from 0, formula predicates are subsets of $\{0, \ldots, k+1\}$. Let φ be a propositional subformula of ϕ. The formula predicate associated with φ (denoted by the same name but written in bold face), is recursively defined in the same way as 2.7.6:

φ	$0 \leq i \leq k+1$
p	$\boldsymbol{\varphi}_i \Leftrightarrow p \in L(s_i)$
$\neg\psi$	$\boldsymbol{\varphi}_i \Leftrightarrow \neg\boldsymbol{\psi}_i$
$\zeta \wedge \psi$	$\boldsymbol{\varphi}_i \Leftrightarrow \boldsymbol{\zeta}_i \wedge \boldsymbol{\psi}_i$

The conjunction of all the constraints for all the subformulae φ of ϕ constitutes the formula $|PropConstraints|_k$.

3.3.4. Encoding Temporal Operators

Temporal subformulae constraints ($|TempConstraints|_k$) define the basic temporal behavior of future and past operators, by using their traditional fixpoint characterizations. Encoding for subformulae $\mathbf{X}\psi$, $\zeta\mathbf{U}\psi$ and $\zeta\mathbf{R}\psi$ is the same as Boolean encoding but defined over formula predicates.

Analogously to Boolean encoding, last state constraints define the equivalence between truth in $k+1$ and those one indicated by $\boldsymbol{\ell}$. Constraints have a similar structure to the corresponding Boolean ones, but here they are defined by one QF-EU$\mathcal{D}$ formula, for each subformula φ of ϕ, with respect to the variable $\boldsymbol{\ell}$:

$$\left(\bigvee_{i=1}^{k} \boldsymbol{\ell} = i\right) \Leftarrow \boldsymbol{\varphi}_{k+1}$$
$$\left(\bigvee_{i=1}^{k} \boldsymbol{\ell} = i\right) \Rightarrow (\boldsymbol{\varphi}_{k+1} \Leftrightarrow \boldsymbol{\varphi}_i)$$

Note that if a loop does not exists then the fixpoint semantics of **R** is exactly the bounded semantics defined over finite acyclic path in Section 2.7.3. Finally, to correctly define the semantic of **U** and **R**, their *eventuality* have to be enforced. As explained in Section 2.7.7, if $\zeta \mathbf{U} \psi$ holds at i, then ψ eventually holds in $j \geq i$. When $\zeta \mathbf{R} \psi$ does not hold at i, then ψ eventually does not hold in $j \geq i$. Along finite models of length k, eventualities must hold between 0 and k. If a loop exists, an eventuality may holds within the loop. In the QF-EU$\mathcal{D}$ encoding, one variable $\boldsymbol{j}_\psi \in \mathbb{N}$ is introduced for each ψ occurring in a subformula $\zeta \mathbf{U} \psi$ or $\zeta \mathbf{R} \psi$.

φ	Base
$\zeta \mathbf{U} \psi$	$\left(\bigvee_{i=1}^{k} \boldsymbol{\ell} = i\right) \Rightarrow (\boldsymbol{\varphi}_k \Rightarrow \boldsymbol{\ell} \leq \boldsymbol{j}_\psi \leq k \wedge \boldsymbol{\psi}_{\boldsymbol{j}_\psi})$
$\zeta \mathbf{R} \psi$	$\left(\bigvee_{i=1}^{k} \boldsymbol{\ell} = i\right) \Rightarrow (\neg\boldsymbol{\varphi}_k \Rightarrow \boldsymbol{\ell} \leq \boldsymbol{j}_\psi \leq k \wedge \neg\boldsymbol{\psi}_{\boldsymbol{j}_\psi})$

The conjunction of all the constraints for all the subformulae ζ of ϕ constitutes the formula $|Eventually|_k$. The equivalence of previous formulae and the ones in Section 2.7.7 is easy to be shown.

- Let us suppose that *loopEx* holds and $\zeta \mathbf{U} \psi$ holds at position k. Then, the auxiliary formula $\langle \mathbf{F} \psi \rangle$ holds at position k, and, recursively, $\langle \mathbf{F} \psi \rangle_i \Leftrightarrow \langle \mathbf{F} \psi \rangle_{i-1} \vee (inLoop_i \wedge \psi_i)$ with $\neg \langle \mathbf{F} \psi \rangle_0$. If l_i holds, then $\ell = i$ is the position of the loop. By recursive definition of $\langle \mathbf{F} \psi \rangle$, there exists a position $\ell \leq j \leq k$ such that ψ_j holds satisfying the base case $inLoop_i \wedge \psi_i$. This entails the QF-EU$\mathcal{D}$ formula for $\zeta \mathbf{U} \psi$. Conversely, if $\ell \leq j \leq k$ and φ_k holds, with $\varphi = \zeta \mathbf{U} \psi$, then ψ_j holds such that $\ell \leq j \leq k$. Then, in the propositional encoding, $inLoop_j$ is satisfied, all auxiliary formulae $\langle \mathbf{F} \psi \rangle_i$, such that $j \leq i \leq k$, hold because of the recursive definition. Finally, the formula $loopEx \Rightarrow ((\zeta \mathbf{U} \psi)_k \Rightarrow \langle \mathbf{F} \psi \rangle_k)$ is satisfied.

- Let us suppose that $\zeta \mathbf{R} \psi$ does not hold at position k. Then, the auxiliary formula $\langle \mathbf{G} \psi \rangle$ is false at position k, and, recursively,

$\langle \mathbf{G}\psi\rangle_i \Leftrightarrow \langle \mathbf{G}\psi\rangle_{i-1} \wedge (\neg inLoop_i \vee \psi_i)$ with $\langle \mathbf{F}\psi\rangle_0$. If l_i holds, then $\ell = i$ is the position of the loop. By recursive definition of $\langle \mathbf{G}\psi\rangle$, there exists a position $\ell \leq j \leq k$ such that $\neg\psi_j$ holds satisfying the base case $\neg inLoop_i \vee \psi_i$. This entails the QF-EU$\mathcal{D}$ formula for $\zeta \mathbf{R}\psi$. Conversely, if $\ell \leq j \leq k$ and $\neg\varphi_k$ holds, with $\varphi = \zeta \mathbf{U}\psi$, then $\neg\psi_j$ holds such that $\ell \leq j \leq k$. Then, in the propositional encoding, $inLoop_j$ is satisfied, all auxiliary formulae $\langle \mathbf{G}\psi\rangle_i$, such that $j \leq i \leq k$, do not hold because of the recursive definition. Finally, the formula $loopEx \Rightarrow ((\zeta \mathbf{R}\psi)_k \Leftarrow \langle \mathbf{G}\psi\rangle_k)$ is satisfied.

Complexity analysis

Let us compare the Boolean encoding with the QF-EU$\mathcal{D}$. Let ϕ be the PLTLB formula and $|\phi|$ be its dimension. If $m = \mathcal{O}(|\phi|)$ is the total number of subformulae and n is the total number of temporal operators $\mathbf{U}$ and $\mathbf{R}$ occurring in ϕ, then the Boolean encoding requires $(2k+3) + (k+2)m + (k+1)n$ fresh propositional variables. The QF-EU$\mathcal{D}$ encoding requires only $n+1$ nonnegative integer variables ($\boldsymbol{\ell}$ and $\boldsymbol{j}_\psi$) and m unary predicates (one for each subformula).

As already explained in Section 2.3, Nelson-Oppen theorem [25] provides the upper bound of satisfiability problem when different theories are combined. Since $\mathcal{D} = (\mathbb{Z}, <)$ is a consistent, stably infinite theory, non convex theory then, by the Nelson-Oppen Theorem, the bounded satisfiability of a LTLB formula ϕ can be solved in exponential time. However, the NP-completeness for satisfiability problem of the combined theory QF-EU and IDL, which is enough to encode the bounded satisfiability, is the same as for SAT.

3.3.5. Linear Encoding of LTL(FO) for SMT

LTL(**FO**) temporal encoding is defined by the linear encoding for LTLB of Section 3.3.1 except for the encoding of semantics of terms, uninterpreted functions and relations.

An *arithmetic formula function*, i.e. an uninterpreted function $\boldsymbol{\tau} : \mathbb{N} \to D$, is associated with each term of ϕ. Let τ be such a subterm, then the arithmetic formula function associated with it (denoted by the same name but in written in bold face), is recursively defined with respect to the sequence of valuations $\widehat{\sigma}_k$.

Also in this case we shall write improperly $\boldsymbol{x}_i$ instead of $\boldsymbol{x}(i)$.

τ	$0 \leq i \leq k$
x	$\boldsymbol{x}_i = \widehat{\sigma}_k(i, x)$
$\mathrm{X}\beta$	$\boldsymbol{\tau}_i = \boldsymbol{\beta}_{i+1}$

The conjunction of all the arithmetic constraints for all the subterms α of ϕ constitutes the formula $|ArithConstraints|_k$.

As defined in Section 3, semantics of LTL(**FO**) is defined with respect to an interpretation (model) $(\mathcal{M}, \varepsilon)$, where $\mathcal{M}$ is a structure $\mathcal{M} = (D, \mathcal{I})$ and ε an environment. When the interpretation $\mathcal{I}$ is partial, i.e., some of symbols of $(\mathcal{F}, \mathcal{R})$ are not defined, we can exploit the theory of Uninterpreted Function to encode the bounded satisfiability of LTL(**FO**). When functions and relations in $(\mathcal{F}, \mathcal{R})$ have global semantics they are simply instantiated in the formula. We introduce an arithmetic formula function $\tau : \mathbb{N} \to D$ for all uninterpreted function $f \in \mathcal{F}$ in ϕ with global interpretation such that $f : D^n \to D$. Values for τ are recursively defined as:

τ	$0 \leq i \leq k$
$f(\tau_1, \ldots, \tau_n)$	$\tau_i = f(\boldsymbol{\tau_i^1}, \ldots, \boldsymbol{\tau_i^n})$

Analogously, we introduce an uninterpreted predicate $\alpha : \mathbb{N} \to \{true, false\}$ for all uninterpreted relation $R \in \mathcal{R}$ in ϕ, with global interpretation, such that $R : D^n \to D$.

α	$0 \leq i \leq k+1$
$R(\tau_1, \ldots, \tau_n)$	$\alpha_i \Leftrightarrow R(\boldsymbol{\tau_i^1}, \ldots, \boldsymbol{\tau_i^n})$

When local semantics is adopted, functions and relations may have different value when evaluated at different position in $[0, k]$. We introduce an arithmetic formula function $\tau : \mathbb{N} \to D$ for all uninterpreted function $f \in \mathcal{F}$ occurring in ϕ such that $f : D^n \times \mathbb{N} \to D$, where the $n+1$-th value represents the instant of evaluation of f. Values for τ are recursively defined as:

τ	$0 \leq i \leq k$
$f(\tau_1, \ldots, \tau_n)$	$\tau_i = f(\boldsymbol{\tau_i^1}, \ldots, \boldsymbol{\tau_i^n}, i)$

where f is an n-function over D. Analogously, we introduce an uninterpreted predicate $\alpha : \mathbb{N} \to \{true, false\}$ for all uninterpreted relation $R \in \mathcal{R}$ occurring in ϕ such that $R : D^n \times \mathbb{N} \to D$, where the $n+1$-th value represents the instant of evaluation of f. Values for α are recursively defined as:

α	$0 \leq i \leq k+1$
$R(\tau_1, \ldots, \tau_n)$	$\alpha_i \Leftrightarrow R(\boldsymbol{\tau_i^1}, \ldots, \boldsymbol{\tau_i^n}, i)$

where R is an n-relation over D.

Bounded satisfiability for the existential fragment of LTL(**FO**) can be reduced to satisfiability of formulae of EUF$\cup L$, provided that the union of the two theories is still decidable. Let ϕ be a LTL(**FO**) formula

and V_g be the set of global variables occurring in ϕ such that $V_g^{\exists}, V_g^{free}$ partition V_g. In particular, the set $V_g^{\exists}$ contains global variables which are existentially quantified in ϕ and V_g^{free} is the set of global variables which do not have quantification, i.e., $V_g^{free} \subseteq free(\phi)$. For each variable in $y \in V_g^{free}$ we introduce an uninterpreted (0-arity) function $\boldsymbol{y} \in D$ such that $\boldsymbol{y} = \varepsilon(y)$. Quantified formulae are encoded by introducing for each subformula $\exists y\ \phi$ such that $y \in V_g^{\exists}$ a formula predicate $\varphi : \mathbb{N} \to \{true, false\}$ and a variable $\boldsymbol{y} : \mathbb{N} \to D$:

φ	$0 \leq i \leq k$
$\exists y\ \phi$	$\boldsymbol{\varphi}_i \Leftrightarrow \boldsymbol{\phi}_i\vert_{y \leftarrow \boldsymbol{y}_i}$

where $\boldsymbol{\phi}_i\vert_{y \leftarrow \boldsymbol{y}_i}$ represents variable $\boldsymbol{\phi}$ of subformula ϕ where the occurrence of y is substituted with variable $\boldsymbol{y}$ at the same position i.

Complexity analysis

Let ϕ be a LTL(**FO**) formula, h be the number of variables and m be the total number of temporal operators occurring in ϕ. Complexity analysis for temporal operators is the same as the one of LTL; we need $n+1$ nonnegative integer variables and m unary predicates to encode all temporal subformulae occurring into ϕ and periodicity constraints. Complexity of the first order part is defined with respect to:

- number N_f of functions $f(\tau_1, \ldots, \tau_n)$ occurring in ϕ,
- number N_R of relations $R(\tau_1, \ldots, \tau_n)$ occurring in ϕ and
- number $N_{\exists}$ of quantified variables of the form $\exists x\ \phi'$.

The total number T of arithmetic formula functions required to encode all terms involved in formula ϕ is determined by the number of functions $f(\tau_1, \ldots, \tau_n)$ and terms $\mathrm{X}^i x$ for $0 \leq i \leq \lceil \phi \rceil$. Therefore, $T \leq k(N_f + h\lceil \phi \rceil)$. Relations $R(\tau_1, \ldots, \tau_n)$ require kN_R predicate functions and quantified subformulae are $\leq kN_{\exists}m$. Therefore, the total number of predicate formulae is $P \leq k(N_R + N_{\exists}m)$.

CLTLB(L) temporal encoding is defined by the linear encoding for LTL(**FO**) of Section 3.3.1. In particular, languages considered in the forthcoming sections and in Chapter 4 satisfy conditions needed to combine correctly the theories of equality and uninterpreted functions. PA, IPC* and all its fragments and structure $(D, <, =)$ make the union EUF$\cup$ $\mathcal{D}$ consistent. The k-bounded satisfiability problem for a CLTLB(L) formula is NP-complete when L is DL and quantifier-free PA, PTIME when L is RDL. The NP-hardness of CLTLB(DL) follows by reducing SAT to the satisfiability of a QF-UFIDL formula.

3.4. Extending CLTL language

In this section, we provide an extension of CLTL language. Language CLTL is enriched with past-time temporal modalities over CLTL formulae ($\mathbf{Y}$, $\mathbf{S}$) and over a.t.t.'s (Y). Although both $\mathbf{Y}$, $\mathbf{S}$ are already considered by Demri and D'Souza in [6] (Section 9.3) there are no results concerning initial and global equivalence between the two languages. It is not immediate, in fact, to prove the equivalence since it is not a direct consequence of results shown in [6]. Moreover, authors of [6] do not consider past-time modalities over a.t.t.'s like, for instance, $\mathrm{Y}x = 0$. Even though a formula like $\mathrm{Y}x = 0$ can be intuitively replaced by an equivalent formula $\mathbf{Y}(x = 0)$, this equivalence may no longer hold at instant 0. The effect of the modality Y acting on the variable x represents an action of initialization of values defining the model before the origin. Similarly to LTL, we will prove the equivalence between CLTL and CLTL with past-time modalities, which we denote CLTLB hereafter.

Semantics of CLTLB formulae is defined recursively in a similar way to semantics of LTL($\mathbf{FO}$). Given a structure (S, π, L), the sequence $\pi \in S^\omega$ is now $s_{\lfloor\phi\rfloor} s_{\lfloor\phi\rfloor+1} \dots s_{-1} s_0 s_1 \dots$ where the prefix $s_{\lfloor\phi\rfloor} s_{\lfloor\phi\rfloor+1} \dots s_{-1}$ is needed to correctly evaluate formulae using terms referring to positions before the origin. Values for terms are defined in the same way as for LTL($\mathbf{FO}$) at beginning Section 3 by σ; the definition now includes the semantics for the operator Y over terms.

$$[\mathrm{Y}\tau]_{(\pi,i)} = [\tau]_{(\pi,i-1)}$$

Past-time temporal modalities "yesterday" $\mathbf{Y}$ and "since" $\mathbf{S}$ are defiend in the standard way:

$$\pi, i \models \mathbf{Y}\phi \Leftrightarrow \pi, i-1 \models \phi \wedge i > 0$$
$$\pi, i \models \phi \mathbf{S} \psi \overset{\text{def}}{\Leftrightarrow} \exists 0 \leq j \leq i \; \pi, j \models \psi \text{ and } \forall j < n \leq i \; \pi, n \models \phi$$

The use of temporal modality Y requires π to be isomorphic to $(\mathbb{Z}, <)$. In particular, given a structure (S, π, L), the map L is a total function for all $i \geq \lfloor\phi\rfloor$. It is worth noticing that values of propositional atoms in the prefix $s_{\lfloor\phi\rfloor} s_{\lfloor\phi\rfloor-1} \dots s_{-1}$ does not affect valuation of formulae from the origin 0. In fact, valuation of future formulae, involving temporal modalities $\mathbf{X}$ and $\mathbf{U}$, at position i is defined only by the suffix of π from i onward, for all $i \geq 0$. Past formulae, involving temporal modalities $\mathbf{X}$ and $\mathbf{U}$, are evaluated at position i by considering only truth value of subformulae in the segment $[0, i]$, for all $i \geq 0$.

3.5. Removing the "past" and initial equivalence

3.5.1. Initial equivalence

Let $\mathcal{D} = \langle D, \mathcal{R}, \mathcal{F} \rangle$ be a strucutre where D is a specific domain of interpretation for variables and constants, $\mathcal{R}$ is a family of relations on elements of D which is closed under complement and $\mathcal{F}$ is a family of functions on elements of D (the interpretetion $\mathcal{I}$ is understood). Let L be the first-order language defied by $\mathcal{D}$. By exploiting well-known properties of PLTLB, we prove the equivalence of CLTLB($\mathcal{D}$) to CLTL($\mathcal{D}$) for a quantifier-free constraint system $\mathcal{D}$, with respect to *initial* equivalence.

In [47], Gabbay et al. show that any PLTLB formula is initially equivalent to a PLTL formula, while the two logics are not globally equivalent (see also Schnoebelen [80] for details); the definition of these two notions are given in Section 2.7.1. In order to extend this result to the constrained case, we need to introduce new temporal operators. CLTL(L) and CLTLB(L), as we defined in Section 3 and 3.4, includes the "non-strict" until (respespectively since) operator, in which formula $\phi \mathbf{U} \psi$ (respectively $\phi \mathbf{S} \psi$) holds at instant i when ψ holds at i, and only if ϕ holds starting from i. The "strict" version of until $\mathbf{U}^{>}$, instead, does not require this:

$$\pi, i \models \phi \mathbf{U}^{>} \psi \stackrel{\text{def}}{\Leftrightarrow} \exists j > i \; \pi, j \models \psi \text{ and } \forall i < n < j \; \pi, n \models \phi$$

and similarly for the strict since $\mathbf{S}^{>}$:

$$\pi, i \models \phi \mathbf{S}^{>} \psi \stackrel{\text{def}}{\Leftrightarrow} \exists 0 \leq j < i \; \pi, j \models \psi \text{ and } \forall j < n < i \; \pi, n \models \phi.$$

It is well known that the following global equivalences hold for any ϕ,ψ:

$$\mathbf{X}\phi \equiv_{g\bot} \mathbf{U}^{>}\phi, \quad \phi \mathbf{U} \psi \equiv_g \psi \vee (\phi \wedge \phi \mathbf{U}^{>} \psi);$$
$$\mathbf{Y}\phi \equiv_{g\bot} \mathbf{S}^{>}\phi, \quad \phi \mathbf{S} \psi \equiv_g \psi \vee (\phi \wedge \phi \mathbf{S}^{>} \psi).$$

Using the previous equivalences, Gabbay [81] proved that any PLTLB formula is globally equivalent to a separated PLTLB formula, i.e., a Boolean combination of formulae containing either $\mathbf{U}^{>}$ ($\mathbf{U}^{>}$-formulae) or $\mathbf{S}^{>}$ ($\mathbf{S}^{>}$-formulae), but not both. Since this theorem preserves all semantic properties, i.e., it is actually a rewriting syntactic procedure over formulae, it extends also to the case of CLTLB(L), provided that each arithmetic constraint is represented by a propositional letter. In particular, a.t.t.'s $\mathrm{X}x/\mathrm{Y}x$ are not rewritten using strict-until/-since operators, but are considered as is, since their semantics depends on the underlying sequence σ as defined before. Then, we need to show that $\mathbf{S}^{>}$-formulae can be translated into *initially* equivalent $\mathbf{U}^{>}$-formulae. More precisely, we prove the following:

Theorem 62. *Any CLTLB(L) formula is initially equivalent to a CLTL(L) formula, while the two logics are not globally equivalent.*

Proof sketch. We first prove that CLTL(L) is not globally equivalent to CLTLB(L) by providing a counterexample. Formula $\top \mathbf{S} A$, where $A \in AP$, was shown in [45] to have no globally equivalent PLTL formula. Now, suppose ϕ is a CLTL(L) formula globally equivalent to CLTLB(L) formula $\top \mathbf{S} A$. Then, for the above reason, it should constrain at least one of its arithmetic variables, by a non-trivial arithmetic formula. Since $\top \mathbf{S} A$ does not constrain any arithmetic variables, some of its models cannot be models of ϕ.

To prove the initial equivalence we suppose each formula is written using only $\mathbf{U}^{>}$ and $\mathbf{S}^{>}$ operators, using the equivalences above. From Gabbay's Separation Theorem such a formula can be rewritten to a separated CLTLB(L) formula which is a Boolean combination of $\mathbf{S}^{>}$- and $\mathbf{U}^{>}$-formulae. The proof is concluded by noticing that any $\mathbf{S}^{>}$-formula is trivially initially equivalent to false. □

3.5.2. Removing the past operator Y

The CLTLB(L) language defined so far admits the use of the "previous" operator Y on arithmetic terms. In this section we focus on CLTLB language whose atomic formulae are only defined by the language L and which does not involve atomic propositions. We prove that CLTLB using only the future fragment of the language defining the a.t.t.'s is equivalent CLTLB using both the modalities X and Y. In particular, we define a syntactic translation function p such that $\sigma, 0 \models \phi \Leftrightarrow \sigma, \lfloor \phi \rfloor \models p(\phi)$.

Let X^i (resp. Y^i) represent the nesting of X (resp. Y) i times and let p : CLTLB(L) $\rightarrow$ CLTLB(L) be the rewriting function defined recursively as:

- $p(\mathrm{X}^i x) \stackrel{\text{def}}{\equiv} \mathrm{X}^{i-\lfloor \phi \rfloor} x$
- $p(\mathrm{Y}^i x) \stackrel{\text{def}}{\equiv} \mathrm{Y}^{i+\lfloor \phi \rfloor} x$
- $p(R(\tau_1, \ldots, \tau_n)) \stackrel{\text{def}}{\equiv} R(p(\tau_1), \ldots, p(\tau_n))$
- $p(\neg \phi) \stackrel{\text{def}}{\equiv} \neg p(\phi)$
- $p(\phi \wedge \psi) \stackrel{\text{def}}{\equiv} p(\phi) \wedge p(\phi)$
- $p(\mathbf{X}\phi) \stackrel{\text{def}}{\equiv} \mathbf{X} p(\phi)$
- $p(\mathbf{Y}\phi) \stackrel{\text{def}}{\equiv} \mathbf{Y} p(\phi)$

- $p(\phi \mathbf{U} \psi) \stackrel{\text{def}}{\equiv} p(\phi) \mathbf{U} p(\psi)$
- $p(\phi \mathbf{S} \psi) \stackrel{\text{def}}{\equiv} p(\phi) \mathbf{S} p(\psi)$

Given a CLTLB(L) formula ϕ it is easy to see that Y does not occur in $p(\phi)$ since the following equivalences hold $\mathrm{X}^i = \mathrm{Y}^{-i}$ and $\mathrm{X}^{-i} = \mathrm{Y}^i$ (e.g., $\mathrm{X}^{-3} = \mathrm{Y}^3$). The equisatisfiability of formulae is guaranteed by moving the origin of ϕ by $-\lfloor \phi \rfloor$ instants in the past. By shifting the sequence $\pi = s_{\lfloor \phi \rfloor} s_{\lfloor \phi \rfloor + 1} \ldots s_{-1} s_0 s_1 \cdots \in S^\omega$ we obtain a new model of the form $\pi' = s'_0 s'_1 \ldots s'_{-1-\lfloor \phi \rfloor} s'_{-\lfloor \phi \rfloor} s'_1 \cdots \in S^\omega$ which is the same sequence as π over a new ordering; i.e., $s_i = s'_{i-\lfloor \phi \rfloor}$. Since only X occurs in $p(\phi)$, then models π for CLTLB(L) formulae without Y are isomorphic to $(\mathbb{N}, <)$.

Theorem 63. *Let ϕ be a CLTLB(L) formula, then $\sigma, 0 \models \phi \Leftrightarrow \sigma, \lfloor \phi \rfloor \models p(\phi)$.*

Proof. Let $s = \lfloor \phi \rfloor$. We show that for all $i \geq 0$, $\sigma, i \models \phi \Leftrightarrow \sigma, i+s \models p(\phi)$ by induction on the structure of the formula ϕ.

The **base case** of the induction is given on the atomic formulae $\phi = R(\tau_1 \ldots \tau_n)$. Since $\sigma, i \models_{\mathcal{D}} \phi \Leftrightarrow R([x_{\tau_1}]_{(\sigma, i+|\tau_1|)}, \ldots, [x_{\tau_n}]_{(\sigma, i+|\tau_n|)})$, by shifting the instant i of s the satisfaction relation is $\sigma, i \models_{\mathcal{D}} \phi \Leftrightarrow R([x_{\tau_1}]_{(\sigma, i+s+|\tau_1|-s)}, \ldots, [x_{\tau_n}]_{(\sigma, i+s+|\tau_n|-s)})$. Then, we can equivalently write $\sigma, i \models_{\mathcal{D}} \phi \Leftrightarrow R([x_{\tau_1}]_{(\sigma, i+s+|p(\tau_1)|)}, \ldots, [x_{\tau_n}]_{(\sigma, i+s+|p(\tau_n)|)})$ that is $\sigma, i+s \models R(p(\tau_1), \ldots, p(\tau_n))$ and $\sigma, i+s \models p(R(\tau_1, \ldots, \tau_n))$. In fact, if $\tau = \mathrm{X}^i x$ then $p(\tau) = \mathrm{X}^{i-s} x$ and $|p(\tau)| = |\tau| - s$. If $\tau = \mathrm{Y}^i x$ then $p(\tau) = \mathrm{Y}^{i+s} x$ and $|p(\tau)| = -(i+s) = |\tau| - s$, since $|\tau| = -i$.
Inductive step.

- If $\phi = \neg\psi$ then $\sigma, i \models \phi \Leftrightarrow \sigma, i \not\models \psi$. By inductive hypothesis, this is equivalent to $\sigma, i+s \not\models p(\psi)$, i.e. $\sigma, i+s \models p(\phi)$, as $p(\phi) = \neg p(\psi)$.
- If $\phi = \psi_1 \wedge \psi_2$ then $\sigma, i \models \phi \Leftrightarrow \sigma, i \models \psi_1$ and $\sigma, i \models \psi_2$. By inductive hypothesis, this is equivalent to $\sigma, i+s \models p(\psi_1)$ and $\sigma, i+s \models p(\psi_2)$, i.e. $\sigma, i+s \models p(\psi_1) \wedge p(\psi_2)$, and $\sigma, i+s \models p(\phi)$.
- If $\phi = \mathbf{X}\psi$ then $\sigma, i \models \phi \Leftrightarrow \sigma, i+1 \models \psi$. By inductive hypothesis, this is equivalent to $\sigma, i+1+s \models p(\psi)$, i.e., $\sigma, i+s \models \mathbf{X} p(\psi)$, which corresponds to $\sigma, i+s \models p(\phi)$.
- If $\phi = \mathbf{Y}\psi$ then $\sigma, i \models \phi \Leftrightarrow \sigma, i-1 \models \psi$. By inductive hypothesis, this is the same as $\sigma, i-1+s \models p(\psi)$, i.e., $\sigma, i+s \models \mathbf{Y} p(\psi)$, and $\sigma, i+s \models p(\phi)$, as $p(\phi) = \mathbf{Y} p(\psi)$.
- If $\phi = \psi_1 \mathbf{U} \psi_2$ then $\sigma, i \models \phi$ iff there exists $j \geq i$ s.t. $\sigma, j \models \psi_2$ and $\sigma, n \models \psi_1$ forall $i \leq n < j$, that is, by inductive hypothesis,

$\sigma, j+s \models p(\psi_2)$ and $\sigma, n \models p(\psi_1)$ forall $i+s \leq n < j+s$, which in turn is equivalent to $\sigma, i+s \models p(\psi_1)\mathbf{U}p(\psi_2)$ and $\sigma, i+s \models p(\phi)$.

- If $\phi = \psi_1 \mathbf{S} \psi_2$ then $\sigma, i \models \phi$ iff there exists $0 \leq j \leq i$ s.t. $\sigma, j \models \psi_2$ and $\sigma, n \models \psi_1$ forall $j < n \leq i$, that is, by inductive hypothesis $\sigma, j+s \models p(\psi_2)$ and $\sigma, n \models p(\psi_1)$ forall $j+s < n \leq i+s$, which is equivalent to $\sigma, i+s \models p(\psi_1)\mathbf{S}p(\psi_2)$ and $\sigma, i+s \models p(\phi)$.

Finally, $\sigma, 0 \models \phi \Leftrightarrow \sigma, s \models p(\phi)$ by taking $i = 0$. □

3.5.3. Removing the language AP

According to the definition given in Section 3, CLTL(L) is the language CLTL where atomic formulae belong to the language L. When L contains also a set AP of atomic propositions, atomic formulae α are propositional atoms or relations over terms $R(\tau_1, \ldots, \tau_n)$. If we are dealing with the CLTL without past-time modalities Y over variables we can encode propositional atoms to variables ranging over $\{0, 1\}$. Any positive occurrence of an atomic proposition $p \in AP$ in a CLTL formula can be replaced by an equality relation of the form $x_p = 1$. Then, a formula of CLTL($L \cup \mathbf{AP}$) can be easily rewritten into a formula of CLTL(L) preseving the equivalence between them. In this section, we prove that the language **AP** can be removed also in case of generic CLTLB($L \cup \mathbf{AP}$) formulae. We will define a rewriting function r such that $\sigma, 0 \models \phi$ if, and only if, $\theta, 0 \models r(\phi) \wedge \psi$ where θ is the same as σ' except for new fresh variables representing atomic proposition before the origin and ψ is a formula restricting values of the new fresh variables in $\{0, 1\}$.

Let us suppose $AP = \{p_1, \ldots, p_n\}$ to be an ordered set of finite propositional atoms, ϕ be a CLTLB($L \cup \mathbf{AP}$) formula and V be the set of variables occurring in ϕ. Let us define the set $V_{AP} = \{x_{p_1}, \ldots, x_{p_n}\}$ be the set of variables representing propositional atoms of AP such that $V \cap V_{AP} = \emptyset$. Let r : CLTLB($L \cup \mathbf{AP}$) $\rightarrow$ CLTLB(L) be the rewriting function defined recursively as:

- $r(p_i) \stackrel{\text{def}}{\equiv} (x_{p_i} = 1)$
- $r(R(\tau_1, \ldots, \tau_n)) \stackrel{\text{def}}{\equiv} R(r(\tau_1), \ldots, r(\tau_n))$
- $r(\neg\phi) \stackrel{\text{def}}{\equiv} \neg r(\phi)$
- $r(\phi \wedge \psi) \stackrel{\text{def}}{\equiv} r(\phi) \wedge r(\phi)$
- $r(\mathbf{X}\phi) \stackrel{\text{def}}{\equiv} \mathbf{X}r(\phi)$

- $r(\mathbf{Y}\phi) \stackrel{\text{def}}{\equiv} \mathbf{Y}r(\phi)$
- $r(\phi\mathbf{U}\psi) \stackrel{\text{def}}{\equiv} r(\phi)\mathbf{U}r(\psi)$
- $r(\phi\mathbf{S}\psi) \stackrel{\text{def}}{\equiv} r(\phi)\mathbf{S}r(\psi)$

Removing propositional atoms is a syntactic rewriting which acts on formulae. Therefore, we can provide a syntactic rewriting function r_{model} which acts on models σ of CLTLB($L \cup \mathbf{AP}$) formulae. Let $\theta = r_{model}(\sigma)$ be a sequence $(D^{|V|+n})^\omega$ of valuation over for variables in $V \cup V_{AP}$; i.e., $\theta : \mathbb{Z} \times V \cup \{x_{p_1}, \ldots, x_{p_n}\} \to D$ is the rewriting of σ defined as follows:

$$\theta(i, x) = \sigma'(i, x) \text{ for all } x \in V, \text{ for all } \lfloor \phi \rfloor \leq i$$
$$\theta(i, x_{p_j}) = \begin{cases} 1 & p_j \in \sigma''(i) \\ 0 & p_j \notin \sigma''(i) \end{cases} \text{ for all } j \in [1, n] \text{ and for all } i \geq 0$$

Theorem 64. *Let ϕ be a* CLTLB($L \cup \mathbf{AP}$) *formula where $AP = \{p_1, \ldots, p_n\}$. Then, $\sigma, 0 \models \phi$ if, and only if,*

$$r_{model}(\sigma), 0 \models \left(r(\phi) \wedge \mathbf{G}(\bigwedge_{i=1}^{n} (x_{p_i} = 1) \vee (x_{p_i} = 0)) \right).$$

Proof. We show that for all $i \geq 0$, $\sigma, i \models \phi \Leftrightarrow r_{model}(\sigma), i \models (r(\phi) \wedge \mathbf{G}(\bigwedge_{i=1}^{n}(x_{p_i} = 1) \vee (x_{p_i} = 0)))$. It follows immediately $\sigma, i \models \phi \Leftrightarrow r_{model}(\sigma), i \models r(\phi)$ and $r_{model}(\sigma), i \models \mathbf{G}(\bigwedge_{i=1}^{n}(x_{p_i} = 1) \vee (x_{p_i} = 0))$. Hereafter, we write θ instead of $r_{model}(\sigma)$.

First, we prove by induction the left subformula $\sigma, i \models \phi \Leftrightarrow \theta, i \models r(\phi)$. The **base case** of is given on propositional atoms. Since $\sigma, i \models p_j \Leftrightarrow p_j \in \sigma''(i)$ and by definition of $\theta = r_{model}(\sigma)$, we can conclude that $\theta(i, x_{p_j}) = 1$. By definition $\theta, i \models (x_{p_j} = 1) \Leftrightarrow \theta(i, x_{p_j}) = 1$; hence, $\theta, i \models r(p_j)$.

Inductive step.

- If $\phi = \neg\psi$ then $\sigma, i \models \phi \Leftrightarrow \sigma, i \not\models \psi$. By inductive hypothesis, this is equivalent to $\theta, i \not\models r(\psi)$, i.e. $\theta, i \models r(\phi)$, as $r(\phi) = \neg r(\psi)$.
- If $\phi = \psi_1 \wedge \psi_2$ then $\sigma, i \models \phi \Leftrightarrow \sigma, i \models \psi_1$ and $\sigma, i \models \psi_2$. By inductive hypothesis, this is equivalent to $\theta, i \models r(\psi_1)$ and $\theta, i \models r(\psi_2)$, i.e. $\theta, i \models r(\psi_1) \wedge r(\psi_2)$, and $\theta, i \models r(\phi)$.
- If $\phi = \mathbf{X}\psi$ then $\sigma, i \models \phi \Leftrightarrow \sigma, i + 1 \models \psi$. By inductive hypothesis, this is equivalent to $\theta, i + 1 \models r(\psi)$, i.e., $\theta, i \models \mathbf{X}r(\psi)$, which corresponds to $\theta, i \models r(\phi)$.

- If $\phi = \mathbf{Y}\psi$ then $\sigma, i \models \phi \Leftrightarrow \sigma, i-1 \models \psi$ and $i \geq 0$. By inductive hypothesis, this is the same as $\theta, i-1 \models r(\psi)$ and $i \geq 0$, i.e., $\theta, i \models \mathbf{Y}r(\psi)$, and $\theta, i \models r(\phi)$, as $r(\phi) = \mathbf{Y}r(\psi)$.

- If $\phi = \psi_1 \mathbf{U} \psi_2$ then $\sigma, i \models \phi \Leftrightarrow$ there exists $j \geq i$ s.t. $\sigma, j \models \psi_2$ and $\sigma, n \models \psi_1$ forall $i \leq n < j$, that is, by inductive hypothesis, there exists $j \geq i$ s.t. $\theta, j \models r(\psi_2)$ and $\theta, n \models r(\psi_1)$ forall $i \leq n < j$, which in turn is equivalent to $\theta, i \models r(\psi_1)\mathbf{U}r(\psi_2)$ and $\theta, i \models r(\phi)$.

- If $\phi = \psi_1 \mathbf{S} \psi_2$ then $\sigma, i \models \phi \Leftrightarrow$ there exists $0 \leq j \leq i$ s.t. $\sigma, j \models \psi_2$ and $\sigma, n \models \psi_1$ forall $j < n \leq i$, that is, by inductive hypothesis there exists $0 \leq j \leq i$ s.t. $\theta, j \models r(\psi_2)$ and $\theta, n \models r(\psi_1)$ forall $j < n \leq i$, which is equivalent to $\theta, i \models r(\psi_1)\mathbf{S}r(\psi_2)$ and $\sigma, i \models r(\phi)$.

Finally, we prove the first part $\sigma, 0 \models \phi \Leftrightarrow \theta, 0 \models r(\phi)$, by taking $i = 0$.

Let us prove by induction the second part is $\theta, i \models \mathbf{G}(\bigwedge_{i=1}^{n}(x_{p_i} = 1) \vee (x_{p_i} = 0))$. The **base case** is $\theta, i \models \bigwedge_{i=1}^{n}(x_{p_i} = 1) \vee (x_{p_i} = 0)$ which holds for all $i \geq \lfloor \phi \rfloor$ by definition of $r_{model}(\sigma)$. The **inductive hypothesis** is $\mathbf{G}(\bigwedge_{i=1}^{n} x_{p_i} = 1) \vee (x_{p_i} = 0)$ holds at i for all $i \geq \lfloor \phi \rfloor$, i.e. $\theta, i \models \mathbf{G}(\bigwedge_{i=1}^{n}(x_{p_i} = 1) \vee (x_{p_i} = 0))$. Then, $\theta, i \models \bigwedge_{i=1}^{n}(x_{p_i} = 1) \vee (x_{p_i} = 0) \wedge \mathbf{XG}(\bigwedge_{i=1}^{n}(x_{p_i} = 1) \vee (x_{p_i} = 0))$ is equivalent to $\theta, i \models (\bigwedge_{i=1}^{n}(x_{p_i} = 1) \vee (x_{p_i} = 0))$ and $\theta, i \models \mathbf{XG}(\bigwedge_{i=1}^{n}(x_{p_i} = 1) \vee (x_{p_i} = 0))$. The first conjunct follows from the base case. The second one $\theta, i \models \mathbf{XG}(\bigwedge_{i=1}^{n}(x_{p_i} = 1) \vee (x_{p_i} = 0))$ is equivalent $\theta, i+1 \models \mathbf{G}(\bigwedge_{i=1}^{n}(x_{p_i} = 1) \vee (x_{p_i} = 0))$ and it holds by inductive hypothesis. Therefore, by taking $i = 0$ we conclude that $\theta, 0 \models \mathbf{G}(\bigwedge_{i=1}^{n}(x_{p_i} = 1) \vee (x_{p_i} = 0))$.

□

3.5.4. General equivalence result

Results proved in previous Sections help us to conclude the equivalence between the language CLTLB($L \cup \mathbf{AP}$) and CLTL(L) with only X.

Theorem 65. *Let ϕ be a* CLTLB($L \cup \mathbf{AP}$) *formula. Then, there exists an initial equivalent* CLTL(L) *formula ϕ' without* Y.

Proof. From Theorems 63, 64 we can define ϕ' to be the composition of translation functions r, p, i.e., $\phi' = p(r(\phi))$. Formula ϕ' belongs to CLTLB(L) and its temporal modalities are only $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{U}$, $\mathbf{S}$ and X. Then, from Theorem 62, formula ϕ' is initially equivalent to a CLTL(L) formula (with only X). □

3.6. Completeness of the Bounded Satisfiability Problem for CLTL

In this section, we study the existence of a completeness threshold for the satisfiability problem of CLTLB(L) formulae when the language L of atomic formulae is IPC* or defined by fragments like $(D, =, <)$ where $D = \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$. When previous languages are considered, partial models $\widehat{\sigma}_k$ suffices to deduce satisfiability of formulae over complete models. Therefore, bounded satisfiability problem becomes an alternative method to automata-based approach for solving satisfiability problem of CLTLB.

3.6.1. Bounded Satisfiability Problem for CLTL

Bounded Satisfiability Problem is defined by considering bounded symbolic models of CLTLB(L) formulae (Y and the language **AP** can always be removed). The definition of the problem given in Section 3.2 specializes in the case of CLTLB since all functions have global interpretation inherited from the language L.

Since the length k is fixed, the satisfiability of CLTLB(L) formulae over bounded models is, in general, not complete. Even if the automaton $\mathcal{A}_\phi$ has no accepting runs of length k, it might have one of length $k' > k$. The completeness property is defined as follows:

Definition 66. *A CLTLB(L) formula ϕ has the completeness property if there is a constant $K \in \mathbb{N}$, depending on ϕ, such that ϕ is satisfiable if, and only if, ϕ is K-bounded satisfiable.*

Hence, if ϕ has the completeness property for a value K and there is no finite model σ_K of ϕ, then ϕ is unsatisfiable. A language CLTLB(L) has the completeness property when all formulae $\phi \in$ CLTLB(L) have completeness property. Later, in Section 3.6, we will show that for some fragments of CLTLB which have completeness property, unsatisfiability over complete models is consequence of unsatisfiability over k-bounded models.

3.6.2. Completeness for IPC* and $(D, =, <)$

Informally, the idea for finding a completeness threshold for a CLTLB(L) formula is based on the fact that ultimately periodic symbolic models ρ of CLTLB(L) formulae admit an arithmetic model σ if condition C holds (see Proposition 53). Also, if a CLTLB(L) formula ϕ is satisfiable, then all ultimately periodic symbolic models ρ, such that $\rho, 0 \models_s \phi$, admit a model σ such that $\sigma \models \rho$. Completeness is a consequence of the existence

of a finite value c for which all initialized runs of $\mathcal{A}_\phi$, representing models for ϕ, of length greater than c visit at least one control state twice. Consequently, if a CLTLB(L) formula is not (boundedly) satisfiable by any ultimately periodic model of length less than or equal to the value $c+1$, then the formula is unsatisfiable. Let c be the length of the longest loop-free path of automaton $\mathcal{A}_\phi$, i.e., the recurrence diameter of $\mathcal{A}_\phi$. Definition of the automaton $\mathcal{A}_\phi$ is similar to definition adopted by Demri et al. in [6] but slightly different in definition of automaton $\mathcal{A}_s$ which is adapted to the encoding of the problem.

Automaton $\mathcal{A}_\phi$ for CLTLB(L) formulae

Let ϕ be a CLTL($\mathcal{D}$) formula, let $A \subseteq \mathcal{D}$ be the closure under negation of the set of arithmetic constraints occurring in ϕ, and let $\mathcal{A}_s = (\Sigma, Q', Q_0, \eta, F)$ be the symbolic Büchi automaton of ϕ. Alphabet Σ is the subset $valid(A) \subseteq \wp(A)$ such that for every atomic formula ξ of ϕ, $\beta \in valid(A)$ iff either ξ or $\neg\xi$ belongs to β. The closure of ϕ, denoted $cl(\phi)$, is the smallest set containing all subformulae of ϕ that is also closed under negation. An *atom* $\Gamma \subseteq cl(\phi)$ is a subset of formulae of $cl(\phi)$ that is maximally consistent, i.e., such that, for each formula ξ in ϕ, either $\xi \in \Gamma$ or $\neg\xi \in \Gamma$. It is worth noticing that an atom so defined might be unsatisfiable, i.e., there does not exist a valuation v' over a.t.t.'s such that $v' \models_{\mathcal{D}} \xi$, for all ξ in Γ, since $cl(\phi)$ is closed under negation. A pair (Γ_1, Γ_2) of atoms is *one-step temporally consistent* when Γ_1 and Γ_2 agree on the structure of temporal operators, that is:

- for every $\mathbf{X}\psi \in cl(\phi)$, then $\mathbf{X}\psi \in \Gamma_1 \Leftrightarrow \psi \in \Gamma_2$,
- for every $\mathbf{Y}\psi \in cl(\phi)$, then $\mathbf{Y}\psi \in \Gamma_2 \Leftrightarrow \psi \in \Gamma_1$,
- if $\psi_1 \mathbf{U} \psi_2 \in \Gamma_1$, then $\psi_2 \in \Gamma_1$ or ($\psi_1 \in \Gamma_1$ and $\psi_1 \mathbf{U} \psi_2 \in \Gamma_2$),
- if $\psi_1 \mathbf{S} \psi_2 \in \Gamma_2$, then $\psi_2 \in \Gamma_2$ or ($\psi_1 \in \Gamma_2$ and $\psi_1 \mathbf{S} \psi_2 \in \Gamma_1$).

The automaton $\mathcal{A}_s = (\Sigma, Q, Q_0, \eta, F)$ is then defined as follows:

- Q is the set of atoms;
- $Q_0 = \{\Gamma \in Q : \phi \in \Gamma, \mathbf{Y}\psi \notin \Gamma \text{ for all } \psi \in cl(\phi), \psi_1 \mathbf{S} \psi_2 \in \Gamma \text{ iff } \psi_2 \in \Gamma\}$;
- $\Gamma_1 \xrightarrow{\beta} \Gamma_2 \in \eta$ iff
  - $\beta = \Gamma_1 \cap A$,
  - (Γ_1, Γ_2) is one-step consistent;

- $F = \{F_1, \ldots, F_m\}$, where $F_i = \{\Gamma \in Q \mid \phi_i \mathbf{U} \psi_i \notin \Gamma \text{ or } \psi_i \in \Gamma\}$ and $\{\phi_1 \mathbf{U} \psi_1, \ldots, \phi_m \mathbf{U} \psi_m\}$ is the set of Until formulae occurring in $cl(\phi)$.

The automaton $\mathcal{A}_s$ is a generalized Büchi automaton. In order to provide the automaton $\mathcal{A}_\phi$, we shall translate $\mathcal{A}_s$ into a classical Büchi automaton, still preserving the language of accepted $\omega-$words. For ease of writing, we also denote this automaton with $\mathcal{A}_s$.

Let $\mathcal{A} = (SV(\phi), Q', Q'_0, \delta', F')$ be the automaton over the alphabet of symbolic valuations given by the intersection of automata $\mathcal{A}_\ell$ and $\mathcal{A}_{\neg C}$, as shown in [6]. Automaton $\mathcal{A}_\phi = (SV(\phi), Q'', Q''_0, \eta, F'')$ is defined as the product of $\mathcal{A}_s$ and $\mathcal{A}$, according to the standard intersection of Büchi automata but adapted in the definition of η:

- $Q'' = Q \times Q' \times \{0, 1, 2\}$;
- $Q''_0 = \{(\Gamma, q', 0) : \Gamma \in Q_0 \text{ and } q' \in Q'_0\}$;
- $(\Gamma_1, q', i) \xrightarrow{sv}, (\Gamma_2, p', j) \in \eta$ iff $\Gamma_1 \xrightarrow{\beta} \Gamma_2 \in \delta$, $q' \xrightarrow{sv} p' \in \delta'$ and $sv \models_s \xi$, for all $\xi \in \Gamma_1$, and:
  - if $i = 0$ then $j = 1$;
  - if $i = 1$ and $\Gamma_1 \in F$, then $j = 2$;
  - if $i = 2$ and $q' \in F'$, then $j = 0$;
  - otherwise $i = j$;
- $F'' = Q \times Q' \times \{0\}$.

We are ready to prove completeness of k-bounded satisfiability when the logic admits reduction of satisfiability to emptiness problem for Büchi automata. In order to define a procedure to decide the satisfiability for ϕ we reduce the problem to a finite amount of bounded satisfiability problem. Instead of defining the automaton $\mathcal{A}_\phi$ we define a CLTLB($L \cup \mathbf{AP}$) formula ϕ' such that if it is bounded satisfiable for some $k \in \mathbb{N}$ then the formula ϕ is satisfiable. In particular, we first define formulae ϕ_ℓ and $\phi_{\mathcal{A}_C}$ for automata $\mathcal{A}_\ell$ and $\mathcal{A}_C$ whose models are exactly words of the language recognized by the automata. Finally, ϕ' is the conjunction of the two formulae above, ϕ_ℓ and $\phi_{\mathcal{A}_C}$, with ϕ which is, then, checked for bounded satisfiability. It is worth noticing that atomic proposition encode only control states of automata. They are involved in ϕ' since we are looking for ultimately periodic runs of automata and they are not part of the original formula $\phi \in$ CLTLB(L) for which Theorem 53 holds.

Both automata $\mathcal{A}_\ell$ and $\mathcal{A}_C$ involved in the construction of $\mathcal{A}_\phi$ do not depend on the LTL temporal modalities appearing in ϕ, but only on the

language L, on the set of variables V and constants and, finally, on the length $\lceil\phi\rceil$ of symbolic valuations. The Büchi automaton automaton $\mathcal{A}_\ell$ is a tuple $(SV(\phi), Q_{\mathcal{A}_\ell}, Q_0, \delta_{\mathcal{A}_\ell}, F_{\mathcal{A}_\ell})$ such that $Q = Q_0 = F = SV(\phi)$ and its transition relation is such that $sv \xrightarrow{sv'} sv' \in \delta$ iff (sv, sv') are locally consistent. Its definition is exactly the same as [74]. Sequences of locally consistent symbolic valuations recognized by automaton $\mathcal{A}_\ell$ are also models of the formula

$$\phi_\ell := \mathbf{G}(\bigvee_1^m sv_i).$$

where $m = |SV(\phi)|$ is the cardinal of the set of symbolic valuations. In fact, since

(i) the encoding provided in Section 5 is such that the representation of formulae is not contradictory, i.e. two consecutive symbolic valuations are satisfiable if, and only if, they are locally consistent, and

(ii) the symbolic valuation sv satisfied in a position i is unique (because of the maximal consistency of symbolic valuations, Lemma 4 of [74]),

then $\mathbf{G}(\bigvee_1^m sv_i)$ represents exactly words of $\mathscr{L}(\mathcal{A}_\ell)$. According to Demri and D'Souza [6], the automaton $\mathcal{A}_C$ is not directly built from condition C. Instead, an automaton $\mathcal{A}_{\neg C}$, recognizing the complement language of $\mathscr{L}(\mathcal{A}_C)$, is built first. Then, the automaton $\mathcal{A}_C$ is obtained through Safra's method [82] for complementing Büchi automata. In general, $\mathcal{A}_C$ is defined by $\mathcal{A}_C = (SV(\phi), Q_{\mathcal{A}_C}, Q_0, \delta_{\mathcal{A}_C}, F_{\mathcal{A}_C})$. We use the reduction of the model-checking problem to the satisfiability problem, given in [52], to represent the automaton $\mathcal{A}_C$ by a CLTLB(L) formula. Let $\phi_{\mathcal{A}_C}$ be the formula representing $\mathcal{A}_C$:

$$\phi_{\mathcal{A}_C} := \bigvee_{q_i \in Q_0} q_i \ \wedge \ \mathbf{GF}(\bigvee_{q_i \in F} q_i)$$

$$\mathbf{G}\left(\bigvee_{i\in\{1,\ldots|Q|\}} (q_i \wedge \bigwedge_{j\in\{1,\ldots|Q|\}\setminus\{i\}} \neg q_j) \wedge \bigwedge_{i\in\{1,\ldots|Q|\}} (q_i \Rightarrow \bigvee_{(q_i,sv,q_j)\in\delta} (sv \wedge \mathbf{X}q_j))\right)$$

where $\mathbf{G}\psi$ and $\mathbf{F}\psi$ are shorthands for $\neg(T\mathbf{U}\neg\psi)$ and $\top\mathbf{U}\psi$.
We verify if the following formula is bounded satisfiable with respect to $k \in \mathbb{N}$:

$$I(\boldsymbol{x}_0) \ \wedge \ \phi \ \wedge \ \phi_{\mathcal{A}_C} \ \wedge \ \phi_\ell \tag{3.1}$$

where $I(\boldsymbol{x}_0)$ is a formula of L defining an initialization for variables and $\phi_\ell = \mathbf{G}(\bigvee_1^m sv_i)$ with $m = |SV(\phi)|$. If the formula (3.1) is unsatisfiable for all $k \in [1, c+1]$ then there does not exist any ultimately periodic symbolic model $\rho \in SV(\phi)^\omega$ such that $\rho, 0 \models_s \phi$ and such that there exists an arithmetic model σ with $\sigma \models \rho$. Hence, formula ϕ is unsatisfiable. Otherwise, there exists an ultimately periodic symbolic model ρ of length $k > 0$ which admits a model σ. From the bounded solution, we know exactly the model $\rho = \delta(\pi)^\omega$ and the bounded model σ_k. Then, the infinite model σ is defined from σ_k by iterating infinitely many times the sequence of symbolic valuations in π.

Before entering in details of proofs, we define some useful notations. Because IPC* involves binary relation, in order to simplify notation, we use infix notation and we write $\tau_1 \sim \tau_2$ instead of $R(\tau_1, \tau_2)$. Let us denote the formula (3.1) to be f. Let V be the set of variables of ϕ and $terms(\phi)$ be the set of all the arithmetic term $\mathrm{X}^i x$ such that $0 \leq i \leq \lceil \phi \rceil$ for all $x \in V$. Let $P = cl(f)$ the closure of f (here extended also to temporal operator $\mathbf{R}, \mathbf{T}$) and let $Q \subseteq P$ the set of propositional atoms of P. Let us suppose the formula (3.1) to be satisfiable. A model for the encoded formula $[f]_k$ of f is a pair $(\pi, \widehat{\sigma}_k)$ defined by the following two functions:

- a function $\widehat{\sigma}_k : \{0, \ldots, k + |\tau|\} \times \{\tau\} \to D$ for all τ in $terms(\phi)$;
- a function $\pi : \{0, \ldots, k+1\} \times \{P \cup SV(\phi)\} \to \{0, 1\}$.

It is worth noticing that we consider π a function defined over $\{P \cup SV(\phi)\}$ because the encoding $[f]_k$ is such that every atomic formula τ has a precise truth value for all position $\{P \cup SV(\phi)\}$. This means that if $[f]_k$ the sequence of symbolic valuation ρ such that $\widehat{\sigma}_k \models_s \rho$ is known. In the following, we shall say that the subformula φ holds in i, written φ_i, when

$$(\pi, \widehat{\sigma}_k), i \models_k \varphi$$

The value of the a.t.t. τ at position i, written τ_i, is $\widehat{\sigma}_k(i, \tau)$.

Lemma 67. *The projection of π of $[\phi_{\mathcal{A}_C}]_k$ models are possible initialized ultimately periodic run of automaton $\mathcal{A}_C$.*

Proof. If $[\phi_{\mathcal{A}_C}]_k$ is satisfiable, there exists an ultimately periodic model of symbolic valuations which is accepted by the automaton $\mathcal{A}_C$. The subformula $\mathbf{GF}()$ in $[\phi_{\mathcal{A}_C}]_k$ is satisfied when at least one accepting state of $F_{\mathcal{A}_C}$ is repeated infinitely often, witnessing the Büchi acceptance condition for $\mathcal{A}_C$. Moreover, only one state q_i is visited at each step because

the sequence of $q_0, \ldots, q_k$ satisfies

$$\mathbf{G}\left(\bigvee_{i\in\{1,\ldots|Q|\}}(q_i \wedge \bigwedge_{j\in\{1,\ldots|Q|\}\setminus\{i\}}\neg q_j)\right).$$

When moving from a state q_i to q_{i+1} the satisfied symbolic valuation sv witnessing

$$\mathbf{G}\left(\bigwedge_{i\in\{1,\ldots|Q|\}}(q_i \Rightarrow \bigvee_{(q_i,sv,q_j)\in\delta}(sv \wedge \mathbf{X}q_j))\right).$$

is unique, because of the maximal consistence of symbolic valuations. A symbolic valuation sv is defined at position i to be the set of all the atomic formulae $\tau_1 \sim \tau_2$ between elements $\tau_1 \in terms(\phi)$ and $\tau_2 \in terms(\phi)$. The truth value of atomic formulae $\xi = \tau_1 \sim \tau_2$ is defined by: $\xi_i \Leftrightarrow \tau_{1i} \sim \tau_{2i}$, i.e., $\widehat{\sigma}_k, i \models \xi \Leftrightarrow \widehat{\sigma}_k(i,\tau_1) \sim \widehat{\sigma}_k(i,\tau_2)$. Now, let us consider the conjunct $\theta = \mathbf{GF}\left(\bigvee_{q_i\in F} q_i\right)$, denoted $\mathbf{GF}(\gamma)$ with $\gamma = \bigvee_{q_i\in F} q_i$. By using duality of temporal operator, we shall exploit the following equivalence: $\theta = \mathbf{GF}(\gamma) = \neg(\top\mathbf{U}\neg(\top\mathbf{U}\gamma)) =\perp \mathbf{R}(\top\mathbf{U}\gamma)$. Now, let us denote ψ to be the subformula $\top\mathbf{U}\gamma$. The fixpoint representation for θ and ψ is defined by the $|TempConstraints|_k$ for $1 \leq i \leq k$:

$$\theta_i \Leftrightarrow \psi_i \wedge \theta_{i+1}$$
$$\psi_i \Leftrightarrow \gamma_i \vee \psi_{i+1}.$$

Let us denote $\bigvee_{i=1}^{k}(\ell = i)$ to be *loop*; the eventuality constraints are:

$$loop \Rightarrow \neg\theta_k \Rightarrow \ell \leq j_\psi \leq k \wedge \psi_{j_\psi}$$
$$loop \Rightarrow \psi_k \Rightarrow \ell \leq j_\gamma \leq k \wedge \gamma_{j_\gamma}.$$

Since θ is satisfiable, i.e., θ_0 holds, $\psi_i \wedge \theta_{i+1}$ holds for all $1 \leq i \leq k$. Then, we have $\ell \leq j_\gamma \leq k \wedge \gamma_{j_\gamma}$, since ψ_k holds. Consequently, ℓ must be true and the formula $\bigvee_{i=1}^{k}(\ell = i)$ defines the position of the loop. According to the $|LastStateConstraints|_k$ all the subformulae $\theta \in P$ are satisfied in $k+1$ iff they are satisfied at position ℓ: $\theta_{k+1} \Leftrightarrow \theta_\ell$ for all $\theta \in P$. Moreover, $|LoopConstraint|_k$ enforce $q_k \Leftrightarrow q_{\ell-1}$ for some $q \in Q$. Therefore, the projection of π on the alphabet over Q is the sequence $q_0 \ldots q_\ell \ldots q_k q_\ell$ of control state $q \in Q$ which is an ultimately periodic run of $\mathcal{A}_C$ of the form $q_0, \ldots, q_{\ell-1}(q_\ell, \ldots, q_k)^\omega$. The projection of π on the alphabet $SV(\phi)$ is an ultimately periodic word $\rho = sv_0, \ldots, sv_{\ell-1}(sv_\ell, \ldots, sv_k)^\omega$ of symbolic valuations such that $\widehat{\sigma}_k \models_k \rho$. Finally, the satisfiability of the formula γ

at position j_γ witnesses the repetition of at least one accepting state of $\mathcal{A}_C$ infinitely often. Hence, the sequence $q_0, \ldots, q_{\ell-1}(q_\ell, \ldots, q_k)^\omega$ is an accepting run of $\mathcal{A}_C$ and the sequence $sv_0, \ldots, sv_{\ell-1}(sv_\ell, \ldots, sv_k)^\omega$ is a word belonging to $\mathscr{L}(\mathcal{A}_C)$. Conversely, if the language of the automaton is not empty, then there exists an ultimately periodic word which belongs to $\mathscr{L}(\mathcal{A}_C)$. Since the number of control state of $\mathcal{A}_C$ is finite, the longest prefix which can be aperiodic is of length at most $|Q|$. This suffices to guarantee the existence of a maximal length for k. In other words, if the language of $\mathcal{A}_C$ is not empty, then there exists at least one ultimately periodic word of length bounded by $|Q|$. The formula $\phi_{\mathcal{A}_C}$ is satisfiable by a model given by the sequence of control state of $\mathcal{A}_C$ and the sequence of symbolic valuations constituting the word recognized by $\mathcal{A}_C$.

□

Next lemma requires the definition of graph of models which can be found in [6] by Demri and D'Souza. We recall this definition as introduced by the authors. For structure of the form $(D, <, =)$ (which is part of IPC*) a symbolic valuation sv is represented by a directed graph $G_{sv} = (terms(\phi), E)$. Nodes are terms of the CLTLB formula and edges are between (two) elements (terms). By construction, G_{sv} has an edge between every pair of elements of N and does not have direct cycles containing a strict relation $<$. The notion of graph of symbolic valuations naturally extends to sequences of symbolic valuations which can be represented as infinite directed graph G_ρ. G_ρ is the superimposition of graphs corresponding to symbolic valuations of ρ.

Lemma 68. *Formula (3.1) is satisfiable, for some $k \in [1, c+1]$, if and only if there exists an ultimately periodic model which is accepted by automaton $\mathcal{A}_\phi$.*

Proof. Given the initial assignment $I(\boldsymbol{x}_0)$ the encoding $[\phi]_k$ of the formula ϕ, $[\phi_\ell]_k$ of the formula ϕ_ℓ and $[\phi_{\mathcal{A}_C}]_k$ of the formula $\phi_{\mathcal{A}_C}$ define exactly the semantics for f such that $(\pi, \widehat{\sigma}_k) \models_k [f]_k$. i.e., $(\pi, \widehat{\sigma}_k)$ is such that all the constraints $|LoopConstraint|_k$, $|ArithConstraints|_k$, $|TempConstraints|_k$ $|LastStateConstraints|_k$, $|Eventually|_k$ (see definition Section 3.3) of $[\phi]_k$, $[\phi_{\mathcal{A}_\ell}]_k$ and $[\phi_{\mathcal{A}_C}]_k$ are satisfiable.

Since $(\pi, \widehat{\sigma}_k) \models_k [f]_k$ then $(\pi, \widehat{\sigma}_k) \models_k [\phi_{\mathcal{A}_C}]_k$. Lemma 67 guarantees that the projection of π of $[\phi_\phi]_k$ models are possible initialized ultimately periodic run of automaton $\mathcal{A}_C$. Now, we demonstrate that, given a subformula $\varphi \in cl(\phi)$, if φ_i holds then there exists a control state of $\mathcal{A}_\phi$, defined by an atom W such that $\varphi \in W$ and which is visited by some initialized run. Observe that the encoding of $[\phi]_k$ defines precisely the truth value of all the subformulae of ϕ. Then, if $[\phi]_k$ is satisfiable, the

set of all subformulae

$$W_i = \{\varphi \mid \varphi \in cl(\phi), \varphi \neq q \text{ for } q \in Q, \text{ if } \theta_i \text{ holds then } \varphi = \theta \text{ else } \varphi = \neg\theta\}$$

defines an atom of the automaton $\mathcal{A}_s$. We do not take into consideration atomic propositions $p \in Q$ representing control states of the automaton $\mathcal{A}_\phi$. The sequence $W_0 \dots W_\ell \dots W_k W_\ell$ of sets W_i for $0 \leq i \leq k+1$ is an ultimately periodic sequence of atoms $W_0 \dots W_{\ell-1}(W_\ell \dots W_k)^\omega$ of $\mathcal{A}_s$ due to the satisfiability of formulae $|LastStateConstraints|_k$ and $|LoopContraint|_k$. By induction on the structure of the formula ϕ we prove that for each position $1 \leq i \leq k+1$, if φ_i holds in the finite model (π, σ_k), then there exists a control state (W, q, a) in the automaton $\mathcal{A}_\phi$ accepting φ such that $\varphi \in W$ and there is a symbolic valuation sv which holds at i for which $q \xrightarrow{sv} p$ for some $q, p \in Q$. Consequently, if the encoding $[f]_k$ of f is satisfiable with a finite model (π, σ_k) then there exists an accepting run for ϕ of the automaton $\mathcal{A}_\phi$ of the form

$$(W_0, q_0, 0) \dots (W_{\ell-1}, q_{\ell-1}, a_{\ell-1}) ((W_\ell, q_\ell, a_\ell) \dots, (W_k, q_k, a_k))^\omega$$

which recognize the sequence of symbolic valuations

$$\rho = sv_0 \dots sv_{\ell-1}(sv_\ell \dots sv_k)^\omega.$$

such that $\phi \in W_0$, $\rho \models_s \phi$ and $\sigma_k \models_k \rho$.

It is worth remarking that the encoding enforce the following equivalence:

- $(W_\ell, q_\ell) = (W_{k+1}, q_{k+1})$ because if there exists a loop then for all the subformulae $\varphi_{k+1} \Leftrightarrow \varphi_\ell$.
- $(sv_{\ell-1}, q_{\ell-1}) = (sv_k, q_k)$.
- No constraints are defined for the arithmetic model σ_k.

Now, we are shall prove by structural induction that for $0 \leq i \leq k$, φ_i holds iff there exists a sequence of atom $W_i, \dots, W_m$ such that $\varphi \in W_i$, a sequence of control states $q_i, \dots, q_m$ and a sequence of values $a_i, \dots, a_m$ defining and accepting subrun of $\mathcal{A}_\phi$ for the formula φ.
The **base case** is given on atomic formulae $\varphi = \alpha \sim \beta$. If φ_i with $1 \leq i \leq k$ then there exists a symbolic valuation sv such that sv_i is satisfiable at the position i and also $sv_i \wedge \varphi_i$; i.e., $sv \models_s \varphi$, as required in the rule defining the transition relation η of the automaton $\mathcal{A}_\phi$. Also, there are two control state q and q' such that $q \Rightarrow sv \wedge \mathbf{X}q'$, and then $q \xrightarrow{sv} q'$, and such that both q_i and q'_{i+1} hold. If q_k then q'_{i+1} as well as q'_ℓ hold (due to the $|LastStateConstraints|_k$). The set of subformulae

W_i define an atom constituting a control state of the automaton $\mathcal{A}_s$ such that $sv = W_i \cap cl(A(\phi))$. Hence, control states (W, q, a) is a control state accepting φ such that $(W_i, q, a) \xrightarrow{sv} (U, q', b)$, for some atoms U. It remains to complete the run with values for a and b. If $a = 0$ then $b = 1$. If $a = 1$ then we proceed according to the rules given for the subformula $\varphi = \psi \mathbf{U} \psi$. If $a = 2$ then we check if $q \in F$; if it is the case than $a = 2$ and $b = 0$; otherwise, $a = 2$.
Then, we proceed by considering all the subformulae in $cl(\phi)$; for each $\varphi \in cl(\phi)$ it is possible to build an accepting subrun of $\mathcal{A}_\phi$.
Inductive step.

- If $\varphi = \mathbf{X}\psi$ then for $1 \leq i \leq k$, $\varphi_i \Leftrightarrow \psi_{i+1}$ and for $i = k$ then $\varphi_i \Leftrightarrow \psi_\ell$. If φ_i holds then $\varphi \in W_i$ and, by hypothesis, W_{i+1} is an accepting state for ψ such that $\psi \in W_{i+1}$. Moreover, there are two control state of $\mathcal{A}_\phi$, $q, p \in Q'$ for which q_i, p_{i+1} and a symbolic valuation $sv \in SV(\phi)$ such that sv_i. It follows that $W_i \xrightarrow{sv} W_{i+1}$ such that $sv \in W_i \cap cl(A(\phi))$ is a subrun of $\mathcal{A}_s$ for which $\mathbf{X}\psi \in W_i$ if, and only if, $\psi \in W_{i+1}$ according to the definition of one-step consistent atoms. Also, $q \xrightarrow{sv} p$ is a subrun of $\mathcal{A}_C$. Hence, $(W_i, q, a) \xrightarrow{sv} (W_{i+1}, p, b)$ is an accepting subrun of $\mathcal{A}_\phi$ for the formula φ, no matter what the value of a, b is considered.

- If $\varphi = \mathbf{Y}\psi$ then for $1 \leq i \leq k$, $\varphi_i \Leftrightarrow \psi_{i-1}$ and for $i = 0$ then $\varphi_i \Leftrightarrow \bot$. The same ideas for subformulae $\mathbf{X}\psi$ applies also in this case. The only difference to be considered is the characterization of the atom W of the sequence when $\neg\varphi_0$ holds; in this particular case, the atom W do not contain the subformula φ in order to be considered an initial state.

- If $\varphi = \psi \mathbf{U} \zeta$ then for $1 \leq i \leq k$, $\varphi_i \Leftrightarrow \psi_i \vee (\psi_i \wedge (\psi_i \mathbf{U} \zeta)_{i+1})$. If φ_i holds two cases have to be considered.
  - ζ_j holds for $i \leq j \leq k$; then φ_k does not hold and, according to the fixpoint representation for the formula, φ_z and ψ_z are satisfied in all the states $i \leq z \leq j$. Then $\varphi \in W_i$ and the sequence of atoms $W_i, \ldots, W_j$ denoting control states of $\mathcal{A}_s$ is such that $\varphi \in W_h$ and $\psi \in W_h$, for $i \leq h \leq j$, and $\zeta \in W_j$; by induction hypothesis, atoms W_h are accepting states for formulae φ and ψ and W_j for the formula ζ. Moreover, there is a sequence of control states $q_i, \ldots, q_j$ for which $q_z \xrightarrow{sv_z} q_{z+1}$, $i \leq z \leq j$. It follows that $W_i \xrightarrow{sv_i} \ldots \xrightarrow{sv_{j-1}} W_j$ for $sv_z \in SV(\phi)$ such that $sv_z = W_z \cap cl(A(\phi))$ and $i \leq z \leq j$ is a subrun of $\mathcal{A}_s$ for which if $\psi_1 \mathbf{U} \psi_2 \in X$ then $\psi_2 \in X$ or $(\psi_1 \in$

X and $\psi_1 \mathbf{U} \psi_2 \in Y$) according to the definitions of one-step consistent atoms; also, $q_i \xrightarrow{sv_i} \ldots \xrightarrow{sv_{j-1}} q_j$ is a subrun of $\mathcal{A}_C$. Hence, $(W_i, q_i, a_i) \xrightarrow{sv_i}, \ldots, (W_j, q_j, a_j)$ is an accepting subrun of $\mathcal{A}_\phi$ for the formula φ. Let us provide how to complete the run so far defined. Let $\mathcal{U} = \{\phi_1 \mathbf{U} \psi_1, \ldots, \phi_m \mathbf{U} \psi_m\}$ be the set of Until formulae occurring in ϕ. Then $\varphi \in \mathcal{U}$. Let $\mathcal{U}'$ be the set of Until formulae occurred in the model; $\mathcal{U}' = \emptyset$ when we start to read the model (π, σ_k) from W_0. If $a_i = 0$, or $a_i = 1$, then, since W_j is an accepting state for φ we add φ to the set of occurred Until, $\mathcal{U}' = \mathcal{U}' \cup \{\varphi\}$. If $\mathcal{U} = \mathcal{U}'$ then all the eventuality for all the formulae in $\mathcal{U}$ are occurred; then, we fix $a_j = 2$. Otherwise, if $a_i = 2$ we check if a final control state $q \in F$ occurs between i an j. If it is not the case, than $a_j = 2$. Otherwise, let $i \leq h \leq j$ the position such that q_h holds; then, $a_h = 0$, $a_t = 1$ for $h + 1 \leq t \leq j - 1$ and $a_j = 2$.

  - Otherwise, φ_k holds and ζ_j holds for $\ell \leq j \leq i$; φ_z is satisfied in all the positions $\ell \leq z \leq j$ and $i \leq z \leq k + 1$. This follows from the formula $\varphi_k \Rightarrow \ell \leq j \leq k \wedge \zeta_j$ of the encoding. Similarly for the previous case, we can build the accepting sequence for φ by considering the positions in the finite model where φ_z is satisfied.

- If $\varphi = \psi \mathbf{S} \zeta$ then for $1 \leq i \leq k + 1$, $\varphi_i \Leftrightarrow \psi_i \vee (\psi_i \wedge (\psi_i \mathbf{S} \zeta)_{i-1})$ and $(\psi_i \mathbf{S} \zeta)_0 \Leftrightarrow \zeta_0$. If φ_i holds for $1 \leq i \leq k + 1$ there exists $0 \leq j \leq i$ such that ζ_j, then the sequence of atoms $W_j \ldots W_i$ denoting control states of $\mathcal{A}_s$ is such that $\varphi \in W_h$ and $\psi \in W_h$ for $j \leq h \leq i$; by induction hypothesis, atoms W_h are accepting states for the formulae φ and ψ and W_j for the formula ζ. The sequence of control states $q_j \ldots q_i$, defined similarly to the Until case, is a subrun of $\mathcal{A}_C$. Then, we can define the subrun of $\mathcal{A}_\phi$ accepting φ as $(W_j, q_j, a_j) \xrightarrow{sv_j} \ldots (W_i, q_i, a_i)$. The rules defining the values a_h are the same as for Until subformulae.

- If $\varphi = \psi \mathbf{R} \zeta$ then for $1 \leq i \leq k$, $\varphi_i \Leftrightarrow \zeta_i \wedge (\psi_i \vee (\psi \mathbf{R} \zeta)_{i+1})$. This case can be reduced to the analysis of a subformula containing $\mathbf{U}$ since the automaton $\mathcal{A}_s$ do not involve dual temporal modalities. Indeed, let us consider a sequence of positions z, $i \leq z \leq j$, such that ψ_j and φ_z hold. Now, from this fact we know that the formula $\neg\varphi \Leftrightarrow \neg(\psi \mathbf{R} \zeta)$ is not satisfiable in the same sequence. By duality of $\mathbf{U}$ and $\mathbf{R}$, $\neg\varphi = \neg(\psi \mathbf{R} \zeta) = \neg\psi \mathbf{U} \neg\zeta$. Then, each atom W_z of the sequence $W_j \ldots W_i$ does not contain the subformula $\neg\psi \mathbf{U} \neg\zeta$ but they may contain $\psi \mathbf{U} \neg\zeta$, $\neg\psi \mathbf{U} \zeta$ or $\psi \mathbf{U} \zeta$. Indeed, the sequence

recognizing φ visits, first, atoms which contains $\neg\psi\mathbf{U}\zeta$ or $\psi\mathbf{U}\zeta$ (but not $\psi\mathbf{U}\zeta$), for $i \leq z \leq j$, and atoms containing $\psi\mathbf{U}\zeta$, in the position j. Moreover, each of these subformulae are satisfied in the position in which they occurs, by definition of the encoding for φ.

- If $\varphi = \psi\mathbf{T}\zeta$ then for $1 \leq i \leq k+1$, $\varphi_i \Leftrightarrow \psi_i \wedge (\psi_i \vee (\psi_i\mathbf{T}\zeta)_{i-1})$ and $(\psi_i\mathbf{T}\zeta)_0 \Leftrightarrow \zeta_0$. Similarly for the case of Release, we can build the accepting run of $\mathcal{A}_\phi$ for φ by considering the dual relation $\neg(\psi_i\mathbf{T}\zeta) \Leftrightarrow \neg\mathbf{S}\neg\zeta$. Let us consider a sequence of positions z, $j \leq z \leq i$, such that ψ_j and φ_z hold. Now, from this fact we know that the formula $\neg\varphi = \neg(\psi\mathbf{R}\zeta)$ is not satisfiable in the same sequence. Therefore, each atom W_z of the sequence $W_j \dots W_i$ does not contain the subformula $\neg\psi\mathbf{S}\neg\zeta$ but they may contain $\psi\mathbf{S}\neg\zeta$, $\neg\psi\mathbf{S}\zeta$ or $\psi\mathbf{S}\zeta$. Hence, it is an accepting subrun of $\mathcal{A}_s$ for the formula φ. The sequence of control states q is define as in the previous cases as well as the values for a_z.

Finally, since $[f]_k$ holds then $\phi \in W_0$ and the sequence of length k of triple (W_i, q_i, a_i), defined starting from $(W_0, q_0, 0)$, is a periodic accepting run of $\mathcal{A}_\phi$ for ϕ.

Conversely, let us suppose there exists an ultimately periodic model which is accepted by $\mathcal{A}_\phi$. It is a sequence of symbolic valuations $\rho = uv^\omega$ which can be represented by a finite word uv^ω of finite length k and recognized by a periodic run of $\mathcal{A}_\phi$ which is defined in the following way:

$$\upsilon = (W_0, q_0, 0)(W_1, q_1, 1) \dots (W_{\ell-1}, q_{\ell-1}, a_{\ell-1})((W_\ell, q_\ell, a_\ell), \dots, (W_k, q_k, a_k))^\omega$$

where the sequence of $W_0 W_1 \dots W_{\ell-1}(W_\ell \dots W_k)^\omega$ is a run of the automaton $\mathcal{A}_s$ and $q_0 \dots, q_{\ell-1}(q_\ell \dots q_k)^\omega$ is a run of the automaton $\mathcal{A}_C$. In particular, ρ is defined by the projection on the alphabet of $SV(\phi)$ of the subformulae occurring in every W_i, for $0 \leq, i \leq k$. Since υ is an accepting run then there exists at least one control state (W_j, q_j, a_j) such that $a_j = 0$. The sequence of control states $q_0 \dots q_{\ell-1}(q_\ell \dots q_k)^\omega$ is an accepting run of the automaton $\mathcal{A}_C$ and, by construction, along with the sequence of symbolic valuations, is a model for the formula $\phi_{\mathcal{A}_C}$; analogously, the sequence of control state $W_0 \dots W_{\ell-1}(W_\ell, \dots, W_k)^\omega$ is an accepting run of the automaton $\mathcal{A}_s$. Now, we shall give a justification explaining this. A model for $[\phi]_k$ and $[\phi_{\mathcal{A}_C}]_k$ is given by the union of (the truth value of) all the subformulae in W_i, q_i and the values of variables occurring in ϕ. In particular, we define model π in this way: given a position $0 \leq i \leq k+1$, for all subformulae $\varphi \in cl(\phi)$ we define

- $\tau_k(i, \varphi) = 1$ when $\varphi \in W_i$,

- $\tau_k(i,\varphi) = 0$ when $\neg\varphi \in W_i$.

Moreover, $\pi(i,q) = 1$ when $q = q_i$, else $\pi(i,q) = 0$. The encoding of boolean connectives in $|PropConstraints|_k$ agrees with the definition of consistency of atoms. All the pair (W_i, W_{i+1}) in the sequence are one-step consistent by definition of $|TempConstraints|_k$; then, the encoding of subformulae $\varphi = \mathbf{X}\psi$ and $\varphi = \psi\mathbf{U}\zeta$ follows naturally. The truth value for subformulae $\psi\mathbf{R}\zeta$ is derived by the dual relation: $\neg\psi\mathbf{R}\zeta = \neg\psi\mathbf{U}\neg\zeta$. The sequence of symbolic valuations is consistent and all the a.t.t.'s in the encoding of $[\phi]$ can be uniquely defined by considering at each position i a symbolic valuation sv_i. Now, let us consider the sequence $\rho' = sv_0 \dots sv_{\ell-1}(sv_\ell \dots sv_k)$ of symbolic valuations such that $I(\mathbf{x}_0) \models sv_0$. The model $\sigma_k(i,x)$ for each variable $x \in V$ and for $0 \leq i \leq k + \lceil\phi\rceil$ is defined by an edge respecting assignment of value in D for the graph $G_{\rho'}$ according to Demri and D'Souza [6] (Lemma 5.2) from the initial configuration $I(\boldsymbol{x}_0)$. All the variable x_i and all the a.t.t.'s τ_i are uniquely defined by considering the values of variables in sv_i. Since the model is periodic, there exists a control state (W_ℓ, q_ℓ, a) which is visited in the position $k+1$ of the run υ. It witnesses the satisfaction of the formulae in $|LastStateConstraints|_k$ for which $\varphi_{k+1} \Leftrightarrow \varphi_l$ for all the subformulae of ϕ. Finally, let us consider the set of formulae of $|Eventually|_k$. If the subformula $\varphi = \psi\mathbf{U}\zeta$ belongs to the atom W_k then there exists a position $j \geq k$ such that ζ_j holds. Since the model is periodic then this position occurs in $k \leq j \leq 2k$, i.e., a position in $\ell \leq j_\zeta \leq k$. Moreover, if $\neg(\psi\mathbf{R}\zeta) = \neg\psi\mathbf{U}\neg\zeta$ belongs to W_k then there exists a position $j \geq k$ such that $\neg\zeta_j$ holds. As in the previous case $\ell \leq j \leq k$. Hence, the $|LastStateConstraints|_k$ are satisfied. The initial atom W_0 is such that $\mathbf{Y}\varphi \notin W_0$ and if $\psi\mathbf{S}\zeta \in W_0$ then $\zeta \in W_0$ which witnesses the encoding of subformulae $\varphi = \mathbf{Y}\psi$ and $\varphi = \psi\mathbf{S}\zeta$ in the first position, i.e., $\varphi_0 \Leftrightarrow \bot$ and $\varphi_0 \Leftrightarrow \zeta_0$. □

The completeness bound for BSP of CLTLB(L) formulae is defined by the recurrence diameter of $\mathcal{A}_\phi$. Next theorem requires notions of denseness and openness of sets. Let $(D, <, =)$ be the structure defining the language of atomic formulae of CLTL. We say that D is *dense*, with respect to the order $<$, if for each $d, d' \in D$ such that $d < d'$, there exists $d'' \in D$ such that $d < d'' < d'$. We say that D is *open* when for each $d \in D$, there exist two elements $d', d'' \in D$ such that $d' < d < d''$.

Theorem 69. *For constraint systems IPC*, $(\mathbb{N}, <, =)$, $(\mathbb{Z}, <, =)$, $(D, <, =)$, where D is dense and open, and their extensions with constants, there exists a finite completeness threshold for BSP.*

Proof. The statement is a consequence of Proposition 53. In particular, if $\mathcal{A}_\phi$ accepts a word ρ then it must accept also ultimately periodic words (by the nature of the acceptance condition of the automaton) which admit arithmetic models since they respect condition C. By Lemma 68, if there does not exist a value of k which makes the formula satisfiable, then language $\mathscr{L}(\mathcal{A}_\phi)$ is empty; otherwise, there exists a model σ and an ultimately periodic sequence of symbolic valuations ρ such that $\sigma \models \rho$. □

In practice, when the domain of L is $\mathbb{N}$ or $\mathbb{Z}$, formula (3.1) can be simplified. In fact, if it is satisfiable, the sequence accepted by the automaton $\mathcal{A}_C$ is already locally consistent, as two consecutive symbolic valuations are satisfiable when they are locally consistent, due to the consistency of the encoding of ϕ'. Then, formula ϕ_ℓ can be removed and formula (3.1) becomes $I(\boldsymbol{x}_0) \wedge \phi \wedge \phi_{\mathcal{A}_C}$. When the domain of L has the completion property, instead, formula (3.1) becomes $I(\boldsymbol{x}_0) \wedge \phi \wedge \phi_\ell$. In this case, formula ϕ_ℓ is necessary to define the sequence of locally consistent symbolic valuations, since the automaton $\mathcal{A}_C$ is not needed anymore. Moreover, we can estimate the value of the completeness bound without building automaton $\mathcal{A}_\phi$. Since the size of the set of control states of $\mathcal{A}_\phi$ is $\mathcal{O}(2^{|\phi|})$, we can consider a rough estimation for the completeness bound defined by the value $d \times |SV(\phi)| \times 2^{|\phi|}$, where d is the cardinality of the control state set of $\mathcal{A}_C$ and $|SV(\phi)|$ is representative of the dimension of ϕ_ℓ (which is again exponential in the size of the formula).

4. Bounded Model Checking Problem

4.1. Bounded CLTL model-checking

Results of previous chapter allows us to obtain a complete procedure for CLTLB(L) model-checking. Since model checking and satisfiability problems are reducible to each other, from the traditional transformation proposed by Sistla and Clarke in [52], then a completeness threshold for CLTLB(L) model checking problem is derived from the one of satisfiability. In particular, if we consider counter systems in CS(L) or L-automata (Büchi automata where transitions are labeled by formulae belonging to CLTLB(L)) where L is one of fragments considered in previous chapter, the completeness result for model-checking is immediate.

Following theorem is based on completeness results of Section 3.6 and reduction from model-checking to satisfiability. Let $\langle M, \phi \rangle$ be the CLTLB(L) formula defining the model-checking problem for M and ϕ be the CLTLB(L) formula.

Theorem 70. *Let ϕ be a* CLTLB(L) *formula and M be a counter system in* CS(L) *or L-automata where L is a language such that satisfiability for* CLTLB(L) *is complete. Then, $M, q_0 \models \phi$ if, and only if, there exists $k \geq 0$ such that $M, q_0 \models_k \phi$.*

Proof. The theorem is proved by reducing the model-checking $M, q_0 \models_k \phi$ to satisfiability problem for a CLTLB(L) formula $\langle M, \phi \rangle$. Then, formula $\langle M, \phi \rangle$ is checked for k-bounded satisfiability. If $\langle M, \phi \rangle$ is satisfiable over k-bounded models then $M, q_0 \models_k \phi$ holds and, consequently, $M, q_0 \models \phi$. Otherwise, the bound k is increased. The existence of un upper bound for the value k guarantees that the procedure eventually terminates. If there does not exists a value k such that $M, q_0 \models_k \phi$ then $M, q_0 \not\models \phi$. □

When we are dealing with generic counter systems in CS(QFP), Model-checking problem over k-bounded models becomes an effective semi-decision for "extended" reachability property. Transition relation of a

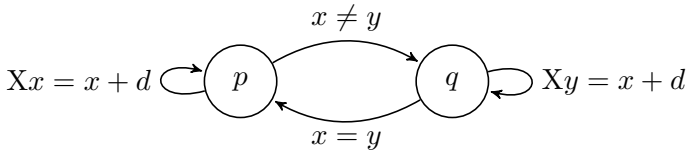


Figure 4.1.: Two variables counter system - $d \in \{+1, -1\}$

counter system $(Q, n, \delta) \in \mathrm{CS}(\mathrm{QFP})$ can be translated into a CLTLB(QFP) formula which represents all initialized finite runs of length k:

$$\mu_k := \bigwedge_{i=0}^{k-1} q_i \Rightarrow \mathbf{X} q'_{i+1} \wedge \xi$$

for all $(q, \xi, q') \in \delta$. The language for properties is CLTLB(L) with $L \subseteq \mathrm{QFP}$ and formulae involving counters are of the form $\top \mathbf{U} \phi$ and ϕ is a Boolean combination of atomic formulae over atomic terms. For instance, we would like to verify whether the system in Figure 4.1 satisfies the formula $\mathbf{GF}(p \Rightarrow \mathbf{F}q) \wedge \mathbf{F}(x < y)$. By verifying satisfiability of the conjunction of μ_k and ϕ over bounded models, i.e., we check satisfiability of $[\mu \wedge \phi]_k$, we may answer to a reachability problem only when formula $[\mu \wedge \phi]_k$ is satisfiable. Since reachability problem for generic counter systems is undecidable, there does not exists a value K such that checking satisfiability of formulae $[\mu \wedge \phi]_k$, with $1 \leq k \leq K$, always allow us to give an answer to a model-checking problem. In fact, being a semi-procedure, when $[\mu \wedge \phi]_k$ is unsatisfiable we can not deduce non validity of property ϕ.

Nonetheless, k-bounded model-checking problem is complete if we restrict models to the class of reversal-bounded counter systems. We provide a simple practical example to make the bridge between arguments of the previous chapter and forthcoming results. Let us build the r-reversal bounded counter system $\mathcal{S}'$ from the system in Figure 4.1 by using the following standard construction. We define $Q' = \{p, q\} \times \{\searrow, \nearrow\}^2 \times [0, r]^2$ and δ such that the system $(\mathcal{S}, (q, \{\nearrow\}^2, \{0\}^2), \boldsymbol{x})$ is r-reversal-bounded by construction. Relation δ' is defined as follows according to vectors $\boldsymbol{mode} \in \{\searrow, \nearrow\}^n$, representing modes for each counter, and $\boldsymbol{rev} \in [0, r]^n$ which stores the number of reversal performed by $\mathcal{S}$ during a computation; $(q, \boldsymbol{mode}, \boldsymbol{rev}) \xrightarrow{\xi} (q', \boldsymbol{mode}', \boldsymbol{rev}') \in \delta'$ when $q \xrightarrow{\xi} q'$ and

	ξ	$\boldsymbol{mode}(i)$	$\boldsymbol{mode}'(i)$	$\boldsymbol{rev}'(i)$
1.	$\boldsymbol{d}(i) \geq 0$	$\nearrow$	$\nearrow$	$\boldsymbol{rev}(i)$
2.	$\boldsymbol{d}(i) > 0$	$\searrow$	$\nearrow$	$\boldsymbol{rev}(i)+1$
3.	$\boldsymbol{d}(i) \leq 0$	$\searrow$	$\searrow$	$\boldsymbol{rev}(i)$
4.	$\boldsymbol{d}(i) < 0$	$\nearrow$	$\searrow$	$\boldsymbol{rev}(i)+1$

In the case 2. and 4. the transition is defined if $\boldsymbol{rev}(i) < r$. It is worth noting that the following two proposition are equivalent:

(a) there is a run ρ in $\mathcal{S}$ such that $(q, \boldsymbol{x}) \xrightarrow{*} (q', \boldsymbol{x}')$ and each counter has at most r reversal.

(b) there is a run ρ' in $\mathcal{S}'$ such that $((q, \{i\}^n, \{0\}^n), \boldsymbol{x}) \xrightarrow{*} ((q', \boldsymbol{m}, \boldsymbol{r}), \boldsymbol{x}')$ and each counter has at most r reversal, for some pair $(\boldsymbol{m}, \boldsymbol{r})$

Let us suppose we would verify whether there exists an infinite run of the r-reversal bounded counter system $\mathcal{S}'$ such that the control state q is visited infinitely often and the counter y is strictly monotonic. Let $\mu_{\mathcal{S}'}$ be the formula encoding the transition relation of $\mathcal{S}'$. Then, we simply check for k-bounded satisfiability the following formula:

$$\bigvee_{j \in [0,r]} \bigvee_{m \in \{\nearrow, \searrow\}^2} \mu_{\mathcal{S}'} \wedge \mathbf{GF}(q, m, j) \Rightarrow \mathbf{FG}(\mathrm{X}y > y)$$

The construction of the system $\mathcal{S}'$ can be avoided provided that the CLTLB formula restricts the set of runs to the only r-reversal bounded ones. To this end, we introduce auxiliary variables $r_x, r_y \in [0, r]$ which count the number of reversal occurring along runs and such that:

- r_x, r_y will eventually stabilize; i.e., no reversal will occur after some position and their value will remain constant: $\mathbf{FG}(\mathrm{X}r_\alpha = r_\alpha \wedge r_\alpha < r)$
- keep track of reversals occurred: $\mathbf{G}(\mathrm{X}r_\alpha = r_\alpha + 1 \Leftrightarrow \mathrm{Y}\alpha < \alpha \wedge \alpha > \mathrm{X}\alpha)$

where $\alpha \in \{x, y\}$.

In its simplicity, this example represents a basic form of reversal-bounded model-checking which will be argument presented hereafter and which can be found in [9] by Bersani and Demri. Practical applications of reversal-bounded model-checking rely on Bounded CLTLB model-checking. In fact, we will prove that reversal-bounded model-checking can be reduced to checking whether there exists ultimately periodic runs

of bounded length satisfying a given CLTLB formula and some auxiliary constraints restricting the monotonicity of counters.

Previous example shows that existential model-checking, i.e., the problem of finding a run of a counter system $\mathcal{S}$ satisfying a given formula ϕ (written $\mathcal{S} \models \phi$), is, in general, undecidable. In order to regain decidability some restrictions must be imposed because infinite runs of counter systems are not, in general, effectively representable in Presburger arithmetic. As immediate consequence, analyzing all the set of runs is unmanageable since they do not benefits of a useful decidable representation. Therefore, the problem we are going to study is the model-checking problem over counter system $\mathcal{S}$ restricted to r-reversal-bounded runs. As we explained in Section 2.7.3, bounded model-checking for finite systems benefits from nice properties on runs that allow the existence of an upper bound on the length of runs to be checked (*completeness threshold*). Reversal-bounded counter systems can be adopted to perform bounded model-checking by exploiting a technique handling temporal languages with arithmetic given in Chapter 3. In fact, we prove that the length of runs satisfying a temporal formula is bounded by a double exponential in the size of the problem. Therefore, given a positive integer $r \geq 0$, checking whether a counter systems admits a r reversal-bounded infinite run, which can be model of a temporal formula, can be done be checking a finite amount of bounded model-checking problem. In case of positive answer, the process terminates, otherwise the value of r can be incremented. It is worthy to be noticed that the notion of reversal-boundedness we propose is different with respect to the standard one. In fact, we introduce the use of terms instead of restricting the property of reversal-boundedness only to counters. This allow us to encompass the standard notion given by Ibarra and also to focus precisely the source of undecidability of counter systems which are enriched by non trivial arithmetical constraints (i.e., involving more than one variable like $x+2y-z<4$). In particular, we are able to explain why strong reversal-boundedness is required to retain decidability in such class of counter systems. Although terms can be simulated by new fresh variables, we avoid this construction and we focus on the theoretical aspect of terms of guards. Also, the introduction of new fresh variables affects directly the (practical) complexity cost of the problem. Moreover, counter systems we consider are more general than Minsky machines: their guards are definable in quantifier-free fragment of Presburger arithmetic and update vectors in $\mathbb{Z}^n$. Finally, we characterize the computational complexity of the existence of r-reversal-bounded runs but also we effectively express the set of configurations admitting such runs in Presburger arithmetic.

Decidability results obtained by our approach refines results by Dang

et al. in [40] and it is closed to decidability results for reversal-bounded counter systems augmented with data structures such as stacks or queues. Counter systems enriched by a stack is considered by Hague and Lin [83] of which we give more details in Chapter 6. Differently from this work, our temporal language is reacher because it contains control states, past operators and also arithmetical constraints belonging to QFP. It allows us to specify non trivial arithmetic properties on systems like fairness contraint over counters, e.g., $\mathbf{GF}(\mathrm{X}x = x + y)$, which are undecidable in the general case (see undecidability results of $\exists$-Presburger-always problem in Chapter 2.6). Finally, to compare complexity results with respect ot other works, Kopczynski and To in [84, Theorem 22] showed EXPTIME upper bound for LTL model-checking over reversal-bounded counter automata but the logical language has no arithmetical constraints and the number of reversals r is encoded in unary. Complexity results given in Section 4.4 are built for instances where all integers values (k and constants) are encoded in binary and the proof technique relies on existence of small solutions for equation systems while the proof of [83, Theorem 1] exploits Parikh's Theorem.

4.2. Decidability of Reversal Bounded Model Checking Problem

4.2.1. New definition of Reversal Boundedness

In this section, we generalize the notion of reversal-boundedness and the notion of *strong* reversal-boundedness introduced in Chapter 2.6.2

Let $\mathcal{S} = (Q, n, \delta)$ be a counter system and $\mathtt{T}$ be a finite set of terms including $\{x_1, \ldots, x_n\}$. We order the terms in $\mathtt{T}$ with $x_1, \ldots, x_n, t_1, \ldots, t_{n'}$ (then $\mathrm{card}(\mathtt{T}) = n + n'$, where n' can possibly be equal to 0). From a run $\rho = (q_0, \boldsymbol{x}_0), (q_1, \boldsymbol{x}_1), \ldots$ of $\mathcal{S}$, in order to describe the behavior of counters and terms varying along ρ, we define a sequence of *mode vectors* $\boldsymbol{m}_0, \boldsymbol{m}_1, \ldots$ (of the same length as ρ) such that each $\boldsymbol{m}_i$ belongs to $\{\nearrow, \searrow\}^{n+n'}$. Given a term $t = \sum_k a_k x_k$ and a counter vector $\boldsymbol{x}$, we write $t(\boldsymbol{x})$ to denote the integer $\sum a_k \boldsymbol{x}(k)$. The sequence $\boldsymbol{m}_0, \boldsymbol{m}_1, \ldots$ is similarly defined to the sequence of modes in Section 2.6.2 but here extended to the set of terms $\mathtt{T}$.

- $\boldsymbol{m}_0$ is the unique vector in $\{\nearrow\}^{n+n'}$.
- For $j \geq 0$ and $i \in [1, n + n']$ with the ith term in $\mathtt{T}$ equal to t, we have
  1. $\boldsymbol{m}_{j+1}(i) \stackrel{\text{def}}{=} \boldsymbol{m}_j(i)$ when $t(\boldsymbol{x}_j) = t(\boldsymbol{x}_{j+1})$,

2. $\boldsymbol{m}_{j+1}(i) \stackrel{\text{def}}{=} \nearrow$ when $t(\boldsymbol{x}_{j+1}) - t(\boldsymbol{x}_j) > 0$,
3. $\boldsymbol{m}_{j+1}(i) \stackrel{\text{def}}{=} \searrow$ when $t(\boldsymbol{x}_{j+1}) - t(\boldsymbol{x}_j) < 0$.

As we did for general reversal-boundedness in Section 2.6 we define the set $Rev_i = \{j \in \mathbb{N} : \boldsymbol{m}_j(i) \neq \boldsymbol{m}_{j+1}(i)\}$; then, we say that

Definition 71. *ρ is r-$\mathtt{T}$-reversal-bounded for some $r \geq 0$ when for all $i \in [1, n+n']$, $\text{card}(Rev_i) \leq r$.*

Given a counter system $\mathcal{S}$, we write $\mathtt{T}_\mathcal{S}$ to denote the finite set of terms t occurring in atomic guards of the form $t \sim k$ with $\sim \in \{\leq, \geq\}$ and $k \in \mathbb{Z}$. Now, we can give the notion of r-$\mathtt{T}$-reversal-bounded counter systems.

Definition 72. *An initialized counter system $(\mathcal{S}, (q, \boldsymbol{x}))$ is $r-\mathtt{T}_\mathcal{S}$-reversal-bounded when there is $r \geq 0$ such that every run from $(q, \boldsymbol{x})$ is r-$\mathtt{T}_\mathcal{S}$-reversal-bounded.*

When $\mathtt{T}$ is reduced to $\{x_1, \ldots, x_n\}$, $\mathtt{T}$-reversal-boundedness is equivalent to reversal-boundedness of Ibarra [3]. When a sequence of transitions has a unique update vector, the mode vector remains constant. In this case, when an initialized counter system in CS(QFP), involving guards with terms in $\mathtt{T}'$, is strongly reversal-bounded, as defined in Section 2.6.2, then it is $(\mathtt{T}' \cup \{x_1, \ldots, x_n\})$-reversal-bounded.

Terms involved in the temporal formula $\phi \in$ CLTL(QFP) are treated in a specific way since they may contains temporal modalities X over terms (counters). In particular, let us consider to adiacent configuration $(q_i, \boldsymbol{x}_i)$ and $(q_{i+1}, \boldsymbol{x}_{i+1})$ such that $\boldsymbol{x}_{i+1} - \boldsymbol{x}_i = \boldsymbol{v}$. The next value of the u-th counter, denoted by $\mathrm{X}x_u$, is equal to the current value of x_u plus some integer $\boldsymbol{v}(u)$ and the value of the term $(\sum_u a_u \mathrm{X}x_u) + (\sum_u b_u x_u)$ (in ϕ), at the current position i, is equal to the value of $\sum_u (a_u + b_u) x_u$ plus some constant $\boldsymbol{v}(u)$ depending on the transition between positions i and $i+1$. Given an $\phi \in$ CLTL(QFP), we write $\mathtt{T}_\phi$ to denote the finite set of terms of the form $\sum_k (a_k + b_k) x_k$ when $t = (\sum_k a_k \mathrm{X}x_k) + (\sum_k b_k x_k)$ is a term $t \sim k$ occurring in ϕ, with $\sim \in \{\leq, \geq, <, >, =\}$ and $k \in \mathbb{Z}$. Strong reversal-boundedness of runs is easy to obtain directly by adding an LTL(QFP) formula to the property we desire; from a an arbitrary position along the runs, counters are strictly monotonic:

$$\bigvee_{q \xrightarrow{(\xi, \boldsymbol{v})} q' \in \delta} \mathbf{FG}\Big(\bigwedge_{i \in [1,n]} ((\mathrm{X}x_i - x_i) = \boldsymbol{v}(i)) \Big)$$

The main problem we want to face with is the following:

REVERSAL-BOUNDED MODEL-CHECKING

Input	a counter system $\mathcal{S} \in \text{CS(QFP)}$, a configuration $(q, \boldsymbol{x})$, a formula $\phi \in \text{CLTL(QFP)}$ and bound $r \in \mathbb{N}$
Problem	is there an infinite run ρ from $(q, \boldsymbol{x})$ such that $\rho, 0 \models \phi$ and ρ is r-$\mathtt{T}$-reversal-bounded with $\mathtt{T} = \mathtt{T}_{\mathcal{S}} \cup \mathtt{T}_{\phi}$?

The restriction of RBMC to counter systems in the class $\text{CS}(\text{L}_1)$ and to formulae in $\text{CLTL}(\text{L}_2)$ is denoted by $\text{RBMC}(\text{CS}(\text{L}_1),\text{CLTL}(\text{L}_2))$ with $\text{L}_1, \text{L}_2 \subseteq \text{QFP}$.

In the next section, we give our decidability result for RBMC by providing a reduction to repeated reachability problem and after to a reachability problem for CS(QFP) systems. Given a class $\mathcal{C}$ of counter systems the RB-REACH($\mathcal{C}$) for the class is defined as follows:

REVERSAL-BOUNDED REACHABILITY PROBLEM FOR $\mathcal{C}$

Input	- a counter system $\mathcal{S} \in \mathcal{C}$, - two configurations $(q_0, \boldsymbol{x}_0)$ and $(q_f, \boldsymbol{x}_f)$, - $r \geq 0$
Problem	Is there an r-$\mathtt{T}_{\mathcal{S}}$-reversal-bounded run from $(q_0, \boldsymbol{x}_0)$ to $(q_f, \boldsymbol{x}_f)$?

Similarly, the *reversal-bounded control state repeated reachability problem for* $\mathcal{C}$, written RB-REP-REACH($\mathcal{C}$), is defined as follows:

REVERSAL-BOUNDED CONTROL STATE R.R. PROBLEM FOR $\mathcal{C}$

Input	- a counter system $\mathcal{S} \in \mathcal{C}$, - a configuration $(q_0, \boldsymbol{x}_0)$, a control state q_f, - $r \geq 0$
Problem	Is there an infinite r-$\mathtt{T}_{\mathcal{S}}$-reversal-bounded run from $(q_0, \boldsymbol{x}_0)$ such that q_f is repeated infinitely often?

When $(\mathcal{S}, (q_0, \boldsymbol{x}_0))$ is reversal-bounded, then reversal-bounded reachability corresponds to the standard notion of reachability.

4.3. From reversal-bounded model-checking to reachability

In this section, we show how to reduce

1. RBMC into RB-REP-REACH(CS(QFP)),

2. RB-REP-REACH(CS(QFP)) into RB-REP-REACH(CS(QFP(<))),
3. RB-REP-REACH(CS(QFP(<))) into RB-REACH(CS(QFP(<))).

The first reduction is defined by synchronizing counter systems with Büchi automata for temporal formulae according to the standard Vardi-Wolper method in [51] for model-checking. The second one is easier and it is realized by storing information on modulo classes of counters within control states of a new counter system derived from the original one. Finally, reduction of repeated reachability to reachability follows a method similar to the one proposed by Dang et al. in [40].

4.3.1. Towards control state repeated reachability

In this section, we show how to reduce RBMC to RB-REP-REACH(QFP) by synchronizing counter systems with Büchi automata for temporal formulae, as done for LTL model-checking by Vardi and Wolper in [33], and also for Petri nets by Esparza [85]. The result of synchronizing counter system and a Büchi automaton is a new counter system which realizes the transition relation of the original one and which embeds the acceptance condition over infinite runs from the Büchi automaton for the formula. Therefore, runs of the new counter system are restricted only to runs satisfying the LTL(QFP) property.

Let $\mathcal{S} = (Q, n, \delta) \in \text{CS(QFP)}$, $(q, \boldsymbol{x})$, $\phi \in \text{CLTL(QFP)}$ and $r \in \mathbb{N}$ be an instance of RBMC. Let $\mathcal{A}_\phi$ be the Büchi automaton for ϕ. Then, counter system $\mathcal{S}'$ resulting from synchronizing $\mathcal{A}_\phi$ with $\mathcal{S}$ embeds all the terms form $\mathcal{S}$ and ϕ and it is such that $\mathtt{T}_{\mathcal{S}'} = \mathtt{T}_{\mathcal{S}} \cup \mathtt{T}_\phi$. We provide now details about how to build the Büchi automaton $\mathcal{A}_\phi$ by adapting the standard Vardi-Wolper construction to our goals. Let A be the set closed under negation of atomic formulae of the form q (representing control states of $\mathcal{S}$), $t \sim k$ and $t \equiv_c k'$ in ϕ. As usual, $cl(\phi)$ denotes the closure of ϕ which is the smallest set of formulae closed under subformulae, closed under negations (double negations are eliminated) and containing ϕ. The set $atoms(\phi)$, of *atoms* of ϕ contains all the subsets of $cl(\phi)$ that are maximally consistent and such that for every formula $\xi \in A$ then either ξ or $\neg\xi$ belongs to the set.

A pair (X, Y) of atoms is *one-step consistent* when

- for every $\mathbf{X}\xi \in cl(\phi)$, $\mathbf{X}\xi \in X \Leftrightarrow \xi \in Y$,
- for every $\mathbf{Y}\xi \in cl(\phi)$, $\mathbf{Y}\xi \in Y \Leftrightarrow \xi \in X$,
- for every $\xi_1 \mathbf{U} \xi_2 \in cl(\phi)$, $\xi_1 \mathbf{U} \xi_2 \in X$ iff $\xi_2 \in X$ or ($\xi_1 \in X$ and $\xi_1 \mathbf{U} \xi_2 \in Y$),

- for every $\xi_1 \mathbf{S} \xi_2 \in cl(\phi)$, $\xi_1 \mathbf{S} \xi_2 \in Y$ iff $\xi_2 \in Y$ or ($\xi_1 \in Y$ and $\xi_1 \mathbf{S} \xi_2 \in X$).

It is worth noting that the set A contains subset which are trivially false. Then, we have to restrict A to the set of $valid(A)$ which is such that $valid(A) \subseteq \mathcal{P}(A)$ and $X \in valid(A)$ if, an only if, $X \subseteq A$ and for each atomic formula ξ in ϕ either $\xi \in X$ or $\neg\xi \in X$ but not both. However, a set $P \in valid(A)$ may be not satisfiable, i.e., there does not exists an assignment to all variables of arithmetical constraints of P such that all formulae $\xi \in P$ evaluates true.

The generalized Büchi automaton $\mathcal{A}'_\phi = (\Sigma, Q', Q_0, \eta, F)$ is defined as follows:

- $\Sigma \stackrel{\text{def}}{=} Q \times valid(A)$.
- Q' is the set of atoms.
- Q_0 is the set of atoms X such that $\phi \in X$, $X \cap \{\mathbf{Y}\xi : \mathbf{Y}\xi \in cl(\phi)\} = \emptyset$ and for each $\mathbf{S}$-formula $\xi_1 \mathbf{S} \xi_2 \in cl(\phi)$, $\xi_1 \mathbf{S} \xi_2 \in X$ iff $\xi_2 \in X$.
- $X \xrightarrow{(q,\beta)} Y \in \eta \stackrel{\text{def}}{\Leftrightarrow} \{q\} = X \cap Q$, $\beta = X \cap A$ and (X, Y) is one-step consistent.
- let $\{\phi_1 \mathbf{U} \psi_1, \ldots \phi_m \mathbf{U} \psi_m\}$ be the set of Until formulae occurring in $cl(\phi)$, define $F = \{F_1, \ldots, F_m\}$ with $F_i = \{X \in Q \mid \phi_i \mathbf{U} \psi_i \notin X \text{ or } \psi_i \in X\}$.

The non-generalized version $\mathcal{A}_\phi$ of the automaton $\mathcal{A}'_\phi$ can be built in logarithmic space in the size of $\mathcal{A}_\phi$. In $\mathcal{A}_\phi$, the states are in $Q' \times [0, m]$. The construction is given in preliminaries of the thesis in Section 2.4.

The synchronized counter system $\mathcal{S}' = (Q^\times, n, \delta^\times)$ is defined as follows (definition considers the non generalized automaton $\mathcal{A}_\phi$ for ϕ):

- $Q^\times = Q \times Q' \times [0, m]$,
- $(q, X, a) \xrightarrow{(\xi \wedge X(\boldsymbol{b}), \boldsymbol{b})} (q', X', a') \in \delta^\times \stackrel{\text{def}}{\Leftrightarrow}$
  - $(X, a) \xrightarrow{(q,\beta)} (X', a') \in \eta$ and $q \xrightarrow{(\xi, \boldsymbol{b})} q' \in \delta$,
  - $X(\boldsymbol{b})$ is the conjunction of all QFP atomic formulae obtained from $X \cap A$ where each subterm $\mathrm{X}x_i$ is replaced by $x_i + \boldsymbol{b}(i)$, (constants are moved to obtain formulae of the form $t \sim k$).

It is worth noting that $\xi \wedge X(\boldsymbol{b})$ may be unsatisfiable; in that case, the transition is never fired. Observe also that $\mathtt{T}_{\mathcal{S}'} = \mathtt{T}_{\mathcal{S}} \cup \mathtt{T}_\phi$.

Whenever the repeated reachability problem, for the class of systems which $\mathcal{S} \times \mathcal{A}_\phi$ belongs to, is effectively decidable, the RBMC problem

for such a class is consequently decidable. An instance of RBMC can be reduced to several instances of RB-REP-REACH(QFP) for $\mathcal{S}'$. In particular, RBMC can be solved by checking a finite number of instances of RB-REP-REACH(QFP) depending which initial states and accepting states are considered.

The following lemma states precisely the relation between the existence of r-$(\mathtt{T}_{\mathcal{S}} \cup \mathtt{T}_{\phi})$-reversal-bounded run of $\mathcal{S}$ and the existence of a run in the synchronized product $\mathcal{S}'$ which visits infinitely often a control state witnessing the Büchi acceptance condition defined by the automaton $\mathcal{A}_{\phi}$.

Lemma 73 (Lemma 1, [9]). *Let $\mathcal{S} = (Q, n, \delta) \in$ CS(QFP), $(q, \boldsymbol{x})$, $\phi \in$ CLTL(QFP) and $r \in \mathbb{N}$ be an instance of RBMC and $\mathcal{S}'$ be the counter system in* CS(QFP) *obtained by synchronizing $\mathcal{S}$ with $\mathcal{A}_{\phi}$. The propositions below are equivalent:*

(I) *there is an infinite r-$(\mathtt{T}_{\mathcal{S}} \cup \mathtt{T}_{\phi})$-reversal-bounded run ρ of $\mathcal{S}$ from $(q, \boldsymbol{x})$ such that $\rho, 0 \models \phi$;*

(II) *there is an infinite r-$\mathtt{T}_{\mathcal{S}'}$-reversal-bounded run from $((q, X_0, 0), \boldsymbol{x})$ such that $(q_f, X_f, 0)$ is repeated infinitely often for some* initial *atom $X_0 \in Atoms(\phi)$ and for some $(q_f, X_f) \in Q \times Atoms(\phi)$.*

Proof. Let ρ be an infinite r-$(\mathtt{T}_{\mathcal{S}} \cup \mathtt{T}_{\phi})$-reversal-bounded run ρ of $\mathcal{S}$ from $(q, \boldsymbol{x})$ such that $\rho, 0 \models \phi$. Suppose that ρ is of the form $(q_0, \boldsymbol{x}_0), (q_1, \boldsymbol{x}_1), \ldots$ and it is induced by the sequence of transitions $t_0 t_1 t_2 \cdots$ and each transition t_i is of the form $q_i \xrightarrow{(\xi_i, \boldsymbol{b}_i)} q_{i+1}$. We define a function $proj_A(\rho) : \mathbb{N} \to Q \times valid(A)$ which associates, to each position of the time, the control state of the generalized automaton (accepting the run) satisfying the current configuration of ρ defined by counter system $\mathcal{S}$. In particular, $proj_A(\rho)$ is such that, for any position $i \geq 0$, the element $proj_A(\rho)(i) = (q_i, B)$, we have $\xi \in B$ if, and only if, $\rho, i \models \xi$ for all formulae $\xi \in A$. By definition of $\mathcal{A}_{\phi}$, we have $proj_A(\rho) \in \mathrm{L}(\mathcal{A}_{\phi})$.

By means of $proj_A$ we can associate to ρ a synchronized run $\rho_{\mathcal{A}_{\phi}}$ of the automaton $\mathcal{A}_{\phi}$ which is, by construction, such that $proj_A(\rho), 0 \models \phi$.

$$\begin{array}{rccccccccc} \rho = & (q_0, \boldsymbol{x}_0) & \xrightarrow{(\xi_0, \boldsymbol{b}_0)} & (q_1, \boldsymbol{x}_1) & \xrightarrow{(\xi_1, \boldsymbol{b}_1)} & \ldots & (q_i, \boldsymbol{x}_i) & \xrightarrow{(\xi_i, \boldsymbol{b}_i)} & \\ \rho_{\mathcal{A}_{\phi}} = & (X_0, 0) & \xrightarrow{(q_0, \beta_0)} & (X_1, a_1) & \xrightarrow{(q_1, \beta_1)} & \ldots & (q_i, a_i) & \xrightarrow{(X_i, \beta_i)} & \ldots \end{array}$$

where $q_0 = q$, $\boldsymbol{x}_0 = \boldsymbol{x}$, X_0 is an initial atom and for all $i \geq 0$, $\boldsymbol{x}_i \models_{\mathrm{PA}} \xi_i$ and $\boldsymbol{x}_i \models_{\mathrm{PA}} X_i(\boldsymbol{b}_i)$. Moreover, since $proj_A(\rho), 0 \models \phi$, i.e., $proj_A(\rho)$ is an accepting run of $\mathcal{A}_{\phi}$, there is an atom $X_f \in atoms(\phi)$ such that the control state $(X_f, 0)$ in $\mathcal{A}_{\phi}$ is repeated infinitely often. By merging these

two ω-sequences, we have an r-$\mathtt{T}_{\mathcal{S}'}$-reversal-bounded run for $\mathcal{S}'$:

$$\begin{array}{llll} ((q_0, X_0, 0), \boldsymbol{x}_0) & \xrightarrow{\xi_0 \wedge X_0(\boldsymbol{b}_0)} & ((q_1, X_1, 1), \boldsymbol{x}_1) & \xrightarrow{\xi_1 \wedge X_1(\boldsymbol{b}_1)} \quad \ldots \\ & \ldots & ((q_i, X_i, a_i), \boldsymbol{x}_i) & \xrightarrow{\xi_i \wedge X_i(\boldsymbol{b}_i)} \quad \ldots \end{array}$$

where $(q_f, X_f, 0)$ is repeated infinitely for some $q_f \in Q$. All terms occuring in guards of the form $\xi_i \wedge X_i(\boldsymbol{b}_i)$ belong to $\mathtt{T}_{\mathcal{S}'}$ (ξ_i are formulae of $\mathcal{S}$ and $X_i(\boldsymbol{b}_i)$ are sets of terms involved in ϕ).

Conversely, let us suppose that there is an infinite r-$\mathtt{T}_{\mathcal{S}'}$-reversal-bounded run from $((q, X_0, 0), \boldsymbol{x})$ such that $(q_f, X_f, 0)$ is repeated infinitely often for some *initial* atom $X_0 \in atoms(\phi)$ and for some $(q_f, X_f) \in Q \times atoms(\phi)$. The run ρ is of the form:

$$((q_0, X_0, 0), \boldsymbol{x}_0), ((q_1, X_1, a_1), \boldsymbol{x}_1), \ldots$$

and it is induced by a sequence of transitions $t_0 t_1 t_2 \cdots$ such that each t_i is of the form

$$(q_i, X_i, a_i) \xrightarrow{(\xi_i \wedge X_i(\boldsymbol{b}_i), \boldsymbol{b}_i)} (q_{i+1}, X_{i+1}, a_{i+1}).$$

Now, we can filter the information concerning

- counter system, by projecting configurations of the form $(q, \boldsymbol{x})$
- the automaton for ϕ, by projecting atoms and values in $[0, m]$.

By projecting on Q, the run $(q_0, \boldsymbol{x}_0), (q_1, \boldsymbol{x}_1), \ldots$ is an r-$(\mathtt{T}_{\mathcal{S}} \cup \mathtt{T}_{\phi})$-reversal-bounded run of $\mathcal{S}$. By projecting on $Atoms(\phi) \times [0, m]$, we obtain a ω-word

$$(q_0, \beta_0)(q_1, \beta_1) \ldots (q_i, \beta_i) \ldots$$

which belongs to $\mathscr{L}(\mathcal{A}_\phi)$. By definition of $\mathcal{S}^\times$, each configuration is such that $\boldsymbol{x}_i \models_{\mathrm{PA}} \xi_i \wedge X_i(\boldsymbol{b}_i)$. Therefore, the projected run over configuration is a model for ϕ, i.e., $(q_0, \boldsymbol{x}_0), (q_1, \boldsymbol{x}_1), \ldots, 0 \models \phi$. □

As for propositional LTL, the construction defined to obtain $\mathcal{A}_\phi$ is done by considering at most an exponential number of control states with respect to the size of the formulae.

Corollary 74 (Corollary 2, [9]). *There is a polynomial-space reduction from* RBMC *into* RB-REP-REACH(CS(QFP)).

4.3.2. Removing periodicity constraints

In this section, we show that given $\mathrm{L} \subseteq \mathrm{QFP}$ using periodicity constraints of the form $t \equiv_c k$, the reversal-bounded reachability problem for counter systems in CS(L) can be reduced to the corresponding problem restricted to counter systems in CS(L$'$), where L$'$ is the restriction of L without periodicity constraints.

Let us consider the class of counter systems CS(L). The underlying idea to remove periodicity constraints consists in defining a new counter system $\mathcal{S}' \in \mathrm{CS(L')}$ from a given $\mathcal{S} \in \mathrm{CS(L)}$, whose control states store counter values modulo C, where C is the least common multiple of all the constants c appearing in atomic formulae of the form $t \equiv_c k$ in guards of $\mathcal{S}$. The number of control states in $\mathcal{S}'$ is equal to the number of control states in $\mathcal{S}$ multiplied by C, which is in $\mathcal{O}(2^{N^2})$. Hence, this construction entails an exponential blow-up of the number of control states of the new counter system $\mathcal{S}'$. Transitions of $\mathcal{S}'$ are defined accordingly to the update operations on them in order to correctly represent the classes of modulo for each counter. The justification for using value C is given in Appendix A.3. Here, we give an example of the notion we require. Let $2x + 5y \equiv_b a$ be a periodic constraint over a transition starting from q. It is easy to see that, $2x + 5y \equiv_b a$ can be replaced by $\top$ if the new control state q' derived from q in $\mathcal{S}'$ is representative of the class $x \equiv_C c$ and $y \equiv c'$ such that $2c + 5c' \equiv_b a$. Intuitively, periodicity constraints can be removed only by looking at the class of modulo $\equiv_c$ stored within control states of $\mathcal{S}'$.

Let $\mathcal{S}' = (Q', n, \delta')$ be the counter system where $Q' = Q \times [0, C-1]^n$. Given $\boldsymbol{x} \in \mathbb{N}^n$, we write $\tilde{\boldsymbol{x}}$ to denote the unique tuple in $[0, C-1]^n$ such that for $i \in [1, n]$, we have $\boldsymbol{x}(i) \equiv_C \tilde{\boldsymbol{x}}(i)$. The set config_{ok} of valid configuration of $\mathcal{S}'$ are pair of the form $((q, \tilde{\boldsymbol{x}}), \boldsymbol{y})$ such that $\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{x}}$. In order to correctly map configuration $(q, \boldsymbol{x})$ to configuration of the new system $((q, \tilde{\boldsymbol{x}}), \boldsymbol{y})$ we use a function $f : (Q \times \mathbb{N}^n) \to \mathrm{config}_{ok}$ such that $f((q, \boldsymbol{x})) = ((q, \tilde{\boldsymbol{x}}), \boldsymbol{x})$. f and f^{-1} extend naturally to sequences (either finite or infinite ones). Transition relation δ' is defined as follows: if $q \xrightarrow{(\phi, \boldsymbol{b})} q' \in \delta$ then $(q, \tilde{\boldsymbol{x}}) \xrightarrow{(\phi', \boldsymbol{b})} (q', \tilde{\boldsymbol{y}}) \in \delta'$ for all tuples $\tilde{\boldsymbol{x}}$, $\tilde{\boldsymbol{y}}$, such that for $i \in [1, n]$, $\tilde{\boldsymbol{y}}(i) \equiv_C \tilde{\boldsymbol{x}}(i) + \boldsymbol{b}(i)$ and ϕ' is defined from ϕ by substituting its subformulae of the form $\sum_j a_j x_j \equiv_c k$ by:

$$\begin{cases} \top, \text{ when } \displaystyle\sum_j a_j \boldsymbol{x}(j) \equiv_c k \\ \bot, \text{ otherwise.} \end{cases} \tag{4.1}$$

Removing periodicity constraints preserves runs of systems, i.e, there exists a map between runs of $\mathcal{S}$ and $\mathcal{S}'$.

Lemma 75 (Lemma 2, [9]). *Let $\mathcal{S} = (Q, n, \delta)$ be a counter system in* CS(QFP) *and $\mathcal{S}' = (Q', n, \delta')$ be the counter system in* CS(QFP(<)) *defined as above.*

(I) *For every run ρ of $\mathcal{S}$, $f(\rho)$ is also a run of $\mathcal{S}'$.*

(II) *For every run ρ of $\mathcal{S}'$ such that the first configuration belongs to* config_{ok}*, then all configurations in ρ belong to* config_{ok} *and $f^{-1}(\rho)$ is also a run of $\mathcal{S}$.*

Proof. (I) Let ρ be a run $\mathcal{S}$ of the form $(q_0, \boldsymbol{x}_0), (q_1, \boldsymbol{x}_1), \cdots$ and it is induced by the sequence of transitions $t_1 t_2 t_3 \ldots$ and each transition t_i is of the form $q_{i-1} \xrightarrow{(\xi_i, \boldsymbol{b}_i)} q_i$. From ρ we can build a run for $\mathcal{S}'$, $f(\rho) = ((q_0, \tilde{\boldsymbol{x}_0}), \boldsymbol{x}_0), ((q_1, \tilde{\boldsymbol{x}_1}), \boldsymbol{x}_1), \ldots$ which is is induced by the sequence $t'_1 t'_2 t'_3 \cdots$ where t'_i is of the form $(q_{i-1}, \tilde{\boldsymbol{x}_{i-1}}) \xrightarrow{(\xi'_i, \boldsymbol{b}_i)} (q_i, \tilde{\boldsymbol{x}_i})$ and ξ'_i is defined according to the rule (4.1). $f(\rho)$ is a run of $\mathcal{S}$ by the property (PER) in Appendix (A.3) because in $f(\rho)$, $\boldsymbol{x}(i) \equiv_C \tilde{\boldsymbol{x}}(i)$ holds for $i \in [1, n]$ by construction and, therefore, $t(\boldsymbol{x}) \equiv_C k$ if, and only if, $t(\tilde{\boldsymbol{x}}) \equiv_C k$.

(II) Similarly, let $\rho = ((q_0, \tilde{\boldsymbol{x}_0}), \boldsymbol{x}_0), ((q_1, \tilde{\boldsymbol{x}_1}), \boldsymbol{x}_1), \cdots$ be a run such that the configuration $((q_0, \tilde{\boldsymbol{x}_0}), \boldsymbol{x}_0)$ belongs to config_{ok}. By definition, all the configurations in ρ belong to config_{ok} and $f^{-1}(\rho)$ is the sequence $(q_0, \boldsymbol{x}_0), (q_1, \boldsymbol{x}_1), \cdots$. The sequence ρ is induced by the sequence $t_1 t_2 t_3 \ldots$ where each transition t_i is of the form $(q_{i-1}, \tilde{\boldsymbol{x}_{i-1}}) \xrightarrow{(\xi'_i, \boldsymbol{b}_i)} (q_i, \tilde{\boldsymbol{x}_i})$ and ξ'_i is defined by the rule (4.1). Let t_i be a transition of the form $q_{i-1} \xrightarrow{(\xi_i, \boldsymbol{b}_i)} q_i$. By construction, $f^{-1}(\rho)$ is a run of $\mathcal{S}$ by the property (PER) which is induced by the sequence of transitions $t_1 t_2 t_3 \cdots$. □

Therefore, previous lemma allow us to determine that removing periodicity from formulae of transitions can be done in polynomial space in the dimension of systems. The following corollary states this complexity issue which is useful for final analysis of complexity in Section 4.4.

Corollary 76 (Corollary 2, [9]). *Let* $\mathrm{L} = \mathrm{QFP}$ *[resp.* $\mathrm{L} = \mathrm{QFP}(<_1, \equiv)$*]* *and* $\mathrm{L}' = \mathrm{QFP}(<)$ *[resp.* $\mathrm{L}' = \mathrm{QFP}(<_1)$*].*

(I) *There is a polynomial-space reduction from* RB-REACH(CS(L)) *to* RB-REACH(CS(L′)).

(II) *There is a polynomial-space reduction from* RB-REP-REACH(CS(L)) *to* RB-REP-REACH(CS(L′)).

Proof. Let $\mathcal{S} = (Q, n, \delta)$ be a counter system in CS(L) and $\mathcal{S}' = (Q', n, \delta')$ be the counter system in CS(L′) derived from $\mathcal{S}$ by using the procedure

explained above. Given $\rho = (q_0, \boldsymbol{x}_0), (q_1, \boldsymbol{x}_1) \dots$ a run of $\mathcal{S}$, then there exists in $\mathcal{S}'$ an equivalent run ρ' such that $\rho' = f(\rho)$ and $f^{-1}(\rho') = \rho$, by from Lemma 75. If $(q, \boldsymbol{x})$ is a configuration such that $(q_0, \boldsymbol{x}_0) \xrightarrow{*} (q, \boldsymbol{x})$ then there exists an equivalent configuration $f((q, \boldsymbol{x})) = ((q, \tilde{\boldsymbol{x}}), \boldsymbol{x})$ in $\mathcal{S}'$ such that $((q_0, \tilde{\boldsymbol{x}}_0), \boldsymbol{x}_0) \xrightarrow{*} ((q, \tilde{\boldsymbol{x}}), \boldsymbol{x})$. Then, an instance $(\mathcal{S}, r, (q_0, \boldsymbol{x}_0), (q, \boldsymbol{x}))$ of a RB-REACH(CS(L)) can be reduced to $(\mathcal{S}', r, f((q_0, \boldsymbol{x}_0)), f((q, \boldsymbol{x})))$.

The control state repeated reachability is reduced in the same way. Let $\rho = (q_0, \boldsymbol{x}_0), (q_1, \boldsymbol{x}_1), \dots (q, \boldsymbol{x}), \dots (q, \boldsymbol{x}') \dots$ be an infinite run of $\mathcal{S}$ such that q is repeated infinitely often. Then, there exists in $\mathcal{S}'$ an run ρ' such that $f(\rho) = \rho'$ and $((q, \tilde{\boldsymbol{x}}), \boldsymbol{x})$, $((q, \tilde{\boldsymbol{x}}'), \boldsymbol{x}')$, $\dots$ are configurations such that q is repeated infinitely often. Since the number of vectors $\tilde{\boldsymbol{x}}$ is finite (i.e. $(C-1)^n$ where n is the number of counters of $\mathcal{S}$) then there exists at least one configuration $(q, \tilde{\boldsymbol{x}})$ of $\mathcal{S}'$ which is repeated infinitely often. Then, an instance $(\mathcal{S}, k, (q_0, \boldsymbol{x}_0), q)$ of RB-REP-REACH(CS(L)) can be reduced to a finite amount of instances of RB-REP-REACH(CS(L′)), $(\mathcal{S}', k, f((q_0, \boldsymbol{x}_0)), (q, \tilde{\boldsymbol{x}}))$ for all $(q, \tilde{\boldsymbol{x}})$ where $\tilde{\boldsymbol{x}} \in \{0 \dots C-1\}^n$. □

After removing periodicity constraints we have to provide the reduction from a repeated reachability problem to a reachability problem over the class of counter systems in CS(QFP(<)). By exploiting the reversal-boundedness property it is possible to show that infinite runs which visit a control state q infinitely often can be discovered by looking at finite runs witnessing the repetition of q.

4.3.3. From Repeated Reachability to Reachability

The existence of an *infinite* run can be decided by means of the existence of a *finite* run satisfying additional properties. It is worthy to be noticed that such a reduction is not possible with nondeterministic Minksy machines without the reversal-boundedness assumption on terms. The intuition behind the need of additional properties comes from the fact that finite runs become witness of an infinite computation. After r reversals, formulae of $\mathcal{S}$ occurring in transitions over an r-T-reversal-bounded run, stabilize their logical value. In particular, counters are either constant or monotonic increasing whereas terms can be either increasing or decreasing. This defines a partition of the set of counters and terms distinguishing constant or monotonic increasing counters and constant or monotonic increasing/decreasing terms. Beside this condition, we have to verify whether there exists a control state q_f which is visited infinitely often witnessing the Büchi acceptance condition. Since we are looking for a finite run, we have to check whether q_f is repeated twice, similarly to what

we do for bounded model-checking: the projection of the finite run over control states of $\mathcal{S}$ has the form $q_0 q_1 \dots q_{loop-1} q_{loop} v q_{loop}$ which witnesses an infinite ultimately periodic run of the form $q_0 q_1 \dots q_{loop-1}(q_{loop} v)^\omega$. Periodicity of finite runs is fundamental in our approach because, as anticipated in Section 4.1 at the beginning of the chapter, ultimately periodic runs can be easily discovered by encoding the transition relation of counter systems beside some additional constraints over SMT-solver, by taking advantage of technique presented in Chapter 3. Now, we are ready to give details on the equivalence between r-T-reversal-bounded runs and their finite ultimately periodic representation. Let us observe that the additional constraints we are going to define are an immediate consequence of the reversal-boundedness property of runs. Let $\mathcal{S}$ be a counter system in CS(QFP(<)) (then, without periodicity constraints), q_f be a control state and $r \geq 0$. We define K_{max} (respectively K_{min}) be the maximum (respectively the minimum) constant value k occurring in formulae $t \sim k$ over transitions. The set of counters and terms is partitioned as follows: the set of constant counters by the set $\mathtt{Z}_{\rightarrow} \subseteq [1, n]$. The set of constant terms is $\mathtt{T}_{\rightarrow}$ and the set of monotonic increasing/decreasing terms by $\mathtt{T}_{\nearrow}/\mathtt{T}_{\searrow}$.

(⋆) There is an infinite r-$\mathtt{T}_{\mathcal{S}}$-reversal-bounded run from $(q_0, \boldsymbol{x}_0)$ such that q_f is repeated infinitely often.

(⋆⋆) There exist a finite run

$$(q_0, \boldsymbol{x}_0), \dots, (q_{l'}, \boldsymbol{x}_{l'}) \dots (q_l, \boldsymbol{x}_l)$$

where $l' < l$, $j \in [l'+1, l]$, $\mathtt{Z}_{\rightarrow} \subseteq [1, n]$ and $\mathtt{T}_{\rightarrow}, \mathtt{T}_{\searrow}, \mathtt{T}_{\nearrow}$ partitions $(\mathtt{T}_{\mathcal{S}} \setminus \{x_1, \dots, x_n\})$ such that:

1. $q_{l'} = q_l = q_f$ and $(q_0, \boldsymbol{x}_0), \dots, (q_l, \boldsymbol{x}_l)$.
2. $\boldsymbol{x}_j(i) - \boldsymbol{x}_{j-1}(i) = 0$, for $i \in \mathtt{Z}_{\rightarrow}$,
3. $\boldsymbol{x}_j(i) - \boldsymbol{x}_{j-1}(i) \geq 0$, for $i \in [1, n] \setminus \mathtt{Z}_{\rightarrow}$,
4. $\boldsymbol{x}_{l'}(i) \geq K_{max}$, for $i \in [1, n] \setminus \mathtt{Z}_{\rightarrow}$,
5. $t(\boldsymbol{x}_j) - t(\boldsymbol{x}_{j-1}) = 0$, for$t \in \mathtt{T}_{\rightarrow}$,
6. $t(\boldsymbol{x}_j) - t(\boldsymbol{x}_{j-1}) \leq 0$, for $t \in \mathtt{T}_{\searrow}$,
7. $t(\boldsymbol{x}_j) - t(\boldsymbol{x}_{j-1}) \geq 0$, for $t \in \mathtt{T}_{\nearrow}$,
8. For $t \in \mathtt{T}_{\searrow}$, $t(\boldsymbol{x}_{l'}) \leq K_{min}$.
9. For $t \in \mathtt{T}_{\nearrow}$, $t(\boldsymbol{x}_{l'}) \geq K_{max}$.

Lemma 77 (Lemma 3, [9]). *(⋆) is equivalent to (⋆⋆)*

Proof. ($\star$) implies ($\star\star$). Let $(q_0, \boldsymbol{x}_0), (q_1, \boldsymbol{x}_1), \ldots$ be an infinite r-$\mathtt{T}_{\mathcal{S}}$-reversal-bounded run from $(q_0, \boldsymbol{x}_0)$ such that q_f is repeated infinitely often (with $\mathtt{T}_{\mathcal{S}} = \{x_1, \ldots, x_n\} \cup \{t_1, \ldots, t_{n'}\}$). All the atomic guards in $\mathcal{S}$ are of the form $t \sim k$ with $t \in \mathtt{T}_{\mathcal{S}}$ and $k \in [K_{min}, K_{max}]$.

- Let $i \in [1, n]$ be an index belonging to the set of counters. From some position, the value of counter i either remains constant or it diverges to $+\infty$ and the update values (on counter i) are always greater than 0. In the first case, condition 2 of $\star\star$ is witnessed. In the second case, monotonic counters are such that $\boldsymbol{x}_j(i) - \boldsymbol{x}_{j+1}(i) \geq 0$ (condition 3) and there is a position j_1 such that for $j \geq j_1$, $\boldsymbol{x}_j(i) \geq K_{max}$, for all $i \in [1, n] \setminus \mathtt{Z}_{\rightarrow}$.
- Let $i \in [1, n']$ be an index belonging to the set of terms. Because the term t_i has a bounded number of reversals, one of the following conditions holds:
  - From some position, the value of the term t_i remains constant, i.e. there is $j_0 \in \mathbb{N}$, such that for $j \geq j_0$, $t_i(\boldsymbol{x}_{j+1}) - t_i(\boldsymbol{x}_j) = 0$.
  - From some position, the value of the term t_i diverges to $-\infty$ and there is $j_0 \in \mathbb{N}$, such that for $j \geq j_0$, $t_i(\boldsymbol{x}_{j+1}) - t_i(\boldsymbol{x}_j) \leq 0$ and there is a position $j_1 \geq j_0$ such that $t_i(\boldsymbol{x}_{j_1}) \leq K_{min}$.
  - From some position, the value of the term t_i diverges to $+\infty$ and there is $j_0 \in \mathbb{N}$, such that for $j \geq j_0$, $t_i(\boldsymbol{x}_{j+1}) - t_i(\boldsymbol{x}_j) \geq 0$ there is a position $j_1 \geq j_0$ such that $t_i(\boldsymbol{x}_{j_1}) \geq K_{max}$.
- There is a control state q_f which is repeated infinitely often.

Let j a position such that all previous conditions hold, i.e., all counters and terms stabilize their behavior. After this position, the partition $\mathtt{T}_{\rightarrow}, \mathtt{T}_{\searrow}, \mathtt{T}_{\nearrow}$ is defined. Therefore, we can identify two occurrences of q_f, after position j, which delimit the sequence of configurations $(q_{l'}, \boldsymbol{x}) \ldots (q_l, \boldsymbol{x}')$ such that $q_f = q_{l'} = q_l$ witnessing condition 1 and such that $\boldsymbol{x}_{l'} \geq K_{max}$, and $t(\boldsymbol{x}_{l'}) \geq K_{max}$, for terms $t \in \mathtt{T}_{\nearrow}$, $t(\boldsymbol{x}_{l'}) \leq K_{min}$, for terms $t \in \mathtt{T}_{\searrow}$ (witnessing conditions $4 - 9$).

($\star\star$) implies ($\star$). Let ρ be a finite run

$$(q_0, \boldsymbol{x}_0) \xrightarrow{t_1} (q_1, \boldsymbol{x}_1) \cdots \xrightarrow{t_{l'}} (q_{l'}, \boldsymbol{x}_{l'}) \cdots \xrightarrow{t_l} (q_l, \boldsymbol{x}_l)$$

satisfying conditions $1 - 9$. For each transition t_i, we assume that the guard is ϕ_i and the update vector is $\boldsymbol{b}_i$. The finite sequence of transitions $t_1 t_2, \ldots, t_{l'} \ldots, t_l$ is a witness of an infinite sequence $t_1 t_2, \ldots, t_{l'} (t_{l'+1} \ldots, t_l)^{\omega}$ by repeating infinitely many time the suffix $t_{l'+1} \ldots, t_l$. We show that

we still obtain an infinite r-T-reversal-bounded run. Let us consider the infinite sequence of configurations below

$$\rho' = (q_0, \boldsymbol{x}_0) \xrightarrow{t_1} (q_1, \boldsymbol{x}_1) \cdots \xrightarrow{t_{l'}} (q_{l'}, \boldsymbol{x}_{l'}) \cdots \xrightarrow{t_l} (q_l, \boldsymbol{x}_l) = (q_l, \boldsymbol{y}_l) \xrightarrow{t_{l'+1}} \cdots$$
$$\xrightarrow{t_{l'+1}} (q_{l'+1}, \boldsymbol{y}_{l+1}) \cdots \xrightarrow{t_l} (q_l, \boldsymbol{y}_{l+(l-l')}) \cdots$$

Now, we give a rule to define values for counters. The rule exploits reversal-boundeness property guaranteeing that the behavior of counters is infinitely stable from a position onwards and which can be realized by iterating infinitely many times the updates of sequence $t_{l'+1} \ldots, t_l$. It is worth noting that guards $t \sim k$ over transitions have constant logical value. For instance, let $t \in \mathtt{T}_{\nearrow}$ and $t < k$ be a formula of guard ϕ at position l. Let us suppose that $t(\boldsymbol{x}_l) < k$ does not holds, by 8. Therefore, since all the updates are fixed between l' and l, we have that $t(\boldsymbol{x}_{l+(l-l')}) < k$ does not hold because $\boldsymbol{x}_{l+(l-l')} = \boldsymbol{x}_l + (\boldsymbol{x}_l - \boldsymbol{x}_{l'})$, where the difference $(\boldsymbol{x}_l - \boldsymbol{x}_{l'})$ is constant. The rule is defined as follows:

$$\boldsymbol{y}_{l+u(l-l')+u'} = \boldsymbol{x}_{l+u'} + u(\boldsymbol{x}_l - \boldsymbol{x}_{l'})$$

for $k \geq 0$ and $k' \in [0, l-l'-1]$. From previous considerations, we can see that ρ' is a run such that q_f is repeated infinitely often and the sequence of transitions is an ultimately periodic word. Moreover, $\boldsymbol{y}_{l+u(l-l')+u'} \models \phi_{l'+1+u'}$ since $\boldsymbol{x}_{l+u'} \models \phi_{l'+1+u'}$, for $u \geq 0$ and $u' \in [0, l-l'-1]$; and it is r-$\mathtt{T}_{\mathcal{S}}$-reversal-bounded since, after position l', all terms can not have reversals. □

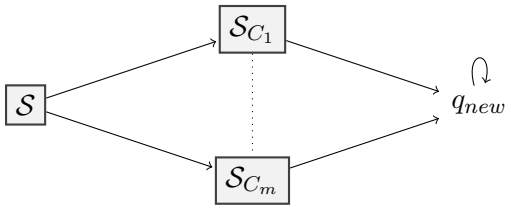
By means of the reduction given in the following theorem, we are able to capture condition $\star\star$. The intuition behind it is to build a new counter system $\mathcal{S}'$ from the original one $\mathcal{S}$ which reaches a final control state if, and only if, condition $\star\star$ is satisfied. The new system consists of as many copy of $\mathcal{S}$ as the number of partitions of T such that each copy enforces the realization of one partition.

Theorem 78 (Theorem 1, [9]). *There is a polynomial-space many-one reduction from* RB-REP-REACH(CS(L)) *into* RB-REACH(CS(L)), *where* L *is* QFP($<$).

Proof. Let $\mathcal{S}$ be in CS(QFP($<$)), $(q_0, \boldsymbol{x}_0)$ be an initial configuration, q_f be a control state and $r \geq 0$. The proof consists in the following reduction: given an instance of repeated reachability, we define an instance of RB-REACH(CS(QFP($<$))) over a new counter automaton $\mathcal{S}' = (Q', n, \delta')$ in CS(QFP($<$)) enforcing condition ($\star\star$). The new system $\mathcal{S}'$ is made of the original version of $\mathcal{S}$ endowed with an exponential

number (in the number of counters and terms) of modified copies of $\mathcal{S}$ each corresponding to a partition $C = (\mathtt{Z}_{\rightarrow}, \mathtt{T}_{\rightarrow}, \mathtt{T}_{\searrow}, \mathtt{T}_{\nearrow})$, over counters and terms, plus a new fresh control state q_{new}. The structure of counter system $\mathcal{S}'$ is provided by the next figure:

Figure 4.2.: Structure of system $\mathcal{S}'$



Condition ($\star\star$) holds if, and only if, there is an $(r+1)$-$\mathtt{T}_{\mathcal{S}'}$-reversal-bounded run of $\mathcal{S}'$ from $(q_0, \boldsymbol{x}_0)$ to $(q_{new}, \mathbf{0})$. A S_C system is obtained from $\mathcal{S}$ by keeping only transitions with update vector $\boldsymbol{b}$ satisfying the following conditions which are defined in function of partition C. Informally, after r reversals, if the i-th counter is constant, i.e., $i \in \mathtt{Z}_{\rightarrow}$ within C, then only transitions of $\mathcal{S}$ such that $\boldsymbol{b}(i) = 0$ are retained in the transition relation of S_C. Otherwise, all transitions with $\boldsymbol{b}(i) \geq 0$ are preserved.

condition C:

1. for $i \in \mathtt{Z}_{\rightarrow}$, $\boldsymbol{b}(i) = 0$;
2. for $i \notin \mathtt{Z}_{\rightarrow}$, $\boldsymbol{b}(i) \geq 0$.

Similarly, the same applies to terms. The condition is captured by the following condition T.

condition T:

1. for $t \in \mathtt{T}_{\rightarrow}$, $t(\boldsymbol{b}) = 0$;
2. for $t \in \mathtt{T}_{\searrow}$, $t(\boldsymbol{b}) \leq 0$;
3. for $t \in \mathtt{T}_{\nearrow}$, $t(\boldsymbol{b}) \geq 0$.

For instance, let $t = a\boldsymbol{x}(i) + c\boldsymbol{x}(j)$ be a term in $\mathtt{T}_{\nearrow}$ and x, x' be the instance of x at current and next position. Because of the increasing monotonicity of t, then $a(\boldsymbol{x}'(i) - \boldsymbol{x}(i)) + c(\boldsymbol{x}'(j) - \boldsymbol{x}(j)) \geq 0$ where $\boldsymbol{x}'(i) - \boldsymbol{x}(i)$ equals to the increment $\boldsymbol{b}(i)$. Therefore, $a(\boldsymbol{x}'(i) - \boldsymbol{x}(i)) + c(\boldsymbol{x}'(j) - \boldsymbol{x}(j)) \geq 0$ that is $a\boldsymbol{b}(i) + c\boldsymbol{b}(j) \geq 0$ therefore $t(\boldsymbol{b}) \geq 0$. Analogously, we derive $t(\boldsymbol{b}) = 0$ and $t(\boldsymbol{b}) \leq 0$ for the other two cases. Previous conditions enforce condition $2, 3$ on counters and $5, 6, 7$ on terms of $\star\star$. Now, we have to give other condition on $\mathcal{S}'$ to realize periodicity of q_f (condition 1)

and to check absolute values of counters and terms. In order to simulate the subrun $(q_{l'}, \boldsymbol{x}_{l'}) \cdots (q_l, \boldsymbol{x}_l)$, such that $q_f = q_l = q_{l'}$, from the original copy, the system moves nondeterministically from q_f of the original copy to a control state q belonging to some S_C system in $\mathcal{S}'$ such that there exists $q_f \xrightarrow{\phi, \boldsymbol{b}} q$ in $\mathcal{S}$. The move chooses which partition C is enforced and it is realized by means of a transitions $q_f \xrightarrow{\phi', \boldsymbol{b}} q$ such that $\boldsymbol{b}$ satisfies previous conditions on counters and terms where ϕ' is endowed with the following guards:

condition G:

1. $\boldsymbol{x}(i) \geq K_{max}$ for $i \in [1, n] \setminus \mathtt{Z}_{\rightarrow}$ (condition 4);
2. $t_i \leq K_{min}$ for $i \in \mathtt{T}_{\searrow}$ (condition 8);
3. $t_i \geq K_{max}$ for $i \in \mathtt{T}_{\nearrow}$ (condition 9).

To enforce repetition of q_f we define an empty transition $q_f \xrightarrow{\top, \mathbf{0}} q_{new}$ from every q_f in each C-copy which nondeterministically jumps to the new accepting control state q_{new}. Self-loops of the form $q_{new} \xrightarrow{\phi'', \boldsymbol{b}_i} q_{new}$ such that ϕ'' is $\boldsymbol{x}(i) > 0$ and $\boldsymbol{b}(i) = -1$ and $\boldsymbol{b}(j) = 0$, with $i \neq j$ for all $i, j \in [1, n]$, are needed to decrement all counters. Observe that:

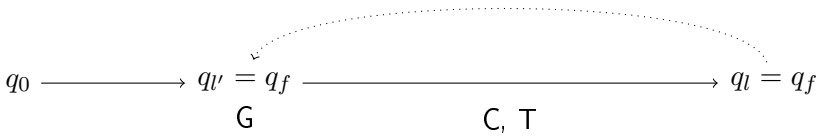
- all previous additional formulae required to guarantee conditions $1 - 9$ still belong to QFP($<$). They do not involve new terms, i.e., $\mathtt{T}_{\mathcal{S}'} = \mathtt{T}_{\mathcal{S}}$, both the systems, $\mathcal{S}$ and $\mathcal{S}'$, have the same set of constants k.
- The numbers of states of $\mathcal{S}'$ is bounded by $\text{card}(Q) \times (1 + 2^n \times (2^{n'} \times 2^{n'})) + 1$ (with $\text{card}(\mathtt{T}_{\mathcal{S}}) = n + n'$).

□

4.3.4. Ultimately periodic runs

The reduction defined by Theorem (78) is fundamental to provide the next theorem. By construction, we are able to characterize the existence of a run satisfying a property ϕ by means of the existence of an ultimately periodic run which is still a model for ϕ. This fact is a direct consequence of the Büchi acceptance condition requiring repetition of an accepting state and of reversal-boundedness of terms. The figure below shows the structure of finite prefixes representing the ultimately periodic runs obtained by construction from $\mathcal{S}'$ defined in the proof of Theorem (78). Conditions C, T, G restricting the set of runs only to the ultimately periodic ones are placed over positions where they hold.

Figure 4.3.: Ultimately periodic runs and conditions



Corollary 79 (Corollary 3, [9]). *Let $\mathcal{S}$ be in* CS(QFP), *$(q, \boldsymbol{x})$ be an initial configuration, ϕ be in* CLTL(QFP) *and $r \in \mathbb{N}$. The propositions below are equivalent:*

(I) *there is an infinite run ρ from $(q, \boldsymbol{x})$ such that $\rho, 0 \models \phi$ and ρ is r-T-reversal-bounded with $\mathsf{T} = \mathsf{T}_{\mathcal{S}} \cup \mathsf{T}_{\phi}$.*

(II) *There exists an* ultimately periodic *run ρ satisfying the same properties as in (I).*

Corollary 79 provides the theoretic foundation which allow us to justify the use of bounded approaches like k-bounded satisfiability to solve RBMC. k-bounded satisfiability is effective in solving RBCM because it cap capture ultimately periodic runs and conditions C, T and G.

4.4. Complexity and effective Presburger-definability

In this section we provide complexity analysis of relevant problems we consider so far. We prove that

- RB-REACH(QFP), RB-REP-REACH(QFP) and RBMC are NExpTIME-complete and
- the sets of initial configurations satisfying the properties related to these problems (witness run properties) are effectively definable in Presburger arithmetic.

It is worth to be noticed that the foundation of proof for NExpTIME upper bound of RB-REACH is based on a proof technique analyzing runs used by Rackoff in [86]. Runs of counter systems are partitioned into subruns satisfying suitable conditions on the behavior of terms and counters which are defined with respect to the number of reversal already performed and the absolute value compared with constants involved into formulae over transitions. By partitioning a run we obtain a symbolic representation that we use to build a system of equations defining each

phase and, therefore, the whole run, where variables of equations count the number of iteration performed by loops of transitions within each phase. The existence of small solutions for integer (inequality) systems [87] by Borosh and Treybig allow us to define exactly the smallest instance of legal runs satisfying the partition used to define phases. The length of such runs is proved to be of double exponential magnitude with respect to the size of the RB-REACH problem, where all constants are encoded in binary. Small run property can be used to define a nondeterministic algorithm which guesses such small (r-T-reversal-bounded) runs in (nondeterministic) exponential time. The existence of small run solutions characterizing small runs makes k-bounded approach complete with respect to RBMC. Complexity analysis for RB-REP-REACH and RBMC is derived as consequence from NEXPTIME-completeness of RB-REACH. In fact, reductions from Corollary 74 of RBMC to RB-REP-REACH and from Theorem 78 of RB-REP-REACH to RB-REACH, as well as Corollary 76, concerning periodicity constraints, preserve the double exponential length of small runs. Proofs of next theorems are taken from [9] but here we present a summarized version. Before stating the theorem we provide some essential definitions. Let $\mathcal{S} = (Q, n, \delta)$ be a counter system in CS(QFP($<$)).

- Let AG be the set, closed under negation, of (conjunctions of) atomic arithmetical formulae occurring in $\mathcal{S}$.
- Given $Y \subseteq AG$ and $q \xrightarrow{(\phi,\boldsymbol{b})} q'$, we write $Y \models \phi$ whenever
  1. $Y \models \xi_1 \vee \xi_2$ if, and only if, $Y \models \xi_1$ or $Y \models \xi_2$,
  2. $Y \models \xi_1 \wedge \xi_2$ if, and only if, $Y \models \xi_1$ and $Y \models \xi_2$,
  3. $Y \models \xi$ where ξ is an atomic formulae or its negation, when $\xi \in Y$.

Definition 80. *Let $\mathcal{V}$ be the set $\mathcal{P}(AG) \times \{+,-\}^{n+n'}$. A* counter mode *$\boldsymbol{v}$ is a pair $(X, \boldsymbol{h}) \in \mathcal{V}$.*

A counter mode defines exactly the set of (atomic) formulae of the counter system which are true and the behavior (increasing/decreasing mode) of all terms and counters. Counter modes are involved in the definition of partition of runs. It is worth noting that the number of counter modes is finite and bounded by $2^{\mathcal{O}(N)}$. Given a counter mode, we define the notion of transition compatibility with respect to a counter mode. Intuitively, a transition is compatible with respect to a counter mode $(X, \boldsymbol{h})$ when it can be fired when counter systems behaves according to $(X, \boldsymbol{h})$, i.e., its guards are consequence of the set X and updates are compatible with behavior $\{+,-\}^{n+n'}$ into $\boldsymbol{h}$.

Definition 81. *A transition* $t = q \xrightarrow{(\phi,\boldsymbol{b})} q'$ *is* compatible *with counter mode* $\boldsymbol{v} = (X, \boldsymbol{h})$ *when*

- $X \models \phi$,
- *if* $\boldsymbol{h}(j) = +$ *then* $t_j(\boldsymbol{b}) \geq 0$, *otherwise* $t_j(\boldsymbol{b}) \leq 0$, *for* $j \in [1, n+n']$.

Given a counter mode $\boldsymbol{v}$, a sequence of transition compatible with $\boldsymbol{v}$ defines a phase.

Definition 82. *A* global strict phase *in a finite run is a finite sequence of consecutive transitions compatible with a counter mode.*

Loops of runs play a fundamental role in the analysis provided in the proof. Variables involved into the system of equation providing small solution define exactly the smallest number of times loops are needed to determine the small run.

Definition 83. *A* simple loop *sl with respect to* $\boldsymbol{v}$ *is a sequence* $sl = t_1 \cdots t_\gamma$ *verifying the conditions below.*

1. *The sequence* $t_1 \cdots t_\gamma$ *corresponds to a path in the control graph of* $\mathcal{S}$ *starting and ending by the same control state and the other control states occur only once.*
2. *Each transition of the sequence is compatible with* $\boldsymbol{v}$.

Length of simple loops is bounded by the structure of the graph defining $\mathcal{S}$, i.e., $\gamma \leq \text{card}(Q)$.

We provide now two fundamental measures needed to define the length of small runs. Let nbk be the number of constants occurring in atomic formulae $t \sim k$. Given a r-T-reversal-bounded run, there are $(n+n') \cdot r$ distinct successive phases for the $(n+n')$ terms. During each phase, all terms do not have reversals, i.e., have a fixed mode, according to $\{+,-\}^{n+n'}$, and they can be compared with at most nbk constants. Each term can verify at most $nbk+2$ distinct sets of arithmetical constraints, $t < k_0, k_0 \leq t < k_1, \ldots, k_{nbk-1} \leq t < k_{nbk}, k_{nbk} \leq t$. Therefore, within each phase, we have $(n+n') \cdot (nbk+2)$ different ways of comparing terms with constants of $\mathcal{S}$. So, by considering the $(n+n') \cdot r$ distinct successive phases, a r-T$_\mathcal{S}$-reversal-bounded run admits at most $L \leq ((n+n')^2 \times r \times (nbk+2))$ global strict phases. It is worth noting that L is in $2^{\mathcal{O}(N)}$. Let us consider a simple loop $sl = t_1 \cdots t_\gamma$. Given a counter system $\mathcal{S}$, we show that there are a bounded number of ways of composing loops. Let $scale(\mathcal{S})$ be the maximum update value in $\mathcal{S}$. Then, the *effect* of sl is a vector $\boldsymbol{z} \in [-\text{card}(Q)scale(\mathcal{S}), \text{card}(Q)scale(\mathcal{S})]^n$ such

that $\boldsymbol{z} = \boldsymbol{b}_1 + \cdots + \boldsymbol{b}_\gamma$, where $\boldsymbol{b}_i$ is the update vector of t_i. The *witness* of sl is defined as the first control state in the sequence of transitions and its *loop structure* is a pair $(q, \boldsymbol{z})$ where q is and $\boldsymbol{z}$ are the witness and the effect of sl. Because the effects of simple loops are bounded, the number of distinct effects is bounded by $(2\text{card}(Q)scale(\mathcal{S}) + 1)^n$. Hence, the number α of potential loop structure is bounded by

$$\text{card}(Q) \cdot (2\text{card}(Q)scale(\mathcal{S}) + 1)^n.$$

Now, we are ready to give the sketch of the proof of NEXPTIME-completeness for RB-REACH.

Theorem 84 (Theorem 2, [9])**.** RB-REACH(CS(QFP(<))) *is* NEXPTIME-*complete.*

sketch of proof. Let $\mathcal{S} = (Q, n, \delta)$ be a counter system in CS(QFP(<)), $(q_0, \boldsymbol{x}_0)$ and $(q_f, \boldsymbol{x}_f)$ be configurations, $r \geq 0$ and $\mathtt{T}_\mathcal{S} = \{x_1, \ldots, x_n\} \cup \{t_1, \ldots, t_{n'}\}$.

NEXPTIME-hardness is a direct consequence of [37, Corollary 5(1.)] by considering the set of counter systems $\mathcal{S}$ in CS(QFP($<_1$)) and $\mathtt{T}_\mathcal{S} = \{x_1, \ldots, x_n\}$.

Now, we provide the proof defining the NEXPTIMEupper bound. Given an r-$\mathtt{T}_\mathcal{S}$-reversal-bounded run ρ from $(q_0, \boldsymbol{x}_0)$ to the configuration $(q_f, \boldsymbol{x}_f)$, we construct the shortest r-$\mathtt{T}_\mathcal{S}$-reversal bounded run ρ' from $(q_0, \boldsymbol{x}_0)$ to $(q_f, \boldsymbol{x}_f)$ such that simple loops are fired a number of times that is at most double exponential in N. In fact, it is possible to prove that the short run ρ' is induced by a sequence of transitions π' in δ^* of the form

$$\pi_0 \cdot (q_0 \to q_1) \cdot \pi_1 \cdot (q_1 \to q_2) \cdot \pi_2 \cdot (q_2 \to q_3) \cdots \pi_{\alpha-1} \cdot (q_{\alpha-1} \to q_\alpha) \cdot \pi_\alpha$$

where

- the subpath $\pi = q_0 \to q_1 \to q_2 \to q_3 \cdots \to q_\alpha$ is such that α is at most exponential in N
- each sequence of transitions $\pi_i \in \delta^*$ is a sequence of simple loops with witness q_i, compatible with the same counter mode and which are performed a number of times at most double exponential in N.

The NEXPTIME algorithm guesses the "simple" path π and, for each control state along this path, it guesses the number of times each simple loop is taken. The small solution of the system of equation representing the rearrangement of run ρ bounds each simple loop which is repeated at most a double exponential number of times in N. Then, despite the

fact that ρ' is of length at most double exponential, the sequence of transitions π' can be guessed and tested in exponential time. In fact, because the length of π is at most exponential in N and each simple loop is visited a number of times at most double exponential in N, then, the effect of this repetition on counters can be computed (guessed) in exponential time.

By the first step, we partition run ρ into subruns each corresponding to a global strict phase and we obtain the following partition:

$$(q_{l_0}, \boldsymbol{y}_{l_0}) \cdots (q_{l_1-1}, \boldsymbol{y}_{l_1-1}) \xrightarrow{t_0} (q_{l_1}, \boldsymbol{y}_{l_1}) \cdots (q_{l_2-1}, \boldsymbol{y}_{l_2-1}) \xrightarrow{t_1} \cdots$$

$$\cdots (q_{l_{L-1}}, \boldsymbol{y}_{l_{L-1}}) \cdots (q_{l_L-1}, \boldsymbol{y}_{l_L-1}) \xrightarrow{t_{L-1}} (q_{l_L}, \boldsymbol{y}_{l_L})$$

such that

- for $I \in [0, L-1]$, $(q_{l_I}, \boldsymbol{y}_{l_I}) \cdots (q_{l_{I+1}-1}, \boldsymbol{y}_{l_{I+1}-1})$ is induced by a sequence of transitions compatible with $\boldsymbol{v}_I$,
- for $I \in [0, L-2]$, $\boldsymbol{v}_I \neq \boldsymbol{v}_{I+1}$,
- for $I \in [0, L-1]$, $J \in [l_I, l_{I+1}-1]$, if $\boldsymbol{v}_I = (X, \boldsymbol{h})$, then $\boldsymbol{y}_J \models X$.

The second step of the proof allow us to write the system of equation, given the previous partitioning of run ρ. This step is here simplified and details can be found in [10]. It consists in identifying which simple loops are involved into the run and defining π by preserving only one instance of each simple loop occurring in ρ. Each instance witnesses the position of the simple loop and its type. At the end of this step we have a shorter run which visits exactly the same control states of ρ and which contains the same simple control loops involved within ρ. For each phase $I \in [0, L-1]$ and for all counters $i_C \in [1, n]$ we can write an equation defining exactly the relation between the starting and ending values delimiting the subrun $(q_{l_I}, \boldsymbol{y}_{l_I}) \xrightarrow{*} (q_{l_{I+1}-1}, \boldsymbol{y}_{l_{I+1}-1})$ corresponding to I. The subrun $(q_{l_I}, \boldsymbol{y}_{l_I}) \xrightarrow{*} (q_{l_{I+1}-1}, \boldsymbol{y}_{l_{I+1}-1})$ is induced by the smaller sequence of transitions (where duplicated loops are removed):

$$q_0^I \xrightarrow{t_1^I} q_1^I \rightarrow \ldots \xrightarrow{t_{K_I}^I} q_{K_I}^I$$

such that $q_0^I = q_{l_I}$ and $q_{K_I} = q_{l_I-1}$. It is worth noting that the length K_I of subruns defining a phase can be proved to be bounded by $(1 + |Q|)^2$ and the length of each loop is bounded by $(1 + |Q|)$. Then, the system of equation defining the phase is:

$$((\sum_{j \in [1, \alpha_I] \text{ s.t. } \boldsymbol{s}^I(j) > 0} \boldsymbol{s}^I(j) \boldsymbol{z}_j) + \Sigma_{j=1}^{K_I} t_j^I) = \boldsymbol{y}_{l_{I+1}-1} - \boldsymbol{y}_{l_I}$$

where α_I is the number of simple loops compatible with the I-th phase. In particular, $(\sum_{j\in[1,\alpha_I] \text{ s.t. } \boldsymbol{s}^I(j)>0} \boldsymbol{s}^I(j)\boldsymbol{z}_j)$ is the contribute of loops within the I-th phase whereas $\Sigma_{j=1}^{K_I} t_j^I$ is the contribute of transitions in $q_0^I \xrightarrow{t_1^I} q_1^I \rightarrow \ldots \xrightarrow{t_{K_I}^I} q_{K_I}^I$ which are not involved in loops. Moreover, to complete arithmetical constraints defining the phase we have:

$$\boldsymbol{y}_{l_{I+1}-1} \models X_I \wedge \boldsymbol{y}_{l_I} \models X_I$$

if the counter mode of phase I is $(X_I, \boldsymbol{h}_I)$. Vectors defining the number of iterations of each simple loops $\boldsymbol{s}^0, \ldots, \boldsymbol{s}^{L-1}$ and vectors of counters value between two consecutive phases $\boldsymbol{y}_{l_1-1}, \boldsymbol{y}_{l_1}, \ldots, \boldsymbol{y}_{l_{L-1}-1}, \boldsymbol{y}_{l_{L-1}}$ are solution of the system of equations. Then, by small property of solution for systems of equations, if the system has a solution, then it has a solution such that each value is bounded by $2^{2^{p_1(N)}}$, for some polynomial p_1. Therefore, each simple loop of the original run ρ is performed at most a double exponential number of times. In order to define a formula which measures the length of small runs we consider:

- $((n+n')^2 \times r \times (nbk+2))$ number of strict phases,
- containing at most $(1+|Q|)^2$ control states,
- where each control state iterate loops of length at most $1+|Q|$,
- where each loop can be iterated at most a double exponential number of times.

Then, the length of the small run from is still bounded by a double exponential $2^{2^{p(N)}}$ for some polynomial p.

Finally, after having computed the length of small runs we are able to provide a nondeterministic algorithm which guesses on-the-fly a small r-$\mathtt{T}$-reversal-bounded run from $(q_0, \boldsymbol{x}_0)$ to $(q_f, \boldsymbol{x}_f)$. If ρ' is a small r-$\mathtt{T}_{\mathcal{S}}$-reversal-bounded from $(q_0, \boldsymbol{x}_0)$ to $(q_f, \boldsymbol{x}_f)$ then, the sequence of transitions can be decomposed as

1. a path π (sequence of transitions) of length at most

$$((n+n')^2 \times r \times (nbk+2)) \times [1+(1+\mathrm{card}(Q))^2]$$

2. where each control state on the path is a the witness of a simple loops and each simple loop is visited at most $2^{2^{p_1(N)}}$ times,
3. each term in $\mathtt{T}_{\mathcal{S}}$ performs at most r reversals.

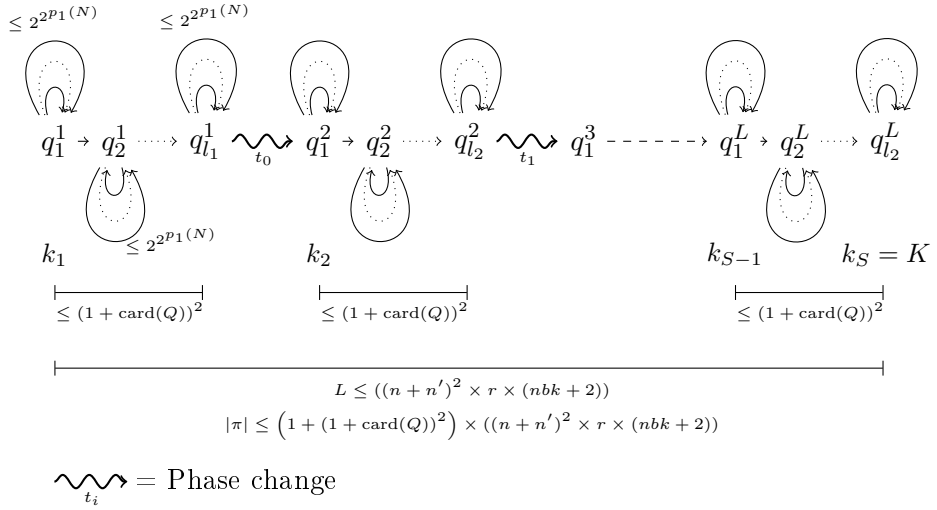


Figure 4.4.: Structure of the path π

The structure of the small run is drawn in the next figure:

So, there are $-1 = k_0 < k_1 < \cdots k_S = K$ with $S \leq ((n + n')^2 \times r \times (nbk + 2))$ such that for $J \in [0, S - 1]$, $(q_{k_J+1}, \boldsymbol{y}_{k_J+1}) \xrightarrow{*} (q_{k_{J+1}}, \boldsymbol{y}_{k_{J+1}})$ is only made of simple loops and each simple loop is applied at most $2^{2^{p_1(N)}}$ times. The number of simple loops is only exponential in N, because it is bounded by $(\mathrm{card}(\delta) + 1)^{\mathrm{card}(Q)}$ $(\leq 2^{N^2})$ from the control graph of $\mathcal{S}$. Hence, a nondeterministic exponential-time algorithm consists in guessing successively on-the-fly each segment starting in $(q_{k_1}, \boldsymbol{y}_{k_1})$, $\ldots$, $(q_{k_S}, \boldsymbol{y}_{k_S})$ by applying simple loops on each control state q_{k_0+1}, $\ldots$, $q_{k_{S-1}+1}$, a number of times bounded by $2^{2^{p_1(N)}}$. The effect of applying a simple loop a number of times bounded by $2^{2^{p_1(N)}}$ can be computed in exponential time in N at once. Therefore, we get the NEXPTIME upper bound because we perform previous guess for an exponential number of control states.

□

By proof of previous Theorem 84 we can derive system of equations defining small runs of systems. In particular, given the two configurations $(q_0, \boldsymbol{x}_0)$ and $(q_f, \boldsymbol{x}_f)$, if there exists a run given $(q_0, \boldsymbol{x}_0) \xrightarrow{*} (q_f, \boldsymbol{x}_f)$ then there exists a small one between the same configurations with no more than r reversals per term, which is compatible with $\sigma = \boldsymbol{v}_0, \cdots, \boldsymbol{v}_{L-1}$ of counter modes, and realized by a sequence of transitions π. The small run is represented by the equality $\mathsf{S}_{\pi,\sigma}$ in which there are variables for

the values

- $\boldsymbol{s}^0, \ldots, \boldsymbol{s}^{L-1}$ representing the number of iteration of simple loops along π
- and $\boldsymbol{y}_{l_0}, \boldsymbol{y}_{l_1-1}, \boldsymbol{y}_{l_1}, \ldots, \boldsymbol{y}_{l_L-1}, \boldsymbol{y}_{l_L}$ representing counters value among phases (starting and ending vector for each phase).

Corollary 85. *Given $\mathcal{S}$ in* CS(QFP), $r \geq 0$ *and control states q, q', one can effectively compute a Presburger formula $\phi_{q,q'}(x_1, \ldots, x_n, y_1, \ldots, y_n)$ such that for all valuations* **val**, $\mathbf{val} \models_{\text{PA}} \phi$ *iff there is an r-$\mathtt{T}_{\mathcal{S}}$-reversal-bounded run from $(q, (\mathbf{val}(x_1), \ldots, \mathbf{val}(x_n)))$ to $(q', (\mathbf{val}(y_1), \ldots, \mathbf{val}(y_n)))$.*

Sketch of proof. Let $\mathsf{S}_{\pi,\sigma}$ be the system of equation representing small runs from configuration $(q, \boldsymbol{x})$ to $(q', \boldsymbol{x}')$ where

- π is the sequence of transitions realizing a path from q to q' of length at most exponential
- and σ be the sequence of counter modes.

We only need to consider variables for simple loops such that all its control states belong to π. In order to provide a formula $\phi_{q,q'}(x_1, \ldots, x_n, y_1, \ldots, y_n)$ we consider the set $\mathtt{X}_\sigma$ of variables $\boldsymbol{y}_{l_1-1}, \boldsymbol{y}_{l_1}, \ldots, \boldsymbol{y}_{l_L-1}$ (we make $\boldsymbol{y}_{l_0}$ and $\boldsymbol{y}_{l_L}$ free vectors of variables). Presburger formula $\phi_{q,q'}(x_1, \ldots, x_n, y_1, \ldots, y_n)$ is defined by considering all possible path π realizing $q \to q'$ and all sequences σ of counter modes. Set $\mathtt{X}_\sigma$ representing intermediate vectors of counters value is existentially quantified.

$$\bigvee_{\pi, \sigma = \boldsymbol{v}_0, \cdots, \boldsymbol{v}_L} \exists\, \mathtt{X}_\sigma\ \mathsf{S}_{\pi,\sigma}$$

□

Next theorems proves that NEXPTIME-completeness can be pushed to RBMC following the (three) reductions previously defined.

Theorem 86. RB-REP-REACH(CS(QFP(<))) *is* NEXPTIME*-complete.*

Sketch of proof. We prove first the NEXPTIMEupper bound. By Theorem 78, given an instance of RB-REP-REACH problem $\mathcal{S}$, $(q_0, \boldsymbol{x}_0)$ and q_f, we can build an $(r+1)$-$\mathcal{S}'$-reversal-bounded counter system $\mathcal{S}' = (Q', n, \delta')$ such that $(q_0, \boldsymbol{x}_0) \xrightarrow{*} (q_{new}, \mathbf{0})$ in $\mathcal{S}'$ if, and only if, there is an infinite r-$\mathtt{T}_{\mathcal{S}}$-reversal-bounded run from $(q_0, \boldsymbol{x}_0)$ such that q_f is repeated infinitely often. In particular, $\mathcal{S}'$ restricts behaviours of $\mathcal{S}$

only by guessing a partition $C = (\mathtt{Z}_{\rightarrow}, \mathtt{T}_{\rightarrow}, \mathtt{T}_{\searrow}, \mathtt{T}_{\nearrow})$ and enforces $(r+1)$-$\mathtt{T}_{\mathcal{S}'}$-reversal-boundedness by construction. Moreover, the dimension of $\mathcal{S}'$ does modify the double exponential length of finite run needed to witness the existence of an infinite r-reversal-bounded run where q_f is repeated infinitely often. Then, small run property of Theorem 84 still holds.

NEXPTIME-hardness is proved by reducing RB-REACH(CS(QFP($<_1$))) to REP-RB-REACH(CS(QFP($<_1$))). Let $\mathcal{S}$ be a counter system in CS(QFP($<_1$)), $(q, \boldsymbol{x})$, $(q', \boldsymbol{x}')$ and $r \geq 0$. We define $\mathcal{S}'$ from $\mathcal{S}$ such that the following two conditions are equivalent:

(I): there is an r-$\mathtt{T}_{\mathcal{S}}$-reversal-bounded run from $(q, \boldsymbol{x})$ to $(q', \boldsymbol{x}')$

(II): there is an $(r+1)$-$\mathcal{S}'$-reversal-bounded infinite run from $(q, \boldsymbol{x})$ such that q^2_{new} is repeated infinitely often.

$$q' \xrightarrow{(\top, -\boldsymbol{x}')} q^1_{new} \xrightarrow{(\boldsymbol{x} = 0, \mathbf{0})} q^2_{new} \circlearrowright^{(\top, \mathbf{0})}$$

□

Similarly to Corollary 85 we can provide an exact characterization of the set of initial configurations allowing control state q_f to be repeated infinitely often.

Corollary 87. *Given $\mathcal{S}$ in* CS(QFP)*, $r \geq 0$ and control states q, q_f, one can effectively compute a Presburger formula $\phi_{q,q_f}(x_1, \ldots, x_n)$ such that for all valuations* $\mathbf{val}$*,* $\mathbf{val} \models_{\mathrm{PA}} \phi$ *if, and only if, there is an infinite r-$\mathtt{T}_{\mathcal{S}}$-reversal-bounded run from $(q, (\mathbf{val}(x_1), \ldots, \mathbf{val}(x_n)))$ such that q_f is repeated infinitely often.*

Sketch of proof. Let us consider a counter system $\mathcal{S}$ in CS(QFP($<$)) (periodicity constraints can be removed). From $\mathcal{S}$ we build $\mathcal{S}' = (Q', n, \delta')$ as in the proof of Theorem 78. Formula $\phi_{q,q_f}(x_1, \ldots, x_n)$ is defined by reversal-bounded reachability ϕ' of $\mathcal{S}'$. In particular, we have to force reachability from the initial configuration $(q, \boldsymbol{x})$ to configuration $(q_{new}, \mathbf{0})$ where $\boldsymbol{x}$ is a vector of free variables.

$$\phi_{q,q_{new}}(x_1, \ldots, x_n, \mathbf{0})$$

□

By previous results we are able to provide the complexity analysis of RBMC problem. This result is proved by estimating the size of the

instanced of problems defined to reduce RBMC to RB-REACH. It is possible to show that the instance of RB-REACH problem, obtained from an instance of a RBMC problem, benefits of the small property of runs which are still bounded by a double exponential with respect to the size of the RBMC instance.

Theorem 88. RBMC *is* NEXPTIME*-complete.*

Sketch of proof. Let $\mathcal{S} = (Q, n, \delta)$ be a counter system in CS(QFP), $(q, \boldsymbol{x})$ be an initial configuration, ϕ be a formula in CLTL(QFP) and $r \geq 0$. First, we prove the NEXPTIMEupper bound. By using previous reductions, we are able to reduce an instance of a RBMC problem to an instance of RB-REACH(CS(QFP($<$))) such that reachability can still be checked in NEXPTIME with respect to the size of the instance of RBMC.

1. Reduce RBMC into RB-REP-REACH(CS(QFP($<, \equiv$))) (see Corollary 74).
2. Eliminate the periodicity constraints (see Corollary 76).
3. Reduce into RB-REACH(CS(QFP($<$))) (see Theorem 78).

All previous reductions are performed in plynomial space. For each step, it is possible to show that values of parameters required to evaluate the final length of the witness run of the RB-REACH problem, preserve double exponential length property of witness runs. This can be done by using the exact expressions evaluating length of runs from the proof of Theorem 84 which can be found in [10].

To evaluate the instance of RBMC we introduce the following parameters:

- N is the size of the instance.
- C is the lcm of all the constants c occuring in arithmetical expressions of the form $t \equiv_c k$ either in $\mathcal{S}$ or in ϕ ($C \leq 2^{N^2}$).
- $nbupdate(\mathcal{S}) \leq N$ is the number of update vectors occurring in $\mathcal{S}$.
- nbk is the number of constants k in atomic formulae of the form $t \sim k$ occurring in $\mathcal{S}$ or ϕ.
- `Max`: maximal value in $\{|a_i|, |k| : \sum_i a_i x_i \sim k$ in $\mathcal{S}$ or in $\phi\}$.
- $n' = \text{card}((\mathtt{T}_{\mathcal{S}} \cup \mathtt{T}_{\phi}) \setminus \{x_1, \ldots, x_n\})$.
- $S = scale(\mathcal{S})$ ($S \leq 2^N$).

The final instance of RB-REACH is defined by a counter system $\mathcal{S}_3 = (Q_3, n, \delta_3) \in \mathrm{CS}(\mathrm{QFP}(<_1))$ $(q_3, \boldsymbol{x}_3)$, and two configurations $(q'_3, \boldsymbol{x}'_3)$. Measures related to the final instance of RB-REACH problem are:

- $\mathrm{card}(Q_3) \leq \mathrm{card}(Q) \times 2^{|\phi|} \times C^n \times 2^n 2^{2n'}$.
- $\mathtt{Max_3} \leq 2N \times 2^{2N}$.
- $S_3 = scale(\mathcal{S}_3) = scale(\mathcal{S}) = S$.
- Size of $(q_3, \boldsymbol{x}_3)$ is bounded by some polynom in N.
- Size of $(q'_3, \boldsymbol{x}'_3)$ is linear in N.
- $n'_3 = n'$.
- $nbk_3 \leq nbk + n' \times nbupdates(\mathcal{S}) \leq 2N^2$.

By Theorem 84, the witness run from $(q_3, \boldsymbol{x}_3)$ to $(q'_3, \boldsymbol{x}'_3)$, considering the above bounds, is of length bounded by $2^{2^{p'(N)}}$ for some polynomial $p'(\cdot)$. In order to guess this run in NEXPTIME, one can design a non-deterministic decision procedure as in the proof of Theorem 84. It is worth noting that the exponential growth, which affects some measures, like $\mathrm{card}(Q_3)$ and nbk_3, does not modify the overall evaluation of length of runs witnessing reachability at final step. In fact, control states in Q_3 can still be encoded in polynomial space, because the cardinal of Q_3 is only exponential in N, and checking whether two configurations perform one step in $\mathcal{S}_3$ can be checked in exponential time when guessing a run of length at most $2^{2^{p'(N)}}$. In particular, since counter values are at most double exponential from Theorem 84, then previous checking can be performed effectively in exponential time.

NEXPTIME-hardness is proved by reducing RB-REACH(CS(QFP($<_1$))) to RBMC. Let $\mathcal{S}$ be a counter system in CS(QFP($<_1$)), $(q, \boldsymbol{x})$ and $(q', \boldsymbol{x}')$ be configurations and $r \geq 0$. As in proof of Theorem 86 we define $\mathcal{S}'$ from $\mathcal{S}$ such that the following two conditions are equivalent:

(I): there is an r-$\mathtt{T}_{\mathcal{S}}$-reversal-bounded run from $(q, \boldsymbol{x})$ to $(q', \boldsymbol{x}')$

(II): there is an $(r+1)$-$\mathcal{S}'$-reversal-bounded infinite run ρ from $(q, \boldsymbol{x})$ such that $\rho, 0 \models \mathbf{F}\ q^2_{new}$.

$$q' \xrightarrow{(\top, -\boldsymbol{x}')} q^1_{new} \xrightarrow{(\boldsymbol{x} = 0, \mathbf{0})} q^2_{new} \overset{(\top, \mathbf{0})}{\circlearrowright}$$

□

As consequence of reduction provided in Theorem 88to show NEXP-TIME-hardness of RBMC we obtain the following corollary since RB-REACH(QFP) and RB-REP-REACH(QFP) can be reduced directly in logarithmic space to RBMC.

Corollary 89. RB-REACH(QFP) *and* RB-REP-REACH(QFP) *problems are* NEXPTIME*-complete.*

The last part of our analysis concerns properties of ultimately periodic runs. In particular, previous Corollary 79 states a connection between the existence of r-$\mathtt{T}$-reversal-bounded runs and ultimately periodic r-$\mathtt{T}$-reversal-bounded runs satisfying the same CLTL property. This fact is fundamental to develop bounded verification for reversal-bounded counter systems because it allows us to restrict the analysis only to ultimately periodic runs and to take advantage of bounded approach presented in Chapter 3. Developing bounded verification approach for r-$\mathtt{T}$-reversal-bounded counter is supported by a stronger property such that r-$\mathtt{T}$-reversal-bounded runs (satisfying a CLTL property) have bounded length. This fact is essential to prove completeness of r-$\mathtt{T}$-reversal-bounded model-checking of counter systems. Informally, given a counter system in CS(QFP) and a CLTL(QFP) property ϕ we define "bounded" r-$\mathtt{T}$-reversal-bounded model-checking as the problem of checking whether there exists an ultimately periodic run ρ of $\mathcal{S}$ satisfying ϕ such that ρ is induced by an ultimately periodic sequence $t_1 \dots t_{l-1}(t_l \dots t_k)^\omega$ of transitions of $\mathcal{S}$ satisfying some suitable properties. An instance of r-$\mathtt{T}$-reversal-bounded model-checking can be solved by checking only a finite amount of "bounded" r-$\mathtt{T}$-reversal-bounded model-checking. Following theorems generalize bounded model-checking over finite counter systems and open the way to bounded approaches in solving model-checking problems over counter systems.

Theorem 90 (Theorem 5, [9])**.** *Let $\mathcal{S}$ be in* CS(QFP)*, ϕ be in* CLTL(QFP) *$r \geq 0$ and q be a control state. One can effectively build a Presburger formula $\phi_q(x_1, \dots, x_n)$ such that for all valuations* $\mathbf{val}$*,* $\mathbf{val} \models_{\text{PA}} \phi_q$ *iff there is an infinite run ρ from $(q, (\mathbf{val}(x_1), \dots, \mathbf{val}(x_n)))$ such that $\rho, 0 \models \phi$ and ρ is r-$\mathtt{T}$-reversal-bounded with $\mathtt{T} = \mathtt{T}_\mathcal{S} \cup \mathtt{T}_\phi$.*

Proof. Let $atoms(\phi)$ be the set of atoms of ϕ and let us assume that ϕ has m until subformulae. We use Lemma 73 which states that we can build a counter system $\mathcal{S}'$ in CS(QFP) such that the two following conditions are equivalent:

1. there is an infinite r-$(\mathtt{T}_\mathcal{S} \cup \mathtt{T}_\phi)$-reversal-bounded run ρ of $\mathcal{S}$ from $(q, \boldsymbol{x})$ such that $\rho, 0 \models \phi$;

2. there is an infinite r-$\mathtt{T}_{\mathcal{S}'}$-reversal-bounded run from $((q, X_0, 0), \boldsymbol{x})$ such that $(q_f, X_f, 0)$ is repeated infinitely often for some *initial* atom $X_0 \in atoms(\phi)$ and for some $(q_f, X_f) \in Q \times atoms(\phi)$.

Let $\phi'_{\alpha,\beta}(x_1, \ldots, x_n)$ be the reversal-bounded repeated reachability $\mathcal{S}'$ which is defined as in proof of Corollary 87. Then, formula $\phi_q(x_1, \ldots, x_n)$ is defined as disjunction of formulae ϕ' where α is the initial state $(q, X_0, 0)$, for some initial atom $X_0 \in atoms(\phi)$, and β is a final state involving q_f:

$$\bigvee_{X_0,\ (q_f,X_f)\in Q\times atoms(\phi)} \phi'_{(q,X_0,0),(q_f,X_f,0)}(x_1, \ldots, x_n)$$

□

We are also able to improve Corollary 91 since we also have bounds on the length of reversal-bounded runs (see the proof of Theorem 88).

Corollary 91 (Corollary 7, [9]). *Let $\mathcal{S}$ be in* CS(QFP), *$(q, \boldsymbol{x})$ be an initial configuration, ϕ be in* CLTL(QFP) *and $r \in \mathbb{N}$. The propositions below are equivalent:*

(I) *there is an infinite run ρ from $(q, \boldsymbol{x})$ such that $\rho, 0 \models \phi$ and ρ is r-$\mathtt{T}$-reversal-bounded with $\mathtt{T} = \mathtt{T}_{\mathcal{S}} \cup \mathtt{T}_{\phi}$.*

(II) *There exists an* ultimately periodic *run ρ satisfying the same properties as in (I), and the corresponding sequence of transitions $\pi_1(\pi_2)^\omega$ verifies that the length of $\pi_1\pi_2$ is bounded by $2^{2^{p_0(N)}}$, for some polynomial $p_0(\cdot)$ and N is the size of the instance of RBMC.*

As previously anticipated, previous two results are essential for practical verification of r-$\mathtt{T}$-reversal-bounded counter systems. By Theorem 90, given the formula $\phi_q(x_1, \ldots, x_n)$, we can check if an initial configuration verifies the existence of an infinite run satisfying a temporal formula by means of tool handling Presburger arithmetic. Hence, Theorem 90 states that verification problems over r-$\mathtt{T}$-reversal-bounded counter systems are reduced effectively to satisfiability in Presburger arithmetic. Moreover, results on the computational complexity guarantee that the studied method is optimal. Bounded verification approaches naturally come from Corollary 91 and can effectively take advantage of k-bounded satisfiability of CLTLB(QFP) formulae as we developed in Chapter 3. Since an instance of RBMC can be transformed into an instance of RB-REACH(QFP) and by Theorem 84, one could solve the reversal-bounded model checking problem by looking for finite runs of length at most doubly exponential.

5. Case Studies

This chapter provides two (implemented) examples of application where bounded approach, that we have presented in Chapter 3, can be used in verification and synthesis. The first one is presented in Section 5.1; it is a pure descriptive specification of the behavior of an hysteresis variable. Though its nature makes the example a proof-of-concept, we believe that it is meaningful in its own since it provides a clear instance of verification and synthesis of parameters. The second case study is a real application of services substitution in the context of Service Oriented Applications. Informally, services are providers either of information or of concrete operations which can be used or invoked by actors of the environment. Whenever a service changes some of its peculiarity or failures break the regular behavior of the system, finding the best candidate replacing the service and interconnecting new instances of service providers with the rest of users become two essential challenges. In Section 5.2 we study the problem and we provide a method to replace a service A with a service B by means of adaptors which are synthesized over models of CLTLB specification formulae representing their behavior.

5.1. Case Study I: hysteresis phenomena

In this section, we show an example requiring the full expressiveness of CLTLB(DL) over a domain $D = \{\mathbb{Z}, \mathbb{R}\}$. It represents an hysteresis time varying variable y ranging over two possible values $\{0, 1\}$ which is led by an independent variable $x \in D$. A configuration of the system is a pair (x, y) of values in D^2. The hysteresis of variable y is defined with respect to two parameters x_0 and x_1. When $x_0 \leq x \leq x_1$ variable y is low or high and its value depends on previous values of x. Otherwise, if $x < x_L$ then variable y has high value equal to 1; when $x > x_H$ then variable y has low value equal to 0.

Behavior of y is defined by a set of CLTLB(DL) formulae $H(x_L, x_H)$ and it is not represented by a transition system. Satisfiability of $H(x_L, x_H)$ over k-bounded models is used to solve reachability and synthesis problem. When thresholds are defined we verify if the system can reach a "bad" configuration in k instant of time. "Bad" configurations are pairs

of

$$B = \{(x, y) \in \mathbb{N}^2 \mid (x > x_H, y = 1) \text{ or } (x < x_L, y = 0)\}.$$

Let $H(\mathsf{x_L}, \mathsf{x_H})$ be the formula defining the hysteresis and $(\mathsf{x_L}, \mathsf{x_H})$ be two value in D^2, and (x_0, y_0) be the initial configuration for the system. Reachability problem amounts to verify if there exists a behavior of (x, y) such that (x_k, y_k) is reached from (x_0, y_0), i.e. $(x_0, y_0) \rightarrow_k (x_k, y_k)$.

When thresholds are undefined, one parameter x_L, x_h, or both, are free variables of formula $H(x_L, x_H)$. Synthesis problem for parameters x_L and x_H amounts to identify suitable values for x_L and x_H such that behavior of the hysteresis satisfies a property ϕ. Synthesis problem can be reduced to k-bounded satisfiability of the CLTLB(DL) formula $H(x_L, x_H)$. Formula $H(x_L, x_H)$, where (x_L, x_H) are free variables, is k-bounded satisfiable if there exists a model $\widehat{\sigma}_k$ and an environment ε, defining a value for x_L and x_H, such that $\widehat{\sigma}_k \models_k H(x_L, x_H)$.

We define predicates which represent the behavior of variables x and y, in order to have a coincise description of formulae $H(x_L, x_H)$.

- $\mathsf{updown}(z) \stackrel{\text{def}}{\Leftrightarrow} (z = 0) \wedge (\mathrm{Y}z = 1)$ represents a change from high-level value to low-level value.
- $\mathsf{downup}(z) \stackrel{\text{def}}{\Leftrightarrow} (z = 1) \wedge (\mathrm{Y}z = 0)$ symmetrically represents a change from low-level value to high-level value.
- $\mathsf{low}(z) \stackrel{\text{def}}{\Leftrightarrow} (z = 0)$ and $\mathsf{high}(z) \stackrel{\text{def}}{\Leftrightarrow} z = 1$ represent low and high level, respectively.

Behavior of variables is defined by the following formulae.

tr-up-down: $\mathbf{G}(\mathbf{Y}((x < x_H) \wedge \mathsf{high}(y)) \wedge (x \geq x_H) \Rightarrow \mathsf{updown}(y))$ defines sufficient condition to have a movement from high-level to low-level.

tr-down-up: $\mathbf{G}(\mathbf{Y}((x > x_L) \wedge \mathsf{low}(y)) \wedge (x \leq x_L) \Rightarrow \mathsf{downup}(y))$ defines sufficient condition to have a transition from low-level to high-level.

s-up-down: $\mathbf{G}(\mathsf{updown}(y) \Rightarrow \mathbf{Y}(x < x_H) \wedge (x \geq x_H))$ enforces necessary condition on x to have movement from high-level to low-level.

s-down-up: $\mathbf{G}(\mathsf{downup}(y) \Rightarrow \mathbf{Y}(x > x_H) \wedge (x \leq x_H))$ enforces necessary condition on x to have movement from low-level to high-level.

low-state: $\mathbf{G}(\mathsf{low}(y) \Rightarrow (\mathsf{low}(y)\ \mathbf{U}\ \mathsf{downup}(y))\ \mathbf{S}\ \mathsf{updown}(y))$ defines necessary condition for verifying low state of y.

high-state: $\mathbf{G}(\mathsf{high}(y) \Rightarrow (\mathsf{high}(y)\ \mathbf{U}\ \mathsf{updown}(y))\ \mathbf{S}\ \mathsf{downup}(y))$ defines necessary condition for verifying high state of y.

The invariant property defining values for high and low-level of y is $\mathbf{G}(\mathsf{low}(y) \vee \vee \mathsf{high}(y))$. Threshold values are such that $x_H > x_L$; the formula does not require temporal modalities since x_H and x_L are parameters which do not vary over time. Both variables x_H and x_L are not quantified. When they are considered as parameters, we use an initialization formula which defines their values; i.e. $x_L = \mathsf{x_L}$ and $x_H = \mathsf{x_H}$ where $\mathsf{x_H}, \mathsf{x_L} \in D$. Otherwise, if x_H and x_L are free variables they will be given a value by means of the environment ε.

The behavior of the indipendent variables x can be constrained to be "discretely continuous" or not by imposing one of the two formulae:

continuosX: $\mathbf{G}((\mathrm{X}x = x + 1) \vee (\mathrm{X}x = x - 1))$

NcontinuosX: $\mathbf{G}(\exists d\ (\mathrm{X}x = x + d) \vee (\mathrm{X}x = x - d))$

where $d \in D$ is a quantified variable.

Formula $H(x_L, x_H)$ defining the hysteresis phenomenon between variables x and y is defined by the conjunction of all previous formulae.

Given two values $(\mathsf{x_L}, \mathsf{x_H}) \in D^2$, verifying that hysteresis behavior is "safe" amounts to check property like $\mathbf{G}((\mathsf{high}(y) \Rightarrow x \leq x_H) \wedge (\mathsf{low}(y) \Rightarrow (x \geq x_L)))$. This reduces to a reachability problem for $H(\mathsf{x_L}, \mathsf{x_H})$ with respect to "bad" configuration belonging to the set B. Formula $\phi_{bad} = \mathbf{F}((x > x_H \wedge \mathsf{high}(y)) \vee ((x < x_L \wedge \mathsf{low}(y))))$ is conjuncted to $H(\mathsf{x_L}, \mathsf{x_H})$ and, then, the resulting formula is checked for satisfiability. When an initial configuration is not specified, unsatisfiability means that bad configuration can not be reached by any initial configuration, over k-bounded models. It is worth noticing that k-bounded satisfiability defines only a "partial" representation of the arithmetic behavior by means of the model $\widehat{\sigma}_k$. Nonetheless, atomic formulae are represented over infinite models according to the bounded semantics given in Section 3.2.

Synthesis problem for the hysteresis $H(x_L, x_H)$ amounts to define two values $(\mathsf{x_L}, \mathsf{x_H}) \in D^2$ such that $H(\mathsf{x_L}, \mathsf{x_H})$ is feasible. Being x_L, x_H two free variables, the environment ε defines their values $x_L = \mathsf{x_L}$ and $x_H = \mathsf{x_H}$ if $(\pi, \widehat{\sigma}_k) \models_k H(x_L, x_H)$. A deeper analysis can be realized by verifying if "bad" systems can be synthesized. This can be done by checking if $H(x_L, x_H) \wedge \phi_{bad}$ is satisfiable, i.e., there exists a pair of values for parameters $(\mathsf{x_L}, \mathsf{x_H}) \in D^2$ such that a bad configuration can be reached from an initial one.

In Appendix A.2 we give the Lisp code defining the hysteresis which can be verified by the tool presented in Section 7.

5.2. Case Study I: Verification of Service Substitutability

Service Oriented Architectures (SOAs) are a flexible set of design principles that promote interoperability among loosely coupled services that can be used across multiple business domains. In this context applications are typically composed of services made available by third-party vendors. Thus, an organization does not have total control of every part of the application, hence failures and service unavailability should be taken into account at runtime. On the other hand, during the application execution new services might become available, that enable new features or provide equivalent functionalities with better quality. Therefore the ability to support the evolution of service compositions, for example by allowing applications to substitute existing services with others discovered at runtime, becomes crucial.

Most of the frameworks proposed in recent years for the runtime management of service compositions make the assumption that all semantically equivalent services agree on their interface, as proposed by Antonellis et al. in [88] or by Verma in [89]. In the practice this assumption turns out to be unfounded. The picture is further complicated when one considers *conversational services*, i.e., services that expose operations with input/output data dependencies among them. In fact, in this case the composition must deal with *sequences* of operation invocations, i.e., the *behavior protocol*, instead of single, independent, ones.

The *substitutability* problem is the problem of deciding when a service can be dynamically substituted by another one discovered at runtime. Cavallaro et al. in [90] [91] propose an approach based on Bounded Model Checking (BMC) techniques. Even if the approach proves to be quite effective, the Propositional Satisfiability (SAT) problem, on which the standard encoding of BMC relies, requires to deal with lengthy constraints, which typically limits the efficiency of the analysis phase. In the setting of the runtime management of service compositions this is not acceptable, as delays incurred when deciding whether services are substitutable or not can hamper the operativeness of the application.

We introduce a verification technique, based on Satisfiability Modulo Theories (SMT) and we use CLTLB(DL) and its associated verification technique to model and efficiently analyze service-based applications. SMT-based verification technique has two main advantages:

- unlike in the standard SAT-based approach, arithmetic domains are not approximated by means of a finite representation, which proves to be particularly useful in the service substitutability prob-

lem;

- the implemented prototype is shown to be considerably faster and with smaller memory footprint than existing ones based on the propositional encoding, due to the conciseness of our solution.

The technique exploits decidable arithmetic theories supported by many SMT solvers to natively deal with integer variables (hence, with an infinite domain). For instance, we can refer to Microsoft `Z3` [92] or SRI `yices` [93]. This allows us to decide larger substitutability problems than before, in significantly less time: the response times of our prototype tool make it usable also in a runtime checking setting.

5.2.1. Substitutability Checking of Conversational Services

In an open world setting, as for service oriented systems, application components are usually owned by third parties and may unexpectedly fail and need to be substituted with other previously unforeseen. When this occurs at runtime, the composition (or the framework where the composition is running) should be able to perform the replacement requiring as little human intervention as possible.

The approach presented by Cavallaro et al. in [90] enables service substitution through the automatic definition of suitable *mapping scripts*. These map the sequences of operations that the client is assuming to invoke on the *expected service* into the corresponding sequences made available by the *actual service* (i.e., the service that will be actually used). Mapping scripts are automatically derived given:

- a description of service interfaces in which input and output parameters are associated with each service operation,
- the behavioral protocol associated with each service, described through an automaton.

The mapping between an expected and an actual service assumes that the *compatibility between data* has been previously defined. This relation allows us to map data of different services to the same label. For the sake of simplicity, here we assume that data are compatible if they are called the same way (more sophisticated compatibility relationships are explored by Cavallaro et al. in [94]).

Given this compatibility definition, we say that a sequence of operations seq_{exp} in the automaton of the expected service is *substitutable* by another sequence of operations seq_{act} in the automaton of the actual service if a client designed to use the expected service sequence can use

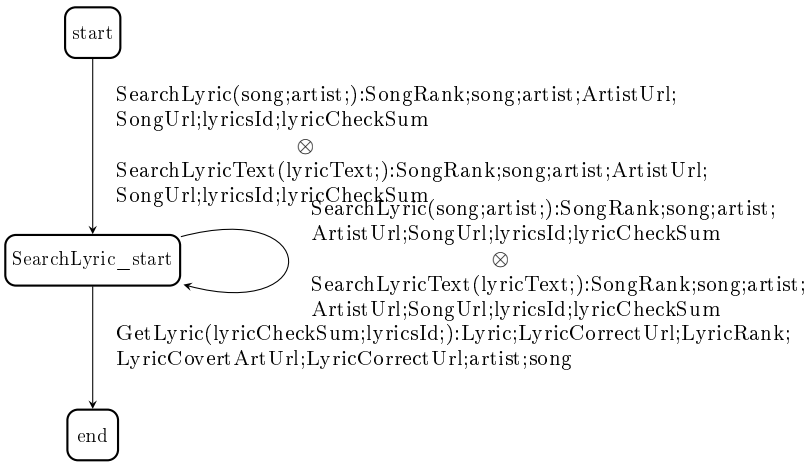


Figure 5.1.: LTS of the ChartLyrics service of Section 5.2.2 (⊗ denotes that the operations are on different transitions).

the actual service sequence without noticing the difference. This happens when the actual operations require as input at most all of the data provided as input to the expected operations and return at least all the data the expected sequence provides as output.

The rationale of this definition can be understood by considering the service substitution process. In this process a client is designed to interact with an expected service and, therefore, it assumes to invoke the expected service operations, providing for each invocation the data required by the operation and awaiting as output the data provided by the operation. When the expected service is substituted by an actual service, in order for the client to be unaware of the change, the actual service should be able to work with the data provided by the client as if to the expected service and should return the data the client is awaiting from the expected service.

The formal model of substitutability allows us to build a reasoning mechanism based on temporal logic that, given an expected service sequence, returns a corresponding actual service sequence. It includes the behavioral protocols of both the expected and the actual services represented as Labeled Transition Systems (LTS) and formalized in temporal logic, in which each transition is labeled with the associated operation. Input and output parameters of each operation are also part of the model (Figures 5.1 and 5.2 show the LTSs of two services discussed in Section 5.2.2).

In order to model the substitution process and to keep track of the

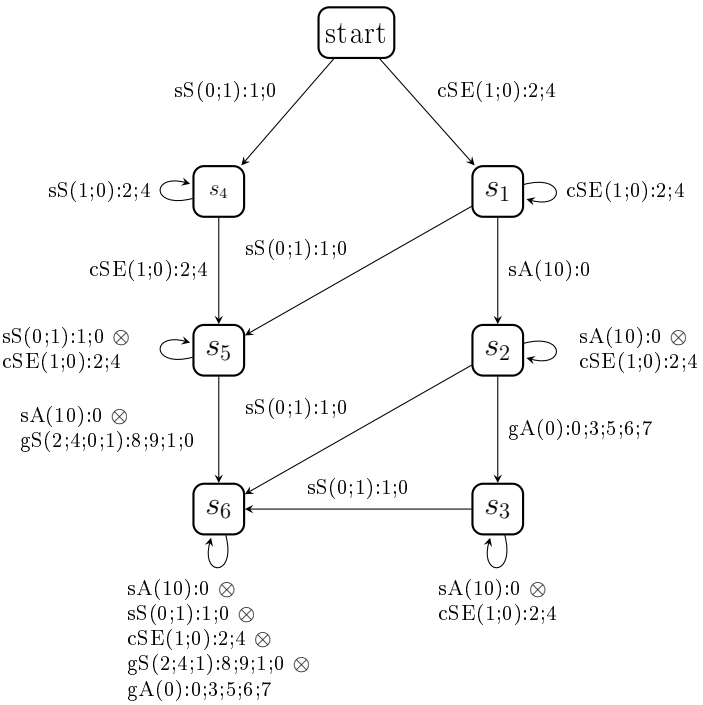


Figure 5.2.: LTS of the *LyricWiki* service discussed in Section 5.2.2.

data exchanged we maintain two kinds of counters:

- *seen*, which is used to check that the actual service can work using a subset of the input data provided by the client to the expected service,
- *needed*, which is used to check that the actual service can provide a superset of the data the client expects to receive as output of the expected service.

The model includes an instance of *seen* (resp. *needed*) for each type of data that can be used as input (resp. output) parameter for an operation. Each time an operation of the expected service is invoked, the instances of *seen* for its input parameters and those of *needed* for its output parameter are all incremented by one.

To illustrate this mechanism, consider the services depicted in Figures 5.1 and 5.2[1], which represent, respectively, the expected and the actual service (the two services are presented in more detail in Section 5.2.2; in

[1] **Operations**: searchSongs (sS), checkSongExists (cSE), searchArtists (sA), getArtist (gA), getSong (gS). **Parameters**: artist (0), song (1), lyricsId (2), item

this Section we focus only on the aspects that are relevant to the example at hand). Operation *SearchLyric* of the expected service of Fig. 5.1 has two input parameters, *song* and *artist*, and five output parameters, *SongRank*, *song*, *artist*, *ArtistUrl*, *SongUrl*, *lyricsId* and *lyricCheckSum*. After its invocation, *seen(song)* and *seen(artist)* are incremented by 1. The same increment takes place for *needed(SongRank)*, *needed(song)*, *needed(artist)*, *needed(ArtistUrl)*, *needed(SongUrl)*, *needed(lyricsId)* and *needed(lyricCheckSum)*. Conversely, when an operation of the actual service is invoked, the instances of the *seen* counter for each input parameter and those of the *needed* counter for each output parameter are all decremented by one. For example, when operation *checkSongExists* of the actual service of Fig. 5.2 is invoked, *seen(song)*, *seen(artist)*, *needed(lyricsId)* and *needed(lyricCheckSum)* are decremented by 1 (and consequently run to 0).

To conform to the notion of substitutability of expected and actual sequences of operations presented above, an actual service operation can be invoked only if the *seen* counter for each of its input parameters is ≥ 0 (i.e. the input parameters have been provided by a client assuming to invoke some operations on the expected service). When the value of a *needed* counter is 0 it means that the actual service provided enough instances of a certain type of data to fulfill client requests. If, on the other hand, the actual service provides more instances of a type of data than those requested, then the corresponding *needed* counter is < 0.

In case the expected service operation sequence analyzed is substitutable by one in the actual service, a mapping script is generated and then interpreted by an *adapter* that intercepts all service requests issued by the client and transforms them into some requests the actual service can understand. Fig. 5.3 shows the placement of adapters into the infrastructure architecture and highlights their nature of intermediaries (see Cavallaro et al. in[90] for more details).

5.2.2. Case Study

To demonstrate our methodology, we use an example concerning two existing conversational services available on the Internet. These two services realize two lyric search engines. One is called *ChartLyrics*[2], the other *LyricWiki*[3].

(3), lyricCheckSum (4), SongUrl (5), year (6), album (7), LyricCorrectUrl (8), Lyrics (9), lyricText (10) ($\otimes$ denotes that the operations are on different transitions).

[2] http://www.chartlyrics.com/api.aspx

[3] http://lyrics.wikia.com/Main_Page

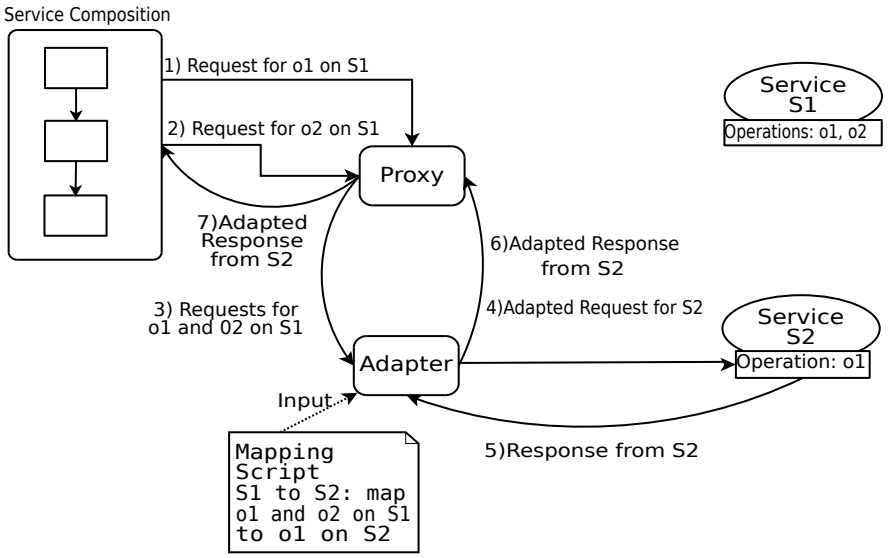


Figure 5.3.: The adaptation runtime infrastructure.

ChartLyrics is a lyrics database sorted by artists or songs; the WSDL[4] of *ChartLyrics* provides three operations:

- *SearchLyric* to search available lyrics,
- *SearchLyricText* to search a song by means of some text within an available lyric text,
- *GetLyric* to retrieve the searched lyric.

LyricWiki is a free site where anyone can go to get reliable lyrics for any song from any artist. The WSDL of *LyricWiki*[5] provides several operations. Five of them are of interest for our purposes:

- *searchSongs* to search for a possible song on *LyricWiki* and get up to ten close matches,
- *checkSongExists* to check if a song exists in the *LyricWiki* database,
- *getSong* to get the lyrics for a searched *LyricWiki* song with the exact artist and song match,
- *searchArtists* to search for a possible artist by name and return up to ten close matches,

[4] http://api.chartlyrics.com/apiv1.asmx?WSDL
[5] http://lyrics.wikia.com/server.php?wsdl

- *getArtist* to get the entire discography for a searched artist.

To get a lyric through *ChartLyrics*, a client can exploit the following sequence of operation invocations: *SearchLyric*, *GetLyric*. Conversely, to get a lyric through *LyricWiki*, a possible sequence of operation invocations is the following: *checkSongExists*, *searchSongs*, *getSong* (see the representation of the conversational protocols of *ChartLyrics* and *LyricWiki*, respectively, in Fig. 5.1 and Fig. 5.2).

If *LyricWiki* were part of a web application realized through a service composition, it could happen that, in certain circumstances, it would need to be replaced by *ChartLyrics* or by any other specialized search engine. This could happen, for instance, to accommodate the preferences of users having their preferred engine, or to handle the cases when *LyricWiki* is unavailable for any reason. The developer could code, by hand, the instructions to deal with any possible engine and its replacement. However, this approach does not allow the application to deal with search engines unknown at design time. A better solution, which would overcome this problem, is to build a mapping mechanism that dynamically handles the mismatches by automatically synthesizing a behavior protocol mapping script. The adaptation realized by the synthesized mapping script could state, e.g., that the sequence of *LyricWiki* operations *checkSongExists*, *searchSongs*, *getSong* maps to the sequence of *ChartLyrics* operations *SearchLyric*, *GetLyric*.

Let us consider as an example the expected service operation sequence *checkSongExists*, *searchSongs*, *getSong*, which brings the *LyricsWiki* behavior protocol automaton from state *start* to state s_6 (see Fig. 5.2). We assume to have established a compatibility relation between services' data. Also, for the sake of brevity, the automata of Figures 5.1 and 5.2 are represented with this relation already established, though in practice this requires an additional mapping step (for more details see [94, 91]).

The automata describing service protocols and the expected service operation sequence are all formalized through suitable CLTLB(DL) formulae *expectedService*, *actualService* and *expectedOperationSequence*, simply describing the transition relation between states. Then, we formulate the problem of checking if the expected service can be substituted by the actual service in terms of a reachability problem over the automata describing the protocols of the expected and actual services. The problem consists of searching for a finite operation sequence on the actual service automaton which is substitutable to the expected operation sequence given as input. We check this condition by verifying that the actual service operation sequence should require no more input parameters than those provided to the expected service sequence, and it should

provide at least the same parameters provided by the expected service sequence. To ensure this property we keep track, through instances of counters *seen* and *needed* (see Section 5.2.1), of how many parameters of any given kind are provided as input to the expected service operations and of how many parameters of any given kind are returned by each actual service operation. In particular, we define:

- $Waiting : N \rightarrow \{true, false\}$ and $Response : \mathbb{N} \rightarrow \{true, false\}$ to be two uninterpreted predicates
- $seen : \mathbb{N} \rightarrow \mathbb{N}$ and $seen : \mathbb{N} \rightarrow \mathbb{N}$ to be two uninterpreted functions.

The behavior of counter *needed* is formalized by the CLTLB(DL) formula

$$\begin{aligned}
&\forall x\big(RetData(x) \Rightarrow \\
&\quad (Waiting(x) \wedge seen(x) \geq 0 \Rightarrow \top \mathbf{U} Response(x)) \wedge \\
&\quad (\neg Waiting(x) \wedge \neg Response(x) \Rightarrow needed(x) = \mathrm{X} needed(x)) \wedge \\
&\quad (\neg Waiting(x) \wedge Response(x) \Rightarrow \\
&\qquad\qquad needed(x) = \mathrm{X} needed(x) + 1) \wedge \\
&\quad (Waiting(x) \Leftrightarrow needed(x) = \mathrm{X} needed(x) - 1)\big)
\end{aligned}$$

where $RetData(x)$ holds if x is the type of an output parameter of a service (either expected or actual); $Waiting(x)$ (resp. $Response(x)$) holds when the expected (resp. actual) service is invoked and x is the type of one of the output parameters of the invoked service. For instance, when operation *checkSongExists* of *LyricWiki* is invoked, $wait(lyricsId)$ holds and when operation *SearchLyric* of *ChartLyrics* is invoked $resp(SongRank)$ holds. To completely state the reachability problem, an initial and a final condition over counters and states of automata are defined. The initial condition sets all counters to 0 and the states of the two LTSs to *start*. The final condition imposes $\forall x(RetData(x) \Rightarrow needed(x) \leq 0)$.

Finally, a solution for the reachability problem can be obtained by checking the satisfiability of the conjunction of the CLTLB(DL) formulae mentioned above.

Considering the example sequence on *LyricsWiki*, a client expecting to invoke this sequence is assuming to provide as input to the first operation of the sequence a song and an artist. This will set the *seen* counter to 1 for both provided inputs. Moreover, it expects the invoked operation to return a *lyricsId* and a *lyricCheckSum*, which will increment the corresponding instances of the *needed* counter to 1. Considering the actual service protocol, our approach searches for an operation accepting a subset of the provided input data and providing a superset of the required return data.

The operation to be selected should leave the *start* state as the state compatibility relation provided as input for the approach mandates the compatibility of state *start* of *LyricsWiki* with state *start* of *ChartLyrics*. In our example the invocation of *checkSongExists* makes *SearchLyric* the only suitable candidate. After the invocation of this actual service operation all instances of *seen* and those instances of *needed* associated to the output parameters of *checkSongExists* are reset to 0. The actual service operation returns also some extra data that are not required by the invoked expected service operation (i.e. song, artist, songRank, artistUrl, songUrl). In this case the reasoning mechanism offers two possible choices: extra data can be discarded (hence ignored also in the future), or they can be initially ignored, but stored for an eventual later use. The former strategy is more conservative, but it may also limit the possibility of the reasoning mechanism to find an adapter. The latter strategy may affect data consistency in some cases, as it allows using as a reply for an operation some data that have been received before the request has been actually issued, but it also opens the possibility of finding adapters in situations in which the former would fail. In this case study we use the latter strategy, hence the *needed* counters for those data that are not required as a response by the invoked expected service operation are set to -1.

After the invocation of *SearchLyric* the actual service goes in *SearchLyric_ start* state. The next operation on the expected sequence to be invoked is *searchSongs*, which requires as input the names of the song to be searched and of its author and provides as return parameters the names of the artist and of the song, if they are found. Since the *needed* counters for both the name of the artist and of the song are set to -1, instances of those data have been previously stored, hence no operation shall be invoked on the actual service, which remains in state *SearchLyric_ start*.

The last operation in the expected sequence is *getSong*, which requires as input artist and song names and the id and checkSum returned by the previously invoked *checkSongExists*. The expected service has again the same three operations of the previous step available, but this time there are two available candidates for selection: *searchSongs* and *GetLyric*. Given the data-flow constraints elicited before, *GetLyric* is the only available operation that can satisfy also the state compatibility relation. After the invocation of *GetLyric* the expected and actual services are in compatible states and the *needed* counter instances are all set to 0. Then, the actual service operation sequence found can be substituted to the expected service sequence.

A mapping script generated for the example sequence in this section is reported in Table 5.1. Each step contains the state in which each one of the analyzed automata is, the operations in seq_{exp} and in seq_{act} that should be invoked in that step, and the exchanged data, if any. For each operation in seq_{exp} the adapter expects to receive an invocation for the expected service, and for each operation in seq_{act} the adapter performs an invocation to the actual service. The table shows also the updates for the *seen* and *needed* counters.

5.2.3. Evaluation and Experimental Results

In order to evaluate the encoding presented in this Section we use the plug-in ae^{2}Zot which is presented in Section 7 and we used it in three sets of experiments.

- We created adapters for sequences of increasing length related to the case study presented in Section 5.2.2. This set of experiments was used as a qualitative evaluation of the approach on examples taken from the real world.

- We ran the same set of experiments on Zot using three different encodings, namely: the SAT-based encoding of PLTL (PLTL/SAT, see [95] for details); the SMT-based one of the same logic (PLTL/SMT, which is the one described in Section 3.3.1 except that counters are encoded only for finite domains) and the SMT-based one of CLTLB(DL), featuring the encoding of the a.t.t.'s presented in Section 3.3.5. We measured elapsed time and occupied memory, and we compared the results to get an estimate of how the introduction of the SMT solver speeds up the adapter-building mechanism.

- We created some service interface models with growing number of parameters and tried to solve them with both the original version of the encoding and with the extensions. This set of experiments has the purpose to compare how much the new encoding scales on models larger than those found in common practice.

All experiments were run using the Common Lisp compiler SBCL 1.0.29.11 on a 2.50GHz Core2 Duo laptop with Linux and 4 GB RAM. The SMT solvers used in our tests are: Microsoft Z3 2.4 (http://research.microsoft.com/en-us/um/redmond/projects/z3/) and SRI Yices 2.0 prototype (http://yices.csl.sri.com/). For the SAT-based PLTL encoding we used MiniSat 2.0 beta (http://minisat.se/).

The first set of experiments was carried out selecting some operation sequences on the expected service presented in Section 5.2.2. The selected sequences set comprises the simple sequence analyzed in the case study plus sequences of growing length obtained trying to execute up to 5 consecutive *searchSongs* and *checkSongExists* operations. We set the time bounds for the experiments using a simple heuristic, based on the sum of the states of the automata of the input services. In those cases in which the abstract sequence featured repeated invocations of the same operation, the time bound was augmented with the number of repetitions of each operation. This set of experiments produced a set of mapping scripts that we checked by inspection. Fig. A.1(a) and Fig. A.1(b) report the overall results. Fig. A.1(b) shows that the CLTLB(DL) encoding has lower memory occupation than the SAT-based PLTL encoding for the same problem. Fig. A.1(a) shows that the CLTLB(DL) encoding on Z3 performs much better than the others.

Lastly, we tried to push the limits of our technique to check its robustness. To do so, we generated simple service protocols featuring operations with a growing number of parameters. We chose this setting for our experiments based on our experience in the common practice, which suggests that services usually exhibit very simple protocols, while operations have sometimes a considerable number of parameters. Note that the models used in these experiments are much bigger than those commonly found in practice. The experiments are based on expected and actual services with 10 states, and a trace bound of 21 time instants. The results are shown in appendix A.1 Figure A.1(c) and in Figure A.1(d). The number of parameters used in experiments ranges from 10 (i.e. each operation has 10 input and 10 output parameters) to 90. As shown in the figures, the CLTLB(DL) encoding on Z3 was the only one we managed to push up to 90 parameters, while we stopped experimenting much earlier with the PLTL encoding on Yices, Z3 and MiniSat. It is worth noticing that in Figures A.1(c)-A.1(d) the combination CLTLB(DL)/Yices is missing because of its poor performance on this set of experiments (the simplest case was solved in more than 500 seconds).

5.2.4. Related Work

Our approach is closely related both to works supporting substitution of services and to works about verification using model checking. Many approaches that support the automatic generation of adapters (or equivalent mechanisms) are based on the use of ontologies and focus on non-conversational services (see for instance [94, 96]). They all assume that the usual WSDL definition of a service interface is enriched with some

kinds of ontological annotations. At run-time, when a service bound to a composition needs to be substituted, a software agent generates a mapping by parsing such ontological annotations. *SCIROCO* [97] offers similar features but focuses on stateful services. It requires all services to be annotated with both a SAWSDL description and a WS-ResourceProperties [98] document, which represents the state of the service. When an invoked service becomes unavailable, *SCIROCO* exploits the SAWSDL annotations to find a set of candidates that expose a semantically matching interface. Then, the WS-ResourceProperties document associated with each candidate service is analyzed to find out if it is possible to bring the candidate in a state that is compatible with the state of the unavailable service. If this is possible, then this service is selected for replacement of the one that is unavailable. All these three approaches offer full run-time automation for service substitution, but as the services they consider are not conversational, they perform the mapping on a per-operation basis. An approach that generates adapters covering the case of interaction protocols mismatches is presented in [99]. It assumes to start from a service composition and a service behavioral description both written in the BPEL language [100]. These are then translated in the *YAWL* formal language [101] and matched in order to identify an invocation trace in the service behavioral description that matches the one expected by the service composition. The matching algorithm is based on graph exploration and considers both control flow and data flow requirements. The approach presented in [102] offers similar features and has been implemented in the open source tool Dinapter (http://sourceforge.net/projects/dinapter). While both these approaches appear to fulfill our need for supporting interaction protocol mapping, they present some shortcoming in terms of performances, as shown in [90].

Step	*Execution trace Content*	Counters value
1	*LyricWikz*State:start ; *LyricWikz*Operation:checkSongExists *LyricWikz*Input: song, artist; *LyricWikz*Output:lyricId, lyricCheckSum *chartLyricsS*tate:start; *LyricWikz*Operation:checkSongExists	All counters set to 0
2	*LyricWikz*State:s_1 *chartLyricsI*nput: song, artist *chartLyricsO*utput:song , artist, artistUrl, songRank, lyricsId, lyricChecksum *chartLyricsS*tate:start; *chartLyricsO*peration:searchLyric	seen(song) = seen(artist) = 1 needed(lyricId) = needed(lyricCheckSum) = 1
3	*LyricWikz*State:s_1; *LyricWikz*Operation:searchSongs *LyricWikz*Input:song, artist; *LyricWikz*Output:song, artist *chartLyricsS*tate:searchLyric_start	seen(song) = seen(artist) = 0 needed(lyricsId) = needed(lyricCheckSum) = 0 needed(artist) = needed(artistUrl) = -1 needed(song) = needed(songRank)= -1
4	*LyricWikz*State:s_5 *chartLyricsS*tate:searchLyric_start *chartLyricsO*peration: *None*	seen(song) = seen(artist) = 1 needed(song) = needed(artist) = 0
5	*LyricWikz*State:s_5; *LyricWikz*Operation: getSong *LyricWikz*Input: lyricId, song, lyricCheckSum, artist *LyricWikz*Output:song, artist, lyricCorrectUrl, Lyric *chartLyricsS*tate:searchLyric_start	No changes
6	*LyricWikz*State:s_6 *chartLyricsI*nput: lyricId, lyricCheckSum *chartLyricsO*utput: song, artist, artistUrl, lyricRank, Lyric, lyricCorrectUrl, ... *chartLyricsS*tate:searchLyric_start *chartLyricsO*peration:getLyric	seen(song) = seen(artist) = 2 seen(lyricCheckSum) = seen(lyricId) = 1 needed(song) = needed(artist) = 1 needed(lyricCorrectUrl) = needed(Lyric) = 1
7	*LyricWikz*State:s_6 *LyricWikz*Operation: *None* *chartLyricsS*tate:end *chartLyricsO*peration: *None*	seen(lyricCheckSum) = seen(lyricId) = 0 needed(song) = needed(artist) = 0 needed(lyricCorrectUrl) = needed(Lyric) = 0 needed(artistUrl) = needed(lyricRank) = -1

Table 5.1.: Mapping script generated for the case study in Section 5.2.2

6. Related works

Bultan at al. present in [103] a symbolic model checker for analyzing programs with unbounded integer domains. Programs are defined by an event-action language where atomic events are expressed by Presburger formulae over programs variables V. Semantics of programs is defined in terms of infinite transition systems where states are determined by values of variables. Events are atomic and they are represented with an enabling condition and an action where the condition constraints the states in which they occur and the action defines a transformation on variables of the program. The language for specification is a CTL-like temporal logic enriched with Presburger definable constraints over V. Solving CTL model-checking problem involves computation of least fixpoints over sets of programs states: the abstract interpretation of Cousot and Cousot [104] provides an approximation method to compute approximation of fixpoints. Model-checking is done conservatively: approximation technique admits false negatives, i.e., it allows the solver to indicate that a property does not hold when it really does. Three phases define the deciding procedure. In translation phase, the model-checker accepts as input a program which is defined by the event-action language. Programs are analyzed symbolically by means of symbolic execution techniques and they are represented by means of Presburger definable transition systems where Presburger formulae represents symbolically the transition relation and the set of program states. Then, the state space is partitioned in order to reduce the complexity of verification and regain decidability for some temporal property, like reachability. Finally, the analysis phase realizes procedures to solve conservatively verification problems. Backward and forward state exploration is performed during model-checking computation. The backward version starts with the set of states satisfying the property and performs, recursively, transformations of the current set by means of a predecessor function; if the computed set is contained in the set of initial states of the program then the property is satisfied. Conversely, the forward version starts from the initial configuration and try to reach set of states satisfying the property. Since the adopted language representing programs is Turing-complete, the process of computing sets of states may not terminates. Therefore, authors suggest a way to compute approximations of set representing

exact fixpoints for desired formulae such that the computing algorithms always terminate. In particular, if ϕ is a property and S_ϕ is the set of states satisfying ϕ, a lower (upper) approximation S^l_ϕ (S^u_ϕ) for ϕ is a set such that $S^l_\phi \subseteq S_\phi$ ($S^u_\phi \supseteq S_\phi$). It is worth noticing that $S^l_\phi \subseteq S^u_\phi$. If I is the set of initial configuration for the program, then verifying that $I \subseteq S^l_\phi$ entails $I \subseteq S_\phi$; conversely, if $I \supseteq S^u_\phi$ then $I \supseteq S_\phi$. However, if $I \not\subseteq S^l_\phi$ we can not conclude anything because it can be a false negative; and symmetrically, when $I \not\supseteq S^u_\phi$. In this case, we can compute a lower bound $S^l_{\neg\phi}$ of $\neg\phi$ such that if $I \cap S^l_{\neg\phi} \neq \emptyset$ and then we can yield a counterexample. Symmetrically, we can compute an upper bound $S^u_{\neg\phi}$ such that if $I \cap S^u_{\neg\phi} = \emptyset$ and then we can prove ϕ. To compute lower (upper) approximation for a formula, we may have to compute an upper (lower) approximation; for instance, in the case of negated formulae like $\phi = \neg\psi$, for which $S/S^u_\phi \subseteq S_{\neg\phi} \subseteq S/S^l_\phi$. Then, it holds that $S^l_\phi = S/S^u_{\neg\phi}$ and $S^u_\phi = S/S^l_{\neg\phi}$ where S is the universe set of states. From abstract interpretation, the widening technique is used to obtain procedures to approximate fixpoints which always terminate. In particular, to compute upper approximation authors generalizes the convex region widening operator which is proposed by Cousot and Halbwachs in [105]. Let V be the set of variables occurring in the program and (S, I, η, L) be the transition system defining the program where S is the set of states, I the set of initial states, $\eta \in S \times S$ is the transition relation and L is a labeling function which defines for every state a Presburger formula over V. The transition relation $\eta \in S \times S$ is defined by program events which are represented by Presburger formulae over the set $V \cup V'$, where V and V' is the set of current and next state variables. If X is a set of states, $pred(X) = \{s \in S \mod s' \in X \text{ and } (s, s') \in \eta\}$ is the predecessor set of X by means of the predecessor function *pred*. For instance, computing the set of states satisfying the CTL formula $\mathbf{EF}\phi$ amounts to compute a least fixpoint of the formula $f_{\mathbf{EF}\phi} = \sigma Z(\phi \vee \mathbf{EX}Z)$. The least fixpoint $\mu Z(\phi \vee \mathbf{EX}Z)$ of $f_{\mathbf{EF}\phi}$ is, exactly, the set $S_{\mathbf{EF}\phi}$ of states satisfying the formula $\mathbf{EF}\phi$, by monotonicity of f and by the Tarski theorem [106]. Given a fixpoint formula $\sigma Z.f$, where $\sigma \in \{\mu, \nu\}$, its k-th approximant $\sigma^k Z.f$ is a formula recursively defined as follows:

$$\mu^0 Z.f = false \quad \mu^{i+1} Z.f = f_{X \leftarrow \mu^i X.f}$$
$$\nu^0 Z.f = true \quad \nu^{i+1} Z.f = f_{X \leftarrow \nu^i X.f}$$

where $f_{Z \leftarrow \alpha}$ is a formula obtained by replacing all occurrence of Z in f by α. It is worth noticing that $S_{false} = \emptyset$ and $S_{true} = S$. All the formulae $\sigma^i Z.f$ are such that variable Z does not occur, i.e., Z is replaced i times; for this reason, we write $f^i(false)$ instead of $\sigma^i Z.f$. Each element of the

sequence $false, f_{\mathbf{EF}\phi}(false), f^2_{\mathbf{EF}\phi}(false), \ldots f^i_{\mathbf{EF}\phi}(false)$

- corresponds to a subset of the least fixpoint $S_{\mathbf{EF}\phi}$ of $\mathbf{EF}\phi$,
- and is such that $f^i_{\mathbf{EF}\phi}(\emptyset) \subseteq f^{i+1}_{\mathbf{EF}\phi}(\emptyset)$.

Then, since each iteration provides a lower bound of the exact fixpoint, authors suggest various method to decide a suitable number of iteration to obtain a conservative lower bound of a desired formula. Computing the sequence $\emptyset, f, f^2, \ldots$ can be done iteratively. Given a symbolic representation for ϕ, the set $S_{\mathbf{EX}\phi}$ of states satisfying $\mathbf{EX}\phi$ equals to $pred(S_\phi)$ where S_ϕ is the set of states satisfying ϕ. Then, starting from $S_0 = S_\phi$ each iteration $S_i = S_{i-1} \cup pred(S_i)$ is the i-th element $f^i_{\mathbf{EF}\phi}(\emptyset)$ of the sequence. When $S_i = S_{i-1}$, for some i, then $S_i = S_{\mathbf{EF}\phi}$ is the least fixpoint. Defining an upper approximation of the least fixpoint for $\mathbf{EF}\phi$ amounts to find a sequence $\emptyset, \hat{f}(\emptyset), \hat{f}^2(\emptyset), \ldots$ such that $\hat{f}^i(\emptyset) \supseteq f^i(\emptyset)$ for all i. To generate $\hat{f}^i(\emptyset)$ authors adopt a modified widening operator $\widehat{\bigtriangledown}$ such that if A, B are two sets then $A \cup B \subseteq A \widehat{\bigtriangledown} B$. The operator $\widehat{\bigtriangledown}$ is a generalization of the widening operator defined by Cousot and Halbwachs in [105] which handles *convex polyhedra*, i.e., conjunction of Presburger constraints. Experimental results, based on the standard Bakery algorithm and Ticket mutual-exclusion algorithm, show the effectiveness of the method when verification involves a mutual exclusion requirement $\neg\mathbf{EF}(pc_1 = C_1 \wedge pc_2 = C_2)$, where pc_i, C_i is the program counter and the label identifying the critical section of the process i, and a starvation-free property $\mathbf{AG}(pc_1 = W_1 \rightarrow \mathbf{AG}(pc_2 = W_2))$. Both the property are verified exactly, since the fixpoint computation converges. In the case of Ticket mutual-exclusion algorithm approximation technique is needed to have convergent fixpoint computation.

Schuele and Schneider provide in [107] a general algorithm to decide bounded LTL(L) model-checking of infinite state systems where the language L is a general underlying logic. The approach is quite different from previous by Bultan et al. since the system is explored only for a bounded number of times but it shares similarity in the fact they reduce model-checking problem to fixpoints verification. Differently to bounded model-checking of Biere et al. in [61], a LTL(L) formula ϕ is translated to an equivalent Büchi automaton $\mathcal{A}_\phi$ which is symbolically represented by means of a structure defining its transition relation and acceptance condition. Then, LTL(L) model-checking problem is reduced to μ-calculus model-checking problem modulo L, i.e., a verification of fixpoint problem for a given Kripke structure with respect to symbolic representations of $\mathcal{A}_\phi$ and the underlying language L. Whenever properties are not be proved or disproved over finite computation, their truth value can not

be defined. For this reason, authors adopt three-valued logic to evaluate the value of formulae whose components may have undefined value. Global and local model-checking of μ-calculus formulae are investigated and adapted to bounded model-checking. Bounded local model-checking follows an inductive style of reasoning. Proof for $M \models \phi$ are obtained by building a tableau by means of syntax-directed rules which use predecessor and successor functions on structure M. Bounded global model-checking is performed essentially by computing approximate fixpoint sets of a formula and by checking whether the initial condition I is a subset of the set of states f^k defined by the approximant. In particular, let V be the set of variables defining the systems; models which authors consider are Kripke structure of the form (S, I, R) where S is the set of states, I a Presburger definable set of initial states, $R \in S \times S$ is the transition relation which can be represented by Presburger formulae over the set $V \cup V'$ of current and next state variables. Bounded model-checking problem $M, s_0 \models \phi$ is reduced to checking whether there exists $q \in Q_0$ such that $M \times \mathcal{A}, (s_0, q) \models F$ where $\mathcal{A} = (Q, Q_0, \delta, F)$ is the symbolic Büchi automaton of ϕ. Most of the work of [107] and the use of symbolic representation of Büchi automaton are already introduced in [108], by the same authors, whose contribute is the definition of a hierarchy of Büchi automata (and, therefore, temporal formulae) for which infinite state bounded model-checking is complete. The language for specification of [108] is the quantifier-free fragment of Presburger LTL, LTL(PA), with past-time temporal modalities. Bounded model-checking problem is defined with respect to Kripke structures (S, I, R) and it is solved by means of reduction to satisfiability of Presburger formulae. In general, acceptance conditions of Büchi automata, requiring that some states are visited infinitely often, can not be handled immediately by bounded approaches which do not consider ultimately periodic models, like, for instance, bounded model-checking approach of Biere et al. [61] or the encoding of Büchi automata of deMoura in [109]. Therefore, Schule and Schneider follow a different approach, tailored to bounded verification, and focus on the analysis of some classes of LTL formulae, denoted $\mathrm{TL}_{\mathbf{F}}$ and $\mathrm{TL}_{\mathbf{G}}$, such that the corresponding Büchi automaton has a simpler accepting condition which does not involve condition on infinite models. $\mathrm{TL}_{\mathbf{F}}$ and $\mathrm{TL}_{\mathbf{G}}$ are the sets of LTL formulae such that each occurrence of a weak/strong temporal operator is negative/positive and positive/negative, respectively. As anticipated, LTL formulae are represented symbolically by an automaton which is built using the method proposed by Clarke et al. in [60] instead of Vardi-Wolper [51] construction. Formulae belonging to $\mathrm{TL}_{\mathbf{F}}$ and $\mathrm{TL}_{\mathbf{G}}$ can be represented by deterministic Büchi automata whose symbolic acceptance condition is

of the form $\mathbf{F}\psi$ and $\mathbf{G}\psi$, where ψ is a propositional formula. Informally, a symbolic automaton for LTL formula is a graph enriched with fairness constraints which recognizes its infinite models. Atomic propositions are introduced to represent symbolically control states of the graph and the structure is translated into a formula. For instance, a LTL formula $\phi\mathbf{U}\psi$, with $\phi, \psi \in$ PA, is represented by means of a symbolic automaton of one states q and one fairness accepting condition of the form: $\mathcal{A} = (\{q\}, true, p \Leftrightarrow \phi \vee \psi \wedge \mathbf{X}p, \mathbf{GF}(\phi \Rightarrow p))$ where *true* is the initial condition. According to the analysis in [110], condition $\mathbf{GF}(\phi \Rightarrow p)$ can be simplified to $\mathbf{F}(\phi \Rightarrow p)$. Unwinding these automata for bounded number of times k is straightforward and does not require a product with the Kripke structure. In order to reduce model-checking problem to satisfiability problem of Presburger formulae, control states are represented by new fresh integer variables and transition relation of automata is represented symbolically. Computation of the symbolic automaton is represented at each step by the set of formulae defining its transition relation. In the previous example, $p_i \Leftrightarrow \phi_i \vee \psi_i \wedge p_{i+1}$ for $i \in [0, k]$ and the acceptance condition is $\mathbf{F}(\phi_i \Rightarrow p_i)$. Let M be a Kripke structure and $s \in I$; LTL(PA) bounded model-checking is considered in both existential and universal version: $M, s \models \mathbf{A}\phi$, i.e., all runs of M starting from s satisfy ϕ, and $M, s \models \mathbf{E}\phi$, i.e., there exists a runs of M starting from s which satisfies ϕ. Let ϕ be a $\mathrm{TL}_{\mathbf{F}}$ formula such that $\mathbf{F}\zeta$ is the accepting condition of $\mathcal{A}_\phi$:

- $M, s \models \mathbf{E}\phi$ if, and only if, $\boldsymbol{x}_0 \wedge \bigwedge_{i=0}^{k} R(\boldsymbol{x}_i, \boldsymbol{x}_{i+1}) \wedge \bigvee_{i=0}^{k} \zeta_i$ is satisfiable.
- $M, s \models \mathbf{A}\phi$ if $\boldsymbol{x}_0 \wedge \bigwedge_{i=0}^{k} R(\boldsymbol{x}_i, \boldsymbol{x}_{i+1}) \Rightarrow \bigvee_{i=0}^{k} \zeta_i$ is satisfiable.

for some $k > 0$. Let ϕ be a $\mathrm{TL}_{\mathbf{G}}$ formula such that $\mathbf{F}\zeta$ is the accepting condition of $\mathcal{A}_\phi$:

- $M, s \not\models \mathbf{E}\phi$ if $\boldsymbol{x}_0 \wedge \bigwedge_{i=0}^{k} R(\boldsymbol{x}_i, \boldsymbol{x}_{i+1}) \wedge \bigvee_{i=0}^{k} \zeta_i$ is satisfiable.
- $M, s \not\models \mathbf{A}\phi$ if, and only if, $\boldsymbol{x}_0 \wedge \bigwedge_{i=0}^{k} R(\boldsymbol{x}_i, \boldsymbol{x}_{i+1}) \Rightarrow \bigvee_{i=0}^{k} \zeta_i$ is satisfiable.

for some $k > 0$.

Reducing model-checking problem to Presburger satisfiability is quite standard approach when dealing with infinite state systems. Finkel and Leroux in [4] study reachability and Demri et al. in [72] LTL(PA) model-checking problem for the class of *admissible* counter systems. We say that a partial function $f : \mathbb{N}^n \to \mathbb{N}^n$ is *affine* when, given Presburger formula $\varphi(x_1, \ldots, x_n)$ there exists a matrix $A \in \mathbb{Z}^{n \times n}$ and a vector $\boldsymbol{b}$ such that

$f(\boldsymbol{x}) = A\boldsymbol{x} + \boldsymbol{b}$ for every $\boldsymbol{x} \models_{\text{PA}} \varphi$. A counter system (Q, n, δ) is affine when every transition $q \xrightarrow{\xi} q'$ of δ is such that ξ is affine. Admissible counter systems are affine counter systems such that:

1. there is at most one transition between two control states,
2. the control graph is *flat*, i.e., every control states belong to at most one elementary cycle,
3. each cycle has the finite monoid property.

These condition are required to symbolically represent the effect of cycle of the system by means of a Presburger formula. Since affine relations are closed under composition, it is then possible to symbolically represent finite (and infinite) computations of systems. Flatness guarantees that the language of transitions, called *path schema* of a flat counter system is a finite union of bounded language of the form: $\ell = u_1(v_1)^* u_2(v_2)^* \dots (v_n)^* u_{n+1}$, where Σ is the set of symbols representing transitions of the systems and $u_i \in \Sigma^*$ and $v_i \in \Sigma^+$. Although each cycle v_i of the system can be represented by an affine relation $T \subseteq \mathbb{N}^n \times \mathbb{N}^n$ between counters, which is the result of composition of relations labeling cycle transitions, this does not immediately entail that its transitive and reflexive closure T^* is Presburger definable. Finite monoid property is a sufficient condition guaranteeing that relation $c^T \subseteq \mathbb{N}^n \times \mathbb{N}^n \times \mathbb{N}$ such that $(\boldsymbol{x}, \boldsymbol{y}, i) \in c^T$ if, and only if, $(\boldsymbol{x}, \boldsymbol{y}) \in T^i$, is Presburger definable; i.e., the effect of cycles at the i-the iteration is effectively definable. A relation $T = \{(\boldsymbol{x}, \boldsymbol{y}) \subseteq \mathbb{N}^n \times \mathbb{N}^n \mid \boldsymbol{y} = A\boldsymbol{x} + \boldsymbol{b}\}$ has finite monoid property when the set $A^* = \{A^i \mid i \in \mathbb{N}\}$ is finite. When T has the finite monoid property then T^* is Presburger definable. It is worth noticing that, given a matrix $A \in \mathbb{Z}^{n \times n}$, the problem of checking whether the set A^* is finite is decidable [111]. Authors prove a foundamental theorem for admissible counter systems characterizing the reachability set. Let (Q, n, δ) be a admissible counter system and $q, q' \in Q$ two control states. Then $R_{(q,q')}$ is effectively semilinear, i.e., it is possible to compute a Presburger formula $\varphi_{(q,q')}$ such that for every valuation $(\boldsymbol{x}, \boldsymbol{x'})$, $(\boldsymbol{x}, \boldsymbol{x'}) \models_{\text{PA}} \varphi_{(q,q')} \Leftrightarrow (\boldsymbol{x}, \boldsymbol{x'}) \in R_{(q,q')}$. Reachability is proved by Cortier [112] to be undecidable when finite monoid property is relaxed for flat affine counter systems. Demri et al. in [72] study decidability of model-checking problem for admissible counter systems with respect to first-order CTL* language over Presburger predicates. Decidability of first-order CTL* model-checking problem (which entails decidability of LTL(PA)) is proved by exploiting Presburger definability of the reachability set when models are re-

stricted to the class of admissible counter systems. Flatness guarantees that path schema are extended to infinite runs of the form: $\ell = u_1(v_1)^* u_2(v_2)^* \dots (v_z)^\omega$, where Σ is the set of symbols representing transitions of the systems and $u_i \in \Sigma^*$ and $v_i \in \Sigma^+$. The pair $(\ell, \mathbf{h})$, called *path description*, where $\mathbf{h} = h_1, \dots, h_{z-1}$ is a sequence of naturals and a $\ell = u_1(v_1)^* u_2(v_2)^* \dots (v_z)^\omega$ is a path schema, corresponds to a unique computation $\ell^{\mathbf{h}} = u_1(v_1)^{h_1} u_2(v_2)^{h_2} \dots (v_z)^\omega$. Given an admissible counter system, the set L of all path schema is finite and each path schema in L can be effectively computed. Decidability of model-checking is obtained by reduction to satisfiability of a Presburger formula. Closure under composition of affine relations and Presburger definability of reachability set are used to obtain for every $q \in Q$, two formulae $\varphi_q^\ell(h_1, \dots, h_{z-1}, x_1, \dots, x_n)$ and $\varphi_i^\ell(h_1, \dots, h_{z-1}, x_1, \dots, x_n, y_1, \dots, y_n, i)$, with $\ell = u_1(v_1)^* u_2(v_2)^* \dots (v_z)^\omega$, such that:

- $(\boldsymbol{h}, \boldsymbol{x}) \models_{\mathrm{PA}} \varphi_q^\ell$ if, and only if, there is an infinite run from $(q, \boldsymbol{x})$ obtained from $\ell^{\boldsymbol{h}}$.
- $(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{x}', t) \models_{\mathrm{PA}} \varphi_i^\ell$ if, and only if,
  1. $(\boldsymbol{h}, \boldsymbol{x}) \models_{\mathrm{PA}} \varphi_q^\ell$
  2. t-th tuple of counters' values is $\boldsymbol{x}'$.

for all valuation $(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{x}', t)$. Intuitively, the second formula φ_i^ℓ is used to represent semantics of CTL* formulae with respect to the infinite run such that φ_q^ℓ holds. Therefore, let $\mathcal{S} = (Q, n, \delta)$ be an admissible counter system, $(q, \boldsymbol{x})$ be a configuration of $\mathcal{S}$ and ϕ be a CTL* formula over Presburger predicates; CTL* model-checking $\mathcal{S} \models \phi$ is reduced to satisfiability of the formula

$$\bigvee_{\ell \in L} (\exists \boldsymbol{h}\ \varphi_q^\ell(\boldsymbol{h}, \boldsymbol{x}) \wedge t(0, \phi))$$

where t is a map which translates CTL* formulae to an equivalent first-order formula. Let $R(\boldsymbol{x}, \boldsymbol{y}) \subseteq \mathbb{N}^n \times \mathbb{N}^m$ be a relation where $\boldsymbol{y}$ are variables for quantification not occurring in the system. The map t, restricted to the LTL fragment, is defined as follows (full definition is detailed in [72]):

$$\begin{aligned}
t(i, R(\boldsymbol{x}, \boldsymbol{y})) &\overset{\text{def}}{\Leftrightarrow} \forall \boldsymbol{x}'\ \varphi_i^\ell(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{x}', i) \Rightarrow R(\boldsymbol{x}', \boldsymbol{y}) \\
t(i, \mathbf{X}\phi) &\overset{\text{def}}{\Leftrightarrow} \exists j\ t(j, \phi) \wedge j = i + 1 \\
t(i, \zeta \mathbf{U} \psi) &\overset{\text{def}}{\Leftrightarrow} \exists j\ (t(j, \psi) \wedge \forall z (i \leq z < j) \wedge t(z, \zeta)) \\
t(i, \exists y \phi) &\overset{\text{def}}{\Leftrightarrow} \exists y\ t(i, \phi)
\end{aligned}$$

Complexity for CTL* model-checking problem is not investigated.

Reducing problems to Presburger arithmetic is exploited by Hague and Lin in [83] to compute the reachability set of pushdown systems enriched with reversal-bounded counters (PCo) and, possibly, extended with discrete clocks (PCC). Transitions are labeled by DL formulae whose variables are taken from the set of counters and clocks. Decidability and complexity analysis of reachability problem and control state LTL model checking problem for both the class PCo and PCC is based on reduction to Presburger formulae. Authors prove that reachability for PCo is NP-complete while LTL model-checking problem is coNEXPTIME-complete. When discrete clocks are considered reachability is NEXPTIME-complete and complexity of LTL model-checking problem is the same as PCo. All the complexity results are obtained by considering a unary encoding for the number of reversal r, while all numeric constants and counter increments are encoded in binary. Nonetheless, a careful reading of reduction shows that a binary encoding of r does not affect complexity result obtained for model-checking problem but it involves an exponential grow up for reachability problem. Complexity upper bounds result from the polynomial time reduction of problems to satisfiability of a Presburger formula. Given a PCo $\mathcal{P}$, authors build first a pushdown automaton $\mathcal{P}'$, with the same set of control states as $\mathcal{P}$, which over approximates $\mathcal{P}$ and simulates $\mathcal{P}$ by disregarding counter information. $\mathcal{P}'$ represents transitions of $\mathcal{P}$ by means of a symbolic alphabet Σ' which specify:

- how counters change, by storing n quadruple of the form $(counter_i, u, j, l)$ where $i \in [1, n]$ is a counter index, $u \in \mathbb{Z}$ is the update value, j defines the current mode and $l \in \{0, 1\}$ represents whether the action changes the mode;
- which constraint ξ is valid at mode i.

Being $\mathcal{P}'$ a pushdown system, its Parikh image $\chi_q(\boldsymbol{y})$, where q is a control state of $\mathcal{P}$ and $\boldsymbol{y}$ represents symbols in Σ', is semilinear and it can be effectively computed. Since $\mathcal{P}'$ only guesses the sequence of mode vectors characterizing runs of $\mathcal{P}$, there are runs of $\mathcal{P}'$ which are not valid in $\mathcal{P}$. In order to correctly represent runs of $\mathcal{P}$, authors provide a formula which filters invalid run and asserts the existence of a valid sequence of modes respecting counter tests and updates which are guessed by $\mathcal{P}'$. It is worth noticing that constants defining the system define a partition of $\mathbb{N}$ for each counter. In particular, let m be the number of constants and $c_1, \ldots, c_m$ be the sequence of ordered constants; then, the domain $\mathbb{N}$ of each counter is partitioned into $2m$ distinct regions. Vector modes are defined analogously to Section 2.6.2 and they also contain information

on regions. Legal runs of $\mathcal{P}$ are represented by the following formula:

$$\exists \boldsymbol{y} \exists \boldsymbol{m}_0 \dots \boldsymbol{m}_{N-1} \left(\begin{array}{c} \mathit{initial}(\boldsymbol{m}_0) \wedge \mathit{goodSeq}(\boldsymbol{m}_0, \dots, \boldsymbol{m}_{N-1}) \wedge \\ \wedge \mathit{respect}(\boldsymbol{y}, \boldsymbol{m}_0 \dots \boldsymbol{m}_{N-1}) \wedge \mathit{endConfig}(\boldsymbol{y}) \wedge \\ \wedge \chi_q(\boldsymbol{y}) \end{array} \right)$$

where:

- $\mathit{initial}(\boldsymbol{m}_0)$ enforces the initial mode $\boldsymbol{m}_0$ to be compatible with the initial configuration, i.e., counters region and update vector.
- $\mathit{goodSeq}(\boldsymbol{m}_0, \dots, \boldsymbol{m}_{N-1})$ forces $\boldsymbol{m}_0, \dots, \boldsymbol{m}_{N-1}$ to be a valid sequence of mode vectors, i.e., the number of reversal is bounded for all counter and a reversal occurs for counter x when the sign of the update of x changes.
- $\mathit{respect}(\boldsymbol{y}, \boldsymbol{m}_0 \dots \boldsymbol{m}_{N-1})$ defines behavior of counters. If a counter is non-incrementing (non-decrementing) then only non-negative (non-positive) increments are allowed. Secondly, the value of each counter at the beginning and at the end of each mode must respect the region defined by the current mode vector. Finally, if a transition is fired then its labeling constraint is satisfied by the current configuration.
- $\mathit{endConfig}(\boldsymbol{y})$ asserts that the final configuration $(q, \boldsymbol{y})$ is reachable, i.e., the value of counters is $\boldsymbol{y}$ and it is equal to the effect of the run reaching q.

The previous formula has cubic size in the size of $\mathcal{P}$, (unary) r and the number of counter n. Model-checking problem is solved with the standard automata-theoretic approach of Vardi-Wolper [51] and in particular, it is reduced to control-state repeated reachability on the product system $\mathcal{P} \times \mathcal{A}_{\neg\phi}$, where $\mathcal{A}_{\neg\phi}$ is the Büchi automaton of $\neg\phi$. By reversal-boundedness of counters, all infinite runs of $\mathcal{P} \times \mathcal{A}_{\neg\phi}$ stabilize, i.e., some counters are always non-decreasing and some others are constant. Then, all infinite runs of $\mathcal{P} \times \mathcal{A}_{\neg\phi}$ can be represented by a first finite subrun defined by $\mathcal{P} \times \mathcal{A}_{\neg\phi}$ as PCo and a second subsequent infinite run where $\mathcal{P} \times \mathcal{A}_{\neg\phi}$ is treated as pushdown system with no counters. The pushdown systems is defined from the original PCo by allowing only transitions which do not decrement counters whose value does not stabilize. Therefore, it is possible to build a PCo $\mathcal{P}''$ which simulates $\mathcal{P} \times \mathcal{A}_{\neg\phi}$ for the initial subrun and then, nondeterministically, stops simulating $\mathcal{P} \times \mathcal{A}_{\neg\phi}$ as PCo and starts simulation of the second subrun where $\mathcal{P} \times \mathcal{A}_{\neg\phi}$ is treated as pushdown system. Repeated reachability problem is reduced

to a reachability problem over $\mathcal{P}''$ with respect to a special control state $final$ which is reached if, and only if, $\mathcal{P}''$ simulates both the phase of $\mathcal{P} \times \mathcal{A}_{\neg\phi}$. Beside the representation of runs reaching $final$, as it is defined by the previous formula, auxiliary Presburger formulae are used to filter runs of $\mathcal{P}''$ such that some control states, visited when $\mathcal{P} \times \mathcal{A}_{\neg\phi}$ is treated as pushdown system, are repeated infinitely often. Lower bound for LTL model-checking and reachability are obtained by reducing the Knapsack problem, which is known to be NP-complete: given $a_1, \ldots, a_k, b \in \mathbb{N}$ with binary encoding, the problem is to decide whether $\sum_{i=1}^{k} a_i x_i = b$ for some $x_1, \ldots, x_k \in \{0, 1\}$. Authors propose a reduction which builds a PCo $\mathcal{P}$ with one 1-reversal bounded counter. $\mathcal{P}$ initializes the counter to 0 and repeats for each $i \in [1, k]$ the following action: guess the values for x_i then add $a_i x_i$ to the counter. At the end, the PCo checks whether the counter is equal to b by subtracting b and checking if the result is null. This operation involves only one reversal. If the result is 0 then $\mathcal{P}$ execute a special action ok. The LTL formula which is verified over $\mathcal{P}$ is $\mathbf{F}ok$. The lower bound can be proved for PCo where counters can only be compared against zero and incremented by $\{-1, 0, +1\}$. Therefore, in order to perform the sum $a_i x_i$ and the subtraction of b the stack is used to store their binary representation and to compute weighted sum with increments in $\{-1, 0, +1\}$.

Lin and Libkin [113] provide a general result about infinite state model-checking problem. They define algorithmic meta theorem for LTL model-checking which can be used to infer decidability (and complexity) of the problem for a large class of infinite state systems. Authors identify semantic conditions on word/tree automatic transition systems which guarantee decidability of LTL model-checking problem:

c1: reachability relation is effectively computable and defined by special non deterministic automata, called synchronized transducers;

c2: the class of systems is closed under product with finite systems.

Although many systems satisfy condition **c1**, many classes do not benefit of closure as required by **c2**. Therefore, in order to regain decidability, authors consider various fragments of LTL, along with restriction of condition **c2** to dag-like finite state systems. Removing condition **c2** is possible but at the cost of worse complexity bound. Fundamental notion which authors use in their work is the automatic presentation of transition systems, i.e., infinite transition system that can be finitely represented by means of automata and (synchronized) transducers. Given an alphabet Σ, a (synchronized) transducer T is a non deterministic automaton defining a relation $R \subseteq \Sigma^* \times \Sigma^*$ which reads words of the form

$w \oplus w'$. Alphabet of T is defined by symbols of $\Sigma_\varepsilon \times \Sigma_\varepsilon$ where $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ is the alphabet Σ enriched with a special character ε which fills blank positions when two words of different length are considered. For instance, if $w = abb$ and $w' = bbabb$ then $w \oplus w' = \begin{bmatrix} abb\varepsilon\varepsilon\varepsilon \\ bbaabb \end{bmatrix}$. The relation R recognized by T is the set of pair $\{(w, w') \in \Sigma^* \times \Sigma^* \mid w \oplus w' \in \mathscr{L}(T)\}$. Let A be a finite set of action symbols and $\mathcal{S} = (S, \rightarrow_{a \in A})$ be a transition system over the actions A where S is a set of states and $\rightarrow_{a \in A} \subseteq S \times S$ be a relation on S. An infinite transition system $\mathcal{S}$ can be finitely represented by a structure, called *presentation*, $\theta = (\mathcal{A}, \{T_a\}_{a \in A})$ where $\mathcal{A}$ is a finite state automaton over Σ and each T_a is a transducer over Σ. The induced "automatic" transition system $\mathcal{S}(\theta)$ is such that $S = \mathscr{L}(\mathcal{A})$ and $\rightarrow_{a \in A} = \mathscr{L}(T_a)$, for each $a \in A$; i.e., $s \rightarrow_{a \in A} s'$ if, and only if, $s \oplus s' \in \mathscr{L}(T_a)$, where $s, s' \in S$. Condition **c1** guarantees the effective computability of the control state repeated reachability set which can be computed in polynomial time. Formally, condition **c1** is satisfied when, given a presentation θ, there exists an algorithm which computes a transducer T^+ recognizing the transitive closure $(\bigcup_{a \in A} \rightarrow_a)^+$. Given an automatic transition system $\mathcal{S}(\theta)$ and a finite state automaton $\mathcal{A}'$ over the same alphabet Σ, the set S^∞ of states, from which there exists an infinite path of $\mathcal{S}(\theta)$ visiting states of $\mathscr{L}(\mathcal{A}')$ infinitely often, is regular and a finite state automaton $\mathcal{A}^\infty$, such that $S^\infty = \mathscr{L}(\mathcal{A}^\infty)$, is computable in polynomial time. Second condition **c2** requires that the class of infinte-state systems is closed under product with finite system. Then, if $\theta \in \mathcal{C}$ and F is a finite system then $F \times \mathcal{S}(\theta) \in \mathcal{C}$. Given a LTL formula, when both conditions **c1** and **c2** are realized, checking whether $(\mathcal{S}(\theta), s_0) \models \phi$ is polynomial in the size of the presentation θ and exponential in the size of the formula ϕ. In particular, the set of states of $\mathcal{S}(\theta)$ satisfying $\neg\phi$ is regular and recognized by a finite state automaton which can be computed with the previous complexity bound. Although certain classes satisfy condition **c1**, many of them do not benefit of a regular reachability set when they are combined with finite systems. Therefore, authors analyze some fragments of LTL and a subclass of Presburger definable systems in order to regain decidability. The first LTL fragments which authors consider is denoted LTL_{det}. It is proposed by Maidl in [114] and its syntax is defined as follows: $\phi := p \mid \mathbf{X}p \mid \phi \wedge \phi \mid (p \wedge \phi) \vee (\neg p \wedge \phi) \mid (p \wedge \phi)\mathbf{U}(\neg p \phi) \mid (p \wedge \phi)\mathbf{U}^{>}(\neg p \phi)$, where p is an atomic proposition. The second fragment $\text{LTL}(\mathbf{F}_s, \mathbf{G}_s)$ is defined by LTL formulae in which temporal operators $\mathbf{U}$ and $\mathbf{X}$ are substituted by the derived operator $\mathbf{F}_s\phi \equiv \mathbf{X}\mathbf{F}\phi$ and $\mathbf{G}_s\phi \equiv \neg\mathbf{F}_s\neg\phi$. When extra assumption on presentation θ is given such that states and relations $\rightarrow_a$ are Presburger definable, decidability of model-checking for full LTL

is retained. In this case, presentations are of the form $\theta = (\xi, \eta_a)$ where ξ, η are existential Presburger formulae with k free variables, for some $k \geq 1$. Decidability of LTL model-checking problem for $\mathcal{S}(\theta)$ requires an additional monotonicity condition between states of $\mathcal{S}(\theta)$: let $\boldsymbol{s}, \boldsymbol{s'} \in \mathbb{N}^k$ two tuples of naturals representing two states of $\mathcal{S}(\theta)$ then $\boldsymbol{s} \rightarrow_a \boldsymbol{s} + \boldsymbol{b}$ implies $\boldsymbol{s'} \rightarrow_a \boldsymbol{s'} + \boldsymbol{b}$ for some $\boldsymbol{b} \in \mathbb{N}^k$ and some action $a \in A$, where $\rightarrow_a = \{(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{N}^k \times \mathbb{N}^k \mid (\boldsymbol{x}, \boldsymbol{y}) \models_{\text{PA}} \eta_a\}$. Authors apply meta theorems to obtain decidability and complexity results over various classes of infinite state systems. Although some of them are already known, meta theorems can be applied to the following classes: pushdown and concurrent pushdown systems, prefix-recognizable systems, communication-free Petri nets, some classes of process algebra and rewriting systems.

Decidable fragments of first-order temporal logic are considered in [73] by Hodkinson et al. Although some axiomatizations of first-order temporal logic are known, various incompleteness results induce authors to study useful fragments between propositional and first-order temporal logic. The language is different from the one defined by Thomas in [71] where the temporal modality X can be applied on first-order objects (i.e., terms). Moreover, Hodkinson et al. are interested in satisfiability and they do not consider model-checking problem which requires a formalism defining the interpretation of first-order variables. In other words, variables do not vary over time and their temporal behavior is not relevant. Informally, languages which authors investigate are obtained by restricting both the first-order part and the temporal part. The first restriction is needed to select suitable first-order fragments which benefits of classical already known decidability properties. On the other hand, temporal operators which are used to describe behaviors of objects manipulated by the first order part, are restricted to *monodic* formulae. In particular, a formula is monodic when its subformulae, involving a temporal operator, have at most one free variable. The two conditions result in decidable temporal fragments of first-order temporal logic over various structure of time: $(\mathbb{N}, <)$, $(\mathbb{Z}, <)$, $(\mathbb{Q}, <)$ and, when finite domain of variables are considered, also $(\mathbb{R}, <)$. Authors focus on satisfiability problem of temporal formulae without equality or function symbols which are interpreted in models with constant domain for the first-order part and with strict linear order of time. The two classes `TL` and `TS` of languages differ in the access of time: while `TL` has only implicit access to time by means of the standard temporal operator **U** (until) and **S** (since), `TS` is a two-sorted first-order language such that one sort refers to positions of time and the other is dedicated to first-order domain. The grammar of the two language is analogous to the grammar defined in Section 3. For instance, the formula $\exists x \forall y (P(x) \mathbf{U} Q(y))$ belongs to `TL` but it is not

monadic because the formula $P(x)\mathbf{U}Q(y)$ has two different free variables; while $\exists t_1 \exists t_2 (t1 < t2 \wedge \exists x (P(t_1, x) \Rightarrow Q(t_2, x)))$ belong to TS. Beside the notion of monodic formulae, also the number of parameters of a formula can be restricted. In particular, a formula of is said to be *monadic* when it only contains unary predicates and propositional variables. Authors prove first a result defining the undecidability edge for the satisfiability problem of TL. They show that the monadic fragments of TL, such that formulae have at most two free variable, is enough to encode undecidable problems like recurrent tiling problem for $\mathbb{N} \times \mathbb{N}$, when $(\mathbb{N}, <)$ and $(\mathbb{Z}, <)$ structures the time, and the halting problem for Turing machines, when he time $(\mathbb{N}, <)$ and $(\mathbb{Z}, <)$ and the class of linear order even in the case of finite first-order domains. Both the undecidability proofs use temporal formulae involving subformulae of the form $\phi \mathbf{U} \psi$ with two free variables. This motivates the analysis of monodic fragment of TL which benefits of decidable satisfiability problem. In particular, authors use a specific representation of models called *quasimodel* which synthetically defines, for a given TL formula, which subformulae hold at each position of time. Then, decidability of satisfiability problem for monodic TL is reduced to satisfiability of monadic second-order logic. Given a TL formula ϕ we can write a monadic second-order logic which assert the existence of a quasi-model which satisfies ϕ. Motivated by the non-elementary complexity of satisfiability of monadic second-order logic, authors provide a different proof of decidability for the monodic fragment of first-order temporal logic. Decidability is obtained by an algorithm which checks whether there exists an ultimately periodic model for the formula. Since the number of ultimately periodic model is finite and bounded by a double exponential in the number of subformulae defining the original formula, then it is possible to decide the problem of satisfiability in elementary time provided that satisfiability of first-order fragment adopted to define TL is elementary. Authors investigate also relationship between the language TL and TS. They provide a translation of TL formulae into TS formulae which is used to show that TL and the monodic fragment of TL are expressive complete, i.e., equivalent, with respect to the fragment of $\mathtt{TS}_{1t}$ and $\mathtt{TS}_1$. The fragment of $\mathtt{TS}_{1t}$ ($\mathtt{TS}_{1x}$) is the set of TS formulae without subformulae of the form $\forall x \phi$ ($\forall t \phi$) such that ϕ contains more than one free temporal (domain) variable; the language $\mathtt{TS}_1$ is the union $\mathtt{TS}_{1t} \cup \mathtt{TS}_{1x}$. Moreover, the fragments of monodic TL and TS formulae involving at most two variable are decidable when the structure of the time is one of the following: $(\mathbb{N}, <)$, $(\mathbb{Z}, <)$, $(\mathbb{Z}, <)$, $(\mathbb{R}, <)$ (only for finite first-order domain), the class of all finite strict linear orders and any first-order definable class of strict linear orders. Finally, the monadic fragment of both monodic TL and monodic TS (predicate symbols have

domain $t \times D$ or t, where t is the temporal explicit variable) have decidable satisfiability problem. It is worth noticing that the monadic fragment of TS such that predicates have only one variable for the sort referring to the time is proved to have the same expressiveness as the LTL in the Kamp's thesis [46].

6.1. Related tools

Recent developments of infinite-state verification are often associated with an intense activity of development of tools supporting the theory. This section gives a brief overview of the main and most famous ones related to this thesis. Nonetheless the vastness and rapid growth of research makes the investigation a quite hard task, we can individuate two main different area of development: solvers for base theories and high-level frameworks for verification.

Decidability of Presburger arithmetic along with its relevance in reducing problem like reachability and model-checking over counter systems, has led to development of efficient tools for deciding satisfiability of the full language PA or its fragments like quantifier-free PA or DL. Omega library [115] is one of the best known candidate for handling PA formulae. The Omega project has two major components. One component is a system for manipulating sets of affine constraints over integer variables which was originally designed as a decision test for the existence of integer solutions to affine constraints. Other than providing positive or negative answers to a satisfiability problem, the library constitutes of several routines for elimination of existentially quantified variables, elimination of redundant constraints, simplification of formulae involving negations and for verifying implications between formulae. The extended version of the Omega library is a complete system for simplifying and verifying Presburger formulas. The class of solvers for base theory include also Satisfiability Modulo Theory solvers. SMT-solvers implement different decision procedure for separated theories and provide algorithms to solve satisfiability of combination of them. Nelson-Oppen schema is one possible method for combining disjoint theories. However, many implemented algorithms exploit interaction with an underlying SAT-solver: SMT instances are translated to boolean SAT instances and first solved by a SAT solver. This approach, which is referred to as the *eager* approach, has the advantage of discovering immediately unsatisfiability derived from boolean semantics of formulae. On the other hand, the loss of the semantics of the underlying theories may penalize resolution of trivial facts. This led to the development of a number of SMT solvers which

integrate Boolean reasoning of a DPLL-style search with theory-specific solvers that handle conjunctions of predicates from a given theory (*lazy* approach). Not all SMT-solvers support quantifiers since they may lead to undecidability. We list main SMT-solver and their main characteristics which we can be used in our implemented tool presented in Chapter 7.

yices: it is developed by SRI International. Yices decides the satisfiability of propositional formulas that mix uninterpreted function symbols and equality with interpreted symbols from the following theories: linear real and integer arithmetic, arrays, fixed-size bit-vectors, recursive datatypes, tuples, lambda expressions and quantifiers. It also provides algorithm to produce unsatisfiable core and maximal sat of formulae.

MathSat: is a DPLL-based SMT-solver which supports theories of equality and uninterpreted functions, linear real and integer arithmetic and difference logic. It also supports unsatisfiable core algorithm.

z3: equality over uninterpreted functions and predicate symbols, real and integer arithmetic (with limited support for non-linear arithmetic), bit-vectors, arrays, tuple, records, enumeration types and algebraic (recursive) data-types. It also supports partial decision procedure for universal quantifier and non-linear arithmetic.

Barcelogic: it is a DPLL-based SMT-solver which uses congruence closure to solve EUF theory. It supports EUF and difference logic theories.

OpenSMT: it is a DPLL-based SMT-solver which supports EUF, difference logic and linear arithmetic over $\mathbb{Z}$ and $\mathbb{Q}$ and bit-vectors.

CVC3: it works with several built-in theories: linear arithmetic over $\mathbb{Z}$ and $\mathbb{Q}$, arrays, tuples, records, inductive data types, bit vectors, and equality over uninterpreted function symbols. It also supports quantifiers.

Solvers for theories are usually integrated within high-level verification tools which implement procedure for model-checking or higher form of analysis like program analysis, secure programming development, automatic systems synthesis, network protocols verification and cryptography.

FAST: [116] Fast is a tool for reachability/safety verification of counter systems. It relies on symbolic representation of both the systems

and the specification which are defined by Presburger formulae. The tool was first intended to handle flat counter systems, for which symbolic representation of cycles (acceleration) makes the procedure complete. Although many systems do not have a flat representation, it is often possible to provide an equivalent flat system with the same reachability set which can be correctly verified. When flatness can not be exploited and reachability set is not recursive, termination of the procedure can be obtained by a user-guided interaction. Counter systems are specified by means of a language defining control states and transitions. Verification is done by computing backward and forward analysis by means of functions $pre(t, S, k)$ and $post(t, S, k)$, respectively, which computes the set of reachable configuration from S by using transitions defined in t and by accelerating cycles of length longer than k. For flat counter systems the two functions compute the exact reachability set; when the system is not flattable, both are semi-algorithms. Set theoretical functions $\cup, \cap, \neg, \subseteq$ are used to manipulate set of configurations. For instance, reachability properties are verified with $post(t, S, k) \subseteq GoodSet$ while safety is verified by checking whether $pre(t, BadSet, k) \cap InitSet = \emptyset$, where $GoodSet$ and $InitSet$ are two Presburger definable sets of configurations.

SAL: SAL [117] is a framework for specification and analysis of concurrent systems. It provides finite and infinite model-checker and other tools for checking deadlocks in finite concurrent systems and for generating random executions of finite and infinite state systems. The tool is aimed at verification of concurrent systems which are specified by a language allowing synchronous and asynchronous composition of interacting components. Finite model-checking is performed symbolically and also by explicit state algorithms, while infinite state verification is realized by k-inductive proof, which we recall in Section 2.7.3. Verification is tailored to clock-based automata, like Timed automata, endowed with infinite domain variables. In order to avoid the use of continuous clock, whose Zeno behavior (infinite occurrence of transitions without time progress) makes k-induction useless, timeout-based verification is realized by using time-progress transitions which update timeouts while discrete transitions are instantaneous since they leave time unchanged. Timeouts are temporal objects which store the time at which future discrete transitions will be taken. When an action occurs, since its timeout is elapsed, the action is realized and the timeout is updated to a new value, which is strictly larger than

the current time. The benefit of timeouts is that they enforce a deterministic progress of time. There are no states in which both time-progress and discrete transitions are enabled. When a time-progress transition is enabled, time is advanced to the point where the next discrete transition is enabled. Therefore, all variables of the systems evolve in discrete steps.

Averest: Averest (http://www.averest.org/) is aimed to development of reactive systems. It consists of the various components: beside auxiliary modules, it provides a symbolic model checker and a tool for hardware/software synthesis. Specification language for systems is a synchronous languages (QUARTZ). Statements are "executed" as micro steps in zero time and consumption of time is explicitly enforced by partitioning the micro steps of computation into macro steps that all take the same amount of logical time. Concurrent threads automatically synchronize at the end of a macro step. Specifications given in linear temporal logic (LTL) are translated to ω-automata and/or fixpoints in CTL and the μ-calculus, as we describe before about [107]. Verification is provided by a symbolic model checker for finite and infinite state systems. Finite sets are encoded by their characteristic propositional functions which are represented by means of binary decision diagrams (BDDs) [57] while infinite states are represented by Presburger arithmetic formulae. As the specification language, Averest uses the μ-calculus which subsumes CTL and LTL. It includes a translator from FairCTL, an extension of CTL with fairness constraints, to the μ-calculus.

7. ae^{2}zot - a Tool for Infinite State Verification

ae^{2}zot implements k-bounded satisfiability for LTL(**FO**) and CLTLB(L). Languages L are those supported by SMT-solver such that the theory of EUL is decidable. ae^{2}zot is a plugin of a bigger environment for verification, called Zot, which is constituted by various dedicated plugins. The tool supports different logic languages through a multi-layered approach. An interesting feature of Zot is its ability to support different encodings of temporal logic as SMT problems. Plug-ins approach encourages experimentation, as they are expected to be quite simple, compact and extendible. At the moment, a variant of the eventuality encoding presented in [63] is supported, along with (approximated) dense-time MTL [118], and a bi-infinite encoding [119]. ae^{2}zot plug-in further enriches the available functionality.

Zot offers three basic usage modalities:

1. Bounded satisfiability checking (BSC): given as input a specification formula, the tool returns a (possibly empty) history (i.e., an execution trace of the specified system) which satisfies the specification. An empty history means that it is impossible to satisfy the specification.

2. Bounded model checking (BMC): given as input an operational model of the system and a temporal formula ϕ, the tool returns a (possibly empty) history (i.e., an execution trace of the system) which satisfies it ϕ.

3. History checking and completion (HCC): The input can also contain a partial (or complete) history H. In this case, if H complies with the specification, then a completed version of H is returned as output, otherwise the output is empty.

7.1. Tool structure

Main modules around ae^{2}zot plug-in are `kripke.lisp`, `smt-interface.lisp` and `trio-utils.lisp`. We briefly describes their main functionality.

The file kripke.lisp contains the basic data structure and the definition of the generics. Their definition are quite intuitive and directly explained by comments.

```
(defclass kripke ()
   ((the-k       :accessor kripke-k :type fixnum)


   (the-list    :accessor kripke-list :type hash-table)
; formula -> integer (usually an hash-table)
   (the-back    :accessor kripke-back :type array)
        ; integer -> formula (idem)
   (sf-prop     :accessor kripke-prop :type list)
        ; list of propositions used in the formula
   (sf-bool     :accessor kripke-bool :type list)
        ; list of used boolean subformulae
   (sf-futr     :accessor kripke-futr :type list)
        ; list of used future-tense subformulae
   (sf-past     :accessor kripke-past :type list)
        ; list of used past-tense subformulae
   (max-prop    :accessor kripke-maximum :type fixnum)
        ; used propositions maximum
   (the-formula :accessor kripke-formula :type list)))
        ; this should contain the resulting prop. formula
        ; for the SAT-solver

(defgeneric call (self obj the-time &rest other-stuff)
(:documentation
  "Call translates a formula/proposition and a time
instant into a suitable representation for SAT/SMT solver"))
```

The `smt-interface.lisp` defines the module which provides the abstraction of the underlying SMT solver. Main functions are:

```
(defun to-smt-and-back (the-kripke smt-solver) ...)

(defun translate-smt-output (k) ...)
```

Function `to-smt-and-back` has two parameters:

the-kripke is the structure containing the internal representation of the formula and the corresponding translation into a formula which is processed by an SMT-solver. Then, it will comply with syntax of

the chosen solver and it will invoke the suitable decision procedure for a supported logic.

smt-solver is a flag defining which solver has to be used.

`translate-smt-output (k)` translates the output produced by an SMT-solver to a human readable format. Moreover, it also filter atomic propositions and variables occurring in the original formula.

The module `trio-utils.lisp` provides syntactic abstraction on the languages; for instance, the temporal operator **F** is encoded as `somf`.

The core `ae`^{2}`zot` realizes essentially three functionality:

- it extends the class `kripke` into `eezot-kripke` in order to handle LTL(**FO**) and CLTLB(L) formulae;
- it implements a parser for input formulae which analyzes the syntactic structure and fills a data structure `data` of class `eezot-kripke`.
- it builds the QF-EUL formula which realizes the reduction from k-bounded satisfiability or model-checking problem to satisfiability for QF-EUL.

The class `eezot-kripke` has the following attributes

- `((the-arith :accessor kripke-arith :type list)` is a list of atomic formulae occurring in the input formula.
- `(the-timed-arith :accessor kripke-timed-arith :type hash-table)` is a list of atomic formulae which take values from the model $\widehat{\sigma}_k$.
- `(the-timed-arith-terms :accessor kripke-timed-arith-terms :type list)` is a list of all a.t.t.'s occurring in the input formula.
- `(the-arith-arith-futr :accessor kripke-arith-futr :type list)` is a list of all *future* a.t.t.'s of the form $\mathrm{X}^i x$ occurring in the input formula.
- `(the-arith-arith-past :accessor kripke-arith-past :type list)` is a list of all *past* a.t.t.'s of the form $\mathrm{X}^i x$ occurring in the input formula.
- `(the-arith-arith-ops :accessor kripke-timed-arith-ops :type list)` is a list of all atomic terms defined by arithmetic operators in the set $\{+, -, *, /\}$

Main function defining the functionality of `ae`^{2}`zot` is implemented in function `zot`:

```
(defun zot (the-time spec
    &key
    (loop-free nil)
    (transitions nil)
    (negate-transitions nil)
    (declarations nil)
    (smt-solver :z3)
    (logic :QF_UFIDL)
    (smt-assumptions nil)
    (no-loop nil)
    (with-time t)
    (periodic-vars nil)
    )
```

where `the-time spec` are the bound k and the input formula, respectively. Function `zot` performs the following actions:

1. it pushes negation on atomic formulae by means of function `deneg`: `(setf (formula (deneg (trio-to-ltl spec))))` where function `trio-to-ltl` performs a syntactic translation of operators.

2. It defines the data structure of class `eezot-kripke` by performing `(make-kripke the-time formula))` which parses the input formula in `fma` and fills the data structure `data` of class `eezot-kripke` defining all the previous list of formulae.

3. It produces the final formula by implementing the encoding presented in Section 3.3.1 and 3.3.5. All the following functions read the data structure defined by `(make-kripke the-time formula))`:

   - Lisp function `loopConstraints()` produces formulae defining $|LoopConstrains|_k$.
   - Lisp function `gen-bool()` produces formulae defining $|PropConstraints|_k$.
   - Lisp functions `defun gen-futr()`, `defun gen-past1()` and `defun gen-past2()` produce formulae defining $|TempConstraints|_k$.
   - Lisp function `lastStateFormula()` produces formulae defining $|LastStateConstraints|_k$.
   - Lisp function `gen-evt-futr()` produces formulae defining $|Eventually|_k$.
   - Lisp functions `gen-arith-futr()`, `gen-arith-past()` produces formulae defining semantics of terms and atomic formulae.

- Lisp function `gen-arith-constraints ()` produces formulae defining arithmetic formulae when the linear integer arithmetic is used as language L.

4. Finally, the QF-EUL formula is written into the file `output.smt.txt` which is input of an SMT solver. The solver is invoked and its output `output.1.txt` is processed to produce output for the user which is written into the file `output.hist.txt`. This is done by the module `smt-interface.lisp` by calling `(to-smt-and-back data smt-solver)` where `data` and `smt-solver` are the data structure containing the QF-EUL formula and the solver, respectively.

8. Conclusions and future works

We provide a novel approach to solve the satisfiability problem for the existential fragment of LTL(**FO**) by reducing the problem to satisfiability over ultimately periodic (symbolic) models, which we call k-bounded satisfiability. Although the finite representation uv, of length k, captures only symbolic infinite models of the form uv^ω, it is possible to solve the general satisfiability problem when the language admits models with periodic representation (in this case models of the form uv^ω are still representative of infinite models) and when the number of instances of k-bounded satisfiability to be solved is bounded. This is the case of CLTLB(L), with L is a suitable fragments of Presburger arithmetic. Given a CLTLB(L) formula ϕ and a natural k, our tool reduces an instance of k-bounded satisfiability into an instance of satisfiability of a formula in the decidable theory QF-EUFL (theories supported are QF-UFLIA or QF-UFLRA). Since decision procedures for these theories are implemented by many SMT-solvers, we are able to effectively solve the k-bounded satisfiability problem for CLTLB(L), where L belongs to QFP. Then, for some L, we prove that k-bounded satisfiability is complete with respect to the satisfiability problem for CLTLB(L) formulae. Given a CLTLB(L) formula ϕ, by checking a finite number of k-bounded satisfiability problem of formula ϕ, we can answer to the satisfiability problem of ϕ. Our decision procedure is effective also when we are dealing with model-checking problem: counter systems with transitions labeled by IPC* formulae or Büchi automata where transitions are labeled by CLTLB(L) formulae, provided that L is a suitable fragments of Presburger arithmetic, can be easily represented by CLTLB formulae. Moreover, k-bounded satisfiability can be used to solve reversal-bounded model-checking problem for the class of reversal-bounded counter systems. In fact, we can exploit the property such that reversal-bounded runs satisfying a CLTLB(L) property are ultimately periodic and their length is bounded. This fact allow us to conclude that reversal-bounded model-checking problem can be solved by means of k-bounded satisfiability. Bounded length of runs satisfying property along with the existence of ultimately periodic runs guarantee that our bounded approach is complete. We show the optimal NEXPTIME-completeness of the reversal bounded model-checking problem. In order to implement decision pro-

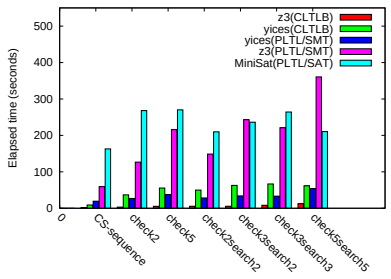
cedures, we demonstrate that given a counter system, a temporal formula ϕ and $r \geq 0$, one can build effectively a Presburger formula encoding the set of configurations $(q, \boldsymbol{x})$ such that there is an r-reversal-bounded infinite run ρ from $(q, \boldsymbol{x})$ such that ϕ is satisfied by ρ. Finally, we have also characterized the complexity of some reversal-bounded reachability problems involved in the solution of original problem.

Future works. Although we provided only one example of language, i.e., CLTLB(L) where L is IPC* or structure $(D, <, =)$ with $D \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$, for which k-bounded satisfiability is complete, and a simple example of reversal bounded model-checking problem, we believe that k-bounded approach can be used to solve general satisfiability and model-checking problems also for other formalisms. In particular, we would like to investigate satisfiability of CLTL$^1_1$(DL) and model-checking problem of one-counter automata with DL constraints. It is already known, in fact, that formulae of CLTL$^1_1$(DL) can be translated to Büchi automata whose accepting runs are ultimately periodic and of bounded length. Moreover, ultimately periodicity can be exploited also for the monodic fragment of LTL(**FO**) already presented in Section 6. Satisfiable formulae of LTL(**FO**) admit, in fact, ultimately periodic model of bounded length.

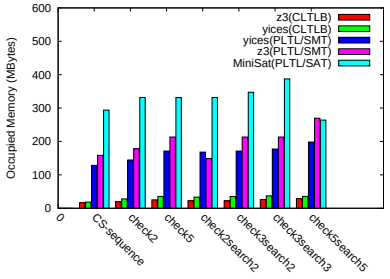
A. Appendix

A.1. Case study SOA: experimental results

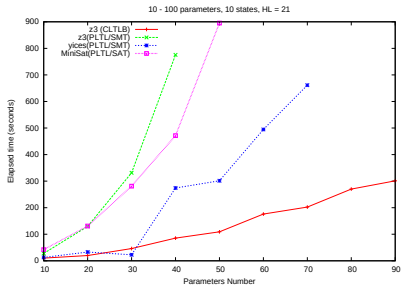
Experimental results concerning service substitutability problem which refer to Section 5.2.



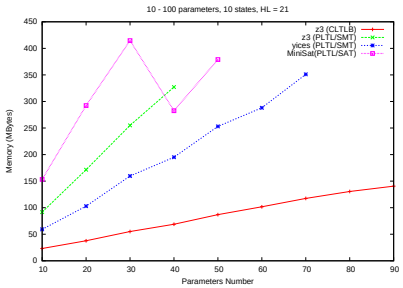
(a) Elapsed times on the second set of experiments



(b) Memory occupations on the second set of experiments



(c) Elapsed times on the third set of experiments



(d) Memory occupations on the third set of experiments

Figure A.1.: Substitutability experimental Results

A.2. Lisp code of hysteresis: case study from Section 5.1

In this section, we provide the Lisp code defining the hysteresis system of Section 5.1. Parts composing the code are commented in order to explain their function.

The first two lines of code loads the plug-in ae^{2}zot and defines the list of used package.

```
(asdf:operate 'asdf:load-op 'ae2zot)
(use-package :trio-utils)
```

To define variables occurring in the formula as well as their dependency with the time, we use `define-var` and `define-tvar` which are two macros defined in `trio-utils.lisp`. Macro `define-var` instantiates variables with no time dependency, i.e., variables which do not vary over time. The two parameters x_L and x_H do not have temporal behavior and then they are defined by means of `define-var`. For defining variables, or in general, uninterpreted functions and predicates varying over time, macro `define-tvar` includes time in the definition of the specified object. For instance, a time-variant object p with domain $\mathbb{Z}$ is defined as $p : \mathbb{N} \to \mathbb{Z}$ where $\mathbb{N}$ represents time.

```
(define-tvar 'x *int*)
(define-tvar 'y *int*)
(define-tvar 'd *int*)
(define-var 'x0 *int*)
(define-var 'x1 *int*)
```

Definition of predicates given in Section 5.1 are Lisp functions with a generic parameter `z`.

```
(defun up-down (z)
  (&& ([=] z 0) ([=] (yesterday z) 1)))


(defun down-up (z)
  (&& ([=] z 1) ([=] (yesterday z) 0)))


(defun low (z)
  ([=] z 0))

(defun high (z)
  ([=] z 1))
```

Following formulae define the behavior of the hysteresis. Each formula is instantiated as a Lisp variable.

```
(defvar tr-up-down
  (alwf
    (-> (and
          (yesterday (&& ([<] (-V- x) (-V- x1)) (high (-V- y))))
          ([>=] (-V- x) (-V- x1)))
        (up-down (-V- y)))))


(defvar tr-down-up
  (alwf
    (-> (and
          (yesterday (&& ([>] (-V- x) (-V- x0)) (low (-V- y))))
          ([<=] (-V- x) (-V- x0)))
        (down-up (-V- y)))))


(defvar s-up-down
  (alwf
    (-> (up-down (-V- y))
        (and
          (yesterday ([<] (-V- x) (-V- x1)))
          ([>=] (-V- x) (-V- x1))))))


(defvar s-down-up
  (alwf
    (-> (down-up (-V- y))
        (and
          (yesterday ([>] (-V- x) (-V- x0)))
          ([<=] (-V- x) (-V- x0))))))


(defvar low-state
  (alwf
    (-> (low (-V- y))
        (since
          (until (low (-V- y)) (down-up (-V- y)))
          (up-down (-V- y))))))
```

```
(defvar high-state
  (alwf
    (-> (high (-V- y))
        (since
          (until (high (-V- y)) (up-down (-V- y)))
          (down-up (-V- y))))))


(defvar high-low-state-is
  (alwf (|| (low (-V- y)) (high (-V- y)))))


(defvar init
  (&&
    ([=] (-V- x) 3)
    ([=] (-V- y) 0)))


(defvar continuous-x
  (alwf (||
          ([=] (next (-V- x)) ([+] (-V- x) 1))
          ([=] (next (-V- x)) ([-] (-V- x) 1)))))


(defvar Ncontinuous-x
  (alwf  ([=] (next (-V- x)) ([+] (-V- x) (-V- d)))))


(defvar safe-state
  (alwf (&&
          (-> ([=] (-V- y) 1) ([<=] (-V- x) (-V- x1)))
          (-> ([=] (-V- y) 0) ([>=] (-V- x) (-V- x0))))))


(defvar thresholds
      ([>] (-V- x1) (-V- x0)))
```

The definition of the system is the conjunction of all the previous formulae.

```
(defvar syst
  (&&
```

```
    init
    high-low-state-is
    high-state
    low-state
    tr-up-down
    tr-down-up
    s-up-down
    s-down-up
    continuous-x
    thresholds))
```

The solver is invoked by the function `zot` requiring the bound k of instant of time and the CLTLB formula. The key `:logic` specifies which type of theory the solver shall use. `QF-UFLIA` is the theory of quantifier-free linear integer arithmetic formulae with uninterpreted functions.

```
(ae2zot:zot
  15
  (&&
    syst
    (!! safe-state))
  :logic :QF_UFLIA)
```

A.3. Periodicity

Let us consider the atomic formula ϕ of the form $\sum_j a_j x_j \equiv_c k$. We recall below how ϕ is equivalent to a positive Boolean formula with atomic formulae of the form $x_j \equiv_{c'} k'$ and c' divides c. Hence, if C is the lcm of all the constants c appearing in atomic formulae of the form $t \equiv_c k$ in guards of $\mathcal{S}$, then the lcm C' of all constants c' appearing in atomic formulae of the form $x_j \equiv_{c'} k'$ after transformation, divides C (hence the size of C' is smaller than the size of C). Let us now explain how atomic formulae of the form $x_j \equiv_{c'} k'$ are obtained. Let $S = \{\boldsymbol{c} \in \{0, \ldots, c-1\}^n : (\sum_j \boldsymbol{c}(j)) \equiv_c k\}$. We have the following logical equivalence

$$\phi \Leftrightarrow \bigvee_{\boldsymbol{c} \in S} (\bigwedge_{j \in [1,n]} a_j x_j \equiv_c \boldsymbol{c}(j))$$

It remains to explain how a linear congruence $a_j x_j \equiv_c \boldsymbol{c}(j)$ can be transformed into an equivalence atomic formula of the form $x_j \equiv_{c'} k'$.

Let us consider the linear congruence $ax \equiv_c b$. If $a < 0$, then we consider $-ax \equiv_c -b + nc$ so that $-b + nc \in [0, c-1]$ (which can be

computed in polynomial time). So, without any loss of generality, we can assume that in $ax \equiv_c b$, $a \geq 0$ and $b \in [0, c-1]$. The formula $ax \equiv_c b$ reduces to $\perp$ if $gcd(a,c)$ does not divide b. Otherwise $ax \equiv_c b$ is equivalent to $\frac{a}{gcd(a,c)}x \equiv_{\frac{c}{gcda,c}} \frac{b}{gcd(a,c)}$, say $a'x \equiv_{c'} b'$ where a' and c' are coprime. Since a' and c' are coprime, with the Extended Euclidean Algorithm, one can compute n_1, n_2 such that $n_1 a' + n_2 c' = 1$ (this can be done in polynomial time, see e.g. [120]). Consequently, $x \equiv_{c'} n_1 b'$, which can be transformed into $x \equiv_{c'} b''$ with $b'' \in [0, c'-1]$.

Let $c_1, \ldots, c_\alpha$ be constants occurring in simplified atomic formulae of the form $x \equiv_c k$ and C be their lcm. For $\boldsymbol{n} \in [0, c_1 - 1] \times \cdots \times [0, c_\alpha - 1]$, there is a unique $c \in [0, C-1]$, such that for all $u \in \mathbb{N}$, ($u \equiv_{c_1} \boldsymbol{n}(1)$ and $\cdots$ and $u \equiv_{c_1} \boldsymbol{n}(\alpha)$) iff $u \equiv_C c$ (Chinese Remainder Theorem). Hence, in order to encode the truth value of $x \equiv_{c_i} k$, it is sufficient to store the unique c such that $x \equiv_C c$. That is the principle of the transformation given in Section 4.3.2. It is worth noting that the value C obtained after the transformation of atomic formulae of the form $t \equiv_c k$ is in quadratic size in N.

Hence, for all atomic formulae $t \equiv_c k$ under consideration, for all $\boldsymbol{x} \in \mathbb{N}^n$ and $\tilde{\boldsymbol{x}} \in [0, C-1]^n$ such that for $i \in [1, n]$, we have $\boldsymbol{x}(i) \equiv_C \tilde{\boldsymbol{x}}(i)$, we have

(PER) $\boldsymbol{x}(t) \equiv_c k$ iff $\tilde{\boldsymbol{x}}(t) \equiv_c k$.

Bibliography

[1] Yehoshua Bar-Hillel, M. Perles, and E. Shamir. On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, 14:143–172, 1961. Reprinted in Y. Bar-Hillel. (1964). *Language and Information: Selected Essays on their Theory and Application*, Addison-Wesley 1964, 116–150.

[2] Géraud Sénizergues. The equivalence problem for deterministic pushdown automata is decidable. In *Proceedings of the 24th International Colloquium on Automata, Languages and Programming*, ICALP '97, pages 671–681, London, UK, 1997. Springer-Verlag.

[3] Oscar H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *J. ACM*, 25(1):116–133, 1978.

[4] Alain Finkel and Jérôme Leroux. How to compose presburger-accelerations: Applications to broadcast protocols. In *FSTTCS*, pages 145–156, 2002.

[5] Hubert Comon and Véronique Cortier. Flatness is not a weakness. In Peter Clote and Helmut Schwichtenberg, editors, *Computer Science Logic*, volume 1862 of *LNCS*, pages 262–276, Heidelberg, 2000. Springer.

[6] Stéphane Demri and Deepak D'Souza. An automata-theoretic approach to constraint LTL. *Inf. Comput.*, 205(3):380–415, 2007.

[7] Marcello M. Bersani, Achille Frigeri, Angelo Morzenti, Matteo Pradella, Matteo Rossi, and Pierluigi San Pietro. Bounded reachability for temporal logic over constraint systems. In Nicolas Markey and Jef Wijsen, editors, *TIME*, pages 43–50, Los Alamitos, 2010. IEEE Computer Society.

[8] Marcello M. Bersani, Achille Frigeri, Matteo Rossi, and Pierluigi San Pietro. Completeness of the bounded satisfiability problem for constraint ltl. In *RP*, page to appear, 2011.

[9] Marcello Bersani and Stéphane Demri. The complexity of reversal-bounded model checking. 2011.

[10] M. Bersani and S. Demri. The complexity of reversal-bounded model checking. Technical Report LSV-11-10, LSV, ENS Cachan, France, May 2011.

[11] StÃľphane Demri. Decidable problems for counter systems. Technical report, ENS Cachan, 2010.

[12] M. Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In *Comptes Rendus du premier congrès de mathématiciens des Pays Slaves, Warszawa*, pages 92–101, 1929.

[13] Derek C. Oppen. A $2^{2^{2^{pn}}}$ upper bound on the complexity of presburger arithmetic. *Journal of Computer and System Sciences*, 16(3):323 – 332, 1978.

[14] M. J. Fischer and M. O. Rabin. Super-exponential complexity of presburger arithmetic. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1974.

[15] Jeanne Ferrante and Charles Rackoff. A decision procedure for the first order theory of real addition with order. *SIAM J. Comput.*, 4(1):69–76, 1975.

[16] Bruno Scarpellini. Complexity of subcases of presburger arithmetic. *Transactions of The American Mathematical Society*, 284:203–203, 1984.

[17] Chr. Papadimitriou. On the complexity of integer programming. *Journal of the Association for Computing Machinery*, 28(4):765–768, 1981.

[18] Stéphane Demri and Régis Gascon. Verification of qualitative $\mathbb{Z}$ constraints. In *CONCUR*, pages 518–532, 2005.

[19] R. J. Parikh. Language generating devices. Technical report, MIT Res. Lab. Elect., Quart. Prog. Rept., 1961.

[20] Rohit J. Parikh. On context-free languages. *J. ACM*, 13:570–581, October 1966.

[21] Seymour Ginsburg and Edwin H. Spanier. Semigroups, Presburger Formulas, and Languages. *Pacific Journal of Mathematics*, 16(2):285–296, 1966.

[22] Seymour Ginsburg and Edwin H. Spanier. Bounded algol-like languages. *Transactions of The American Mathematical Society*, 113:333–333, 1964.

[23] John McCarthy. Towards a mathematical science of computation. In Cicely M. Popplewell, editor, *Information Processing 62: Proceedings of IFIP Congress 1962*, pages 21–28, Amsterdam, 1963. North-Holland.

[24] Jerry R. Burch and David L. Dill. Automatic verification of pipelined microprocessor control. In *Proceedings of the 6th International Conference on Computer Aided Verification*, CAV '94, pages 68–80, London, UK, 1994. Springer-Verlag.

[25] Greg Nelson and Derek C. Oppen. Simplification by cooperating decision procedures. *ACM Trans. Program. Lang. Syst.*, 1:245–257, October 1979.

[26] Peter J. Downey, Ravi Sethi, and Robert Endre Tarjan. Variations on the common subexpression problem. *J. ACM*, 27:758–771, October 1980.

[27] Shuvendu K. Lahiri and Madanlal Musuvathi. An efficient decision procedure for utvpi constraints. In *Frontiers of Combining Systems*, pages 168–183, 2005.

[28] Boris Cherkassky and Andrew Goldberg. Negative-cycle detection algorithms. In Josep Diaz and Maria Serna, editors, *Algorithms âĂŤ ESA '96*, volume 1136 of *Lecture Notes in Computer Science*, pages 349–363. Springer Berlin / Heidelberg, 1996.

[29] V. R. Pratt. Two easy theories whose combination is hard. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1977.

[30] Julius R. Buchi. On a Decision Method in Restricted Second-Order Arithmetic. In *International Congress on Logic, Methodology, and Philosophy of Science*, pages 1–11. Stanford University Press, 1962.

[31] M. Y. Vardi. A temporal fixpoint calculus. In *Proceedings of the 15th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '88, pages 250–259, New York, NY, USA, 1988. ACM.

[32] Pierre Wolper. Temporal logic can be more expressive. In *Proceedings of the 22nd Annual Symposium on Foundations of Computer Science*, pages 340–348, Washington, DC, USA, 1981. IEEE Computer Society.

[33] Moshe Y. Vardi and Pierre Wolper. Reasoning about infinite computations. *Information and Computation*, 115:1–37, 1994.

[34] Marvin L. Minsky. Recursive unsolvability of post's problem of "tag" and other topics in theory of turing machines. *Annals of Mathematics*, 74:437–453, 1961.

[35] Oscar H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *J. ACM*, 25(1):116–133, 1978.

[36] Anthony Widjaja To. Parikh images of regular languages: Complexity and applications. *CoRR*, abs/1002.1464, 2010.

[37] R. Howell and L. Rosier. An analysis of the nonemptiness problem for classes of reversal-bounded multicounter machines. *J. Comput. Syst. Sci.*, 34(1):55–74, 1987.

[38] E. Gurari and O. Ibarra. The complexity of decision problems for finite-turn multicounter machines. In *ICALP'81*, volume 115 of *LNCS*, pages 495–505. Springer, 1981.

[39] Oscar H. Ibarra, Jianwen Su, Zhe Dang, Tevfik Bultan, and Richard A. Kemmerer. Counter machines and verification problems. *Theor. Comput. Sci.*, 289(1):165–189, 2002.

[40] Z. Dang, O. Ibarra, and P. San Pietro. Liveness verification of reversal-bounded multicounter machines with a free counter. In *FSTTCS'01*, volume 2245 of *LNCS*, pages 132–143. Springer, 2001.

[41] Zhe Dang, Oscar H. Ibarra, and Pierluigi San Pietro. Liveness verification of reversal-bounded multicounter machines with a free counter. In *FSTTCS*, pages 132–143, 2001.

[42] Patrick Blackburn, Johan F. A. K. van Benthem, and Frank Wolter. *Handbook of Modal Logic, Volume 3 (Studies in Logic and Practical Reasoning)*. Elsevier Science Inc., New York, NY, USA, 2006.

[43] Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57, Washington, DC, USA, 1977. IEEE Computer Society.

[44] Nicolas Markey. Temporal logic with past is exponentially more succinct, concurrency column. *Bulletin of the EATCS*, 79:122–128, 2003.

[45] E. Allen Emerson. Temporal and modal logic. In *HANDBOOK OF THEORETICAL COMPUTER SCIENCE*, pages 995–1072. Elsevier, 1995.

[46] Johan Anthony Willem Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California at Los Angeles, 1968.

[47] Dov Gabbay, Amir Pnueli, Saharon Shelah, and Jonathan Stavi. On the temporal analysis of fairness. In *Proceedings of the 7th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '80, pages 163–173, New York, NY, USA, 1980. ACM.

[48] P. Wolper. The tableau method for temporal logic: an overview. *Logique et Analyse*, 28:119–136, 1985.

[49] Orna Lichtenstein and Amir Pnueli. Propositional temporal logics: Decidability and completeness. *Logic Journal of The Igpl / Bulletin of The Igpl*, 8:55–85, 2000.

[50] Y. Kesten, Z. Manna, H. Mcguire, and A. Pnueli. A decision algorithm for full propositional temporal logic. In *Lecture Notes in Computer Science*, volume 697, pages 97–109. Springer-Verlag, 1993.

[51] Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification. In *Proc. 1st Symposium on Logic in Computer Science*, pages 332–344, Cambridge, 1986.

[52] A. Prasad Sistla and Edmund M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32(3):733–749, 1985.

[53] D. Harel. Recurring dominoes: making the highly undecidable highly un-derstandable. *Theory of Computation*, 102:51–71, 1985.

[54] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time(preliminary report). In *Proceedings of the fifth annual ACM symposium on Theory of computing*, STOC '73, pages 1–9, New York, NY, USA, 1973. ACM.

[55] W. J. Savitch. Relationship between non-deterministic and deterministic tape classes. *Journal of Computer and System Sciences*, 1970.

[56] Jerry R. Burch, Edmund Melson Clarke, Kenneth L. McMillan, David L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. In *Logic in Computer Science*, pages 428–439, 1990.

[57] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput.*, 35:677–691, August 1986.

[58] Dexter Kozen. Results on the propositional µ-calculus. In *Proceedings of the 9th Colloquium on Automata, Languages and Programming*, pages 348–359, London, UK, 1982. Springer-Verlag.

[59] Mordechai Ben-Ari, Zohar Manna, and Amir Pnueli. The temporal logic of branching time. In *Proceedings of the 8th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '81, pages 164–176, New York, NY, USA, 1981. ACM.

[60] E. Clarke, O. Grumberg, and K. Hamaguchi. Another look at ltl model checking. In *Formal Methods in System Design*, pages 415–427. Springer-Verlag, 1994.

[61] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu. Symbolic model checking without BDDs. In *TACAS*, pages 193–207, 1999.

[62] Orna Kupferman and Moshe Y. Vardi. Model checking of safety properties. *Form. Methods Syst. Des.*, 19:291–314, October 2001.

[63] Armin Biere, Keijo Heljanko, Tommi A. Junttila, Timo Latvala, and Viktor Schuppan. Linear encodings of bounded LTL model checking. *Logical Methods in Computer Science*, 2(5), 2006.

[64] Edmund M. Clarke, Daniel Kroening, Joël Ouaknine, and Ofer Strichman. Completeness and complexity of bounded model checking. In *VMCAI*, pages 85–96, 2004.

[65] Leonardo Mendonça de Moura, Carlos José Pereira de Lucena, and Arndt von Staa. The spider environment. *Softw., Pract. Exper.*, 29(2):99–124, 1999.

[66] Daniel Kroening and Ofer Strichman. Efficient computation of recurrence diameters. In *VMCAI*, pages 298–309, 2003.

[67] Mary Sheeran, Satnam Singh, and Gunnar Stålmarck. Checking safety properties using induction and a sat-solver. In *Proceedings of the Third International Conference on Formal Methods in Computer-Aided Design*, FMCAD '00, pages 108–125, London, UK, 2000. Springer-Verlag.

[68] Leonardo De Moura, Harald Ruess, and Maria Sorea. Bounded model checking and induction: From refutation to verification (extended abstract, category a. In *Proceedings of the 15th International Conference on Computer Aided Verification, CAV 2003, volume 2725 of Lecture Notes in Computer Science*, CAV 2003, pages 14–26. Springer, 2003.

[69] Olaf Burkart, Didier Caucal, Faron Moller, and Bernhard Steffen. Verification on infinite structures, 2000.

[70] Ahmed Bouajjani, Rachid Echahed, and Peter Habermehl. On the verification problem of nonregular properties for nonregular processes. In *Proceedings of the 10th Annual IEEE Symposium on Logic in Computer Science*, pages 123–, Washington, DC, USA, 1995. IEEE Computer Society.

[71] Wolfgang Thomas. *Languages, automata, and logic*, pages 389–455. Springer-Verlag New York, Inc., New York, NY, USA, 1997.

[72] Stéphane Demri, Alain Finkel, Valentin Goranko, and Govert van Drimmelen. Model-checking ctl* over flat presburger counter systems. *Journal of Applied Non-Classical Logics*, 20(4):313–344, 2010.

[73] Ian M. Hodkinson, Frank Wolter, and Michael Zakharyaschev. Decidable fragment of first-order temporal logics. *Ann. Pure Appl. Logic*, 106(1-3):85–134, 2000.

[74] Stéphane Demri and Régis Gascon. The effects of bounding syntactic resources on Presburger LTL. In *TIME*, pages 94–104. IEEE Computer Society, 2007.

[75] Stéphane Demri. Ltl over integer periodicity constraints: (extended abstract). In *FoSSaCS*, pages 121–135, 2004.

[76] Hubert Comon and Yan Jurski. Multiple counters automata, safety analysis and presburger arithmetic. In *CAV*, pages 268–279, 1998.

[77] Rajeev Alur and Thomas A. Henzinger. A really temporal logic. *J. ACM*, 41(1):181–204, 1994.

[78] Marvin L. Minsky. *Computation: finite and infinite machines.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1967.

[79] Philippe Balbiani and Jean-François Condotta. Computational complexity of propositional linear temporal logics based on qualitative spatial or temporal reasoning. In *Proceedings of the 4th International Workshop on Frontiers of Combining Systems*, FroCoS '02, pages 162–176, London, UK, 2002. Springer-Verlag.

[80] Ph. Schnoebelen. The complexity of temporal logic model checking. In *Advances in Modal Logic*, pages 393–436, 2002.

[81] Dov M. Gabbay. The declarative past and imperative future: Executable temporal logic for interactive systems. In *Temporal Logic in Specification*, pages 409–448, London, UK, 1987. Springer-Verlag.

[82] Shmuel Safra. On the complexity of omega-automata. In *FOCS*, pages 319–327, 1988.

[83] Matthew Hague and Anthony Widjaja Lin. Model checking recursive programs with numeric data types. In *CAV*, pages 743–759, 2011.

[84] E. Kopczynski and A. To. Parikh Images of Grammars: Complexity and Applications. In *LICS'10*. IEEE, 2010.

[85] Javier Esparza. Decidability and complexity of petri net problems - an introduction. In *Petri Nets*, pages 374–428, 1996.

[86] C. Rackoff. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6(2):223–231, 1978.

[87] I. Borosh and L. Treybig. Bounds on positive integral solutions of linear diophantine equations. *Proocedings of The American Mathematical Society*, 55:299–304, 1976.

[88] V. De Antonellis, M. Melchiori, L. De Santis, M. Mecella, E. Mussi, B. Pernici, and P. Plebani. A layered architecture for flexible web service invocation. *Softw. Pract. Exper.*, 36(2):191–223, 2006.

[89] K. Verma, K. Gomadam, A. Sheth, J. Miller, and Z. Wu. The METEOR-S approach for configuring and executing dynamic web processes. Technical Report LSDIS 05-001, LSDIS Lab, University of Georgia, Athens, Georgia, 2005.

[90] Luca Cavallaro, Elisabetta Di Nitto, and Matteo Pradella. An automatic approach to enable replacement of conversational services. In *Proc. ICSOC/ServiceWave*, pages 159–174, 2009.

[91] Luca Cavallaro, Elisabetta Di Nitto, Patrizio Pelliccione, Matteo Pradella, and Massimo Tivoli. Synthesizing adapters for conversational web-services from their WSDL interface. In *Proc. SEAMS*, 2010.

[92] Microsof Research. Z3: An efficient SMT solver. http://research.microsoft.com/en-us/um/redmond/projects/z3/, 2009.

[93] Computer Science Lab, SRI international. Yices: An SMT solver. http://yices.csl.sri.com/, 2009.

[94] L. Cavallaro, G. Ripa, and M. Zuccalà. Adapting service requests to actual service interfaces through semantic annotations. In *Proc. PESOS*, 2009.

[95] Matteo Pradella, Angelo Morzenti, and Pierluigi San Pietro. Refining real-time system specifications through bounded model- and satisfiability-checking. In *ASE*, pages 119–127, 2008.

[96] C. Drumm. *Improving Schema Mapping by Exploiting Domain Knowledge*. PhD thesis, Universitat Karlsruhe, Fakultat fur Informatik, 2008.

[97] M. Fredj, N. Georgantas, V. Issarny, and A. Zarras. Dynamic service substitution in service-oriented architectures. In *Proc. SERVICES*, pages 101–104, 2008.

[98] Thomas Schaeck and Richard Thompson. WS-ResourceProperties. http://docs.oasis-open.org/wsrp/Misc/, 2003.

[99] A. Brogi and R. Popescu. Automated generation of BPEL adapters. In *Proceedings of ICSOC*, pages 27–36, 2006.

[100] OASIS. Web Services Business Process Execution Language Version 2.0. `http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf`, 2007.

[101] Wil M. P. van der Aalst and Arthur H. M. ter Hofstede. YAWL: yet another workflow language. *Inf. Syst.*, 30(4):245–275, 2005.

[102] J. A. Martìn and E. Pimentel. Automatic generation of adaptation contracts. In *Proc. FOCLASA*, 2008.

[103] Tevfik Bultan, Richard Gerber, and William Pugh. Model-checking concurrent systems with unbounded integer variables: symbolic representations, approximations, and experimental results. *ACM Trans. Program. Lang. Syst.*, 21:747–789, July 1999.

[104] Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL*, pages 238–252, 1977.

[105] Patrick Cousot and Nicolas Halbwachs. Automatic discovery of linear restraints among variables of a program. In *POPL*, pages 84–96, 1978.

[106] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[107] Tobias Schuele and Klaus Schneider. Bounded model checking of infinite state systems. *Formal Methods in System Design*, 30:51–81, 2007. 10.1007/s10703-006-0019-9.

[108] Tobias Schüle and Klaus Schneider. Bounded model checking of infinite state systems: exploiting the automata hierarchy. In *MEMOCODE*, pages 17–26, 2004.

[109] Leonardo Mendonça de Moura, Harald Rueß, and Maria Sorea. Lazy theorem proving for bounded model checking over infinite domains. In *CADE*, pages 438–455, 2002.

[110] Klaus Schneider. Improving automata generation for linear temporal logic by considering the automaton hierarchy. In *LPAR*, pages 39–54, 2001.

[111] Arnaldo Mandel and Imre Simon. On finite semigroups of matrices. *Theoretical Computer Science*, 5(2):101–111, 1977.

[112] Véronique Cortier. About the decision of reachability for register machines. *ITA*, 36(4):341–358, 2002.

[113] Anthony Widjaja To and Leonid Libkin. Algorithmic metatheorems for decidable ltl model checking over infinite systems. In *FOSSACS*, pages 221–236, 2010.

[114] Monika Maidl. The common fragment of ctl and ltl. In *FOCS*, pages 643–652, 2000.

[115] Wayne Kelly, Vadim Maslov, William Pugh, Evan Rosser, Tatiana Shpeisman, and David Wonnacott. The omega library interface guide. Technical report, University of Maryland at College Park, College Park, MD, USA, 1995.

[116] Sébastien Bardin, Jérôme Leroux, and Gérald Point. Fast extended release. In *CAV*, pages 63–66, 2006.

[117] Leonardo Mendonça de Moura, Sam Owre, Harald Rueß, John M. Rushby, Natarajan Shankar, Maria Sorea, and Ashish Tiwari. Sal 2. In *CAV*, pages 496–500, 2004.

[118] Carlo A. Furia, Matteo Pradella, and Matteo Rossi. Automated verification of dense-time mtl specifications via discrete-time approximation. In *FM*, pages 132–147, 2008.

[119] Matteo Pradella, Angelo Morzenti, and Pierluigi San Pietro. The symmetry of the past and of the future: bi-infinite time in the verification of temporal properties. In *Proc. ESEC/SIGSOFT FSE*, pages 312–320, 2007.

[120] E. Bach and J. Shallit. *Algorithmic Number Theory, Volume I: Efficient Algorithms*. MIT Press, 1996.