UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de Ingenieros Informáticos

A Uniform Approach to Language Containment Problems

DOCTORAL THESIS

Submitted for the degree of Doctor by:

Kyveli Doveri
Master Degree in Mathematics

Madrid, 2023


UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de Ingenieros Informáticos

Doctoral Degree in Software, Systems and Computing

A Uniform Approach to Language Containment Problems

D O C T O R A L T H E S I S

Submitted for the degree of Doctor by:

Kyveli Doveri
Master Degree in Mathematics

Under the supervision of:
Dr. Pierre Ganty

Madrid, 2023

Title: A Uniform Approach to Language Containment Problems

Author: Kyveli Doveri

Doctoral Programme: Doctorado en Software, Sistemas y Computación

Thesis Supervision:

Dr. Pierre Ganty, Associate Research Professor, IMDEA Software Institute (Supervisor)

External Reviewers:

Thesis Defense Committee:

Thesis Defense Date:

This work has been partially supported by PRODIGY Project (TED2021-132464B-I00) funded by MCIN/AEI/10.13039/501100011033/ and the European Union NextGenerationEU/ PRTR

Acknowledgment

First and foremost, I would like to thank my advisor, Pierre Ganty. I am grateful for his guidance and assistance, for his soothing calmness, and most of all for always having time for me – no matter how busy he was I still felt heeded. His help was invaluable and I cannot be more thankful for having such a dedicated advisor.

I would like to thank my co-authors, Francesco Parolini, Francesco Ranzato, Nicolas Mazzocchi, Luka Hadži-Đokić and B Srivathsan who directly contributed to this dissertation and I feel absolutely fortunate working with them.

I am grateful to all my colleagues for the amazing and warm working environment and in particular to Pedro and Elena for their help in the early stages of my doctorate. I thank my friends Dania, Chana and Niki for all the wonderful moments we shared. A special mention goes to Dimitris for both the enjoyable moments and his invaluable assistance with the administrative aspects of the Ph.D. process. Last but not least I would like to thank my dear Fedor for our endless discussions and for his consistent support.

Abstract

We introduce an algorithmic framework to decide the language inclusion for languages of infinite words. We define algorithms for different decidable cases like the inclusion between (nondeterministic) Büchi automata, a PSPACE-complete problem, and the inclusion between Büchi Visibly Pushdown Automata, an EXPTIME-complete problem. All our algorithms rely on a least fixpoint characterization of the languages, leveraging ultimately periodic words, i.e., infinite words of the form uv^ω, with u as a finite prefix and v as a finite period of an infinite word. They are parameterized by quasiorders, indicating which ultimately periodic words need not be tested as counterexamples to inclusion without compromising completeness.

Our first algorithm for the inclusion between Büchi automata is called BAIT and uses two quasiorders on finite words. BAIT is quite simple: it consists of two least fixpoint computations, one for prefixes and the other for periods, manipulating finite sets of states.

Our second algorithm for the inclusion between Büchi automata is called FORKLIFT. Its novelty is that it uses a family of quasiorders. We introduce FORQs (family of right quasiorders) that we obtain by adapting the notion of a family of right congruences put forward by Maler and Staiger in 1993. FORKLIFT consists of two fixpoints for the prefixes and an unbounded number of fixpoints for the periods. Even though it computes more fixpoints, it scales up better than BAIT and the state-of-the-art on a variety of benchmarks, including benchmarks from program verification and theorem proving for word combinatorics.

Our third algorithm, called ωVPLInc, decides the inclusion between Büchi Visibly Pushdown Automata. It uses a pair of quasiorders to prune the search for counterexamples to inclusion. We also implemented our algorithm and conducted an empirical evaluation on benchmarks from software verification.

Additionally, in this thesis we establish a Myhill- Nerode like theorem for specific subclasses of timed languages accepted by one-clock timed automata. The well-known Nerode equivalence for finite words plays a fundamental role in our understanding of the class of regular languages. The equivalence leads to a canonical form, which in turn, is the basis of automata learning algorithms. Such an equivalence has been studied for various subclasses of timed languages, for instance, deterministic timed languages, and real-time languages (which are described by a one-clock timed automaton that is reset in every transition). In this work, we consider the subclass of timed automata with integer resets. This class is known to have good automata-theoretic properties and is also useful for practical modeling. We present a Nerode-style equivalence for this class that depends on a constant K and leads to the construction of a canonical one-clock integer-reset timed automaton with maximum constant K.

Resumen

Introducimos un marco algorítmico para decidir la inclusión de lenguajes de palabras infinitas. Definimos algoritmos para diferentes casos decidibles como la inclusión entre autómatas (no deterministas) Büchi, un problema PSpace-completo, y la inclusión entre autómatas Büchi Visibly Pushdown, un problema EXPTime-completo. Todos nuestros algoritmos se basan en una caracterización de punto fijo mínimo de los lenguajes, aprovechando palabras periódicas en última instancia, es decir, palabras infinitas de la forma uv^ω, con u como prefijo finito y v como período finito de una palabra infinita. Están parametrizados por cuasi-órdenes, indicando qué palabras periódicas en última instancia no necesitan ser probadas como contraejemplos a la inclusión sin comprometer la completitud.

Nuestro primer algoritmo para la inclusión entre autómatas Büchi se denomina BAIT y utiliza dos cuasiórdenes sobre palabras finitas. BAIT es bastante sencillo: consiste en dos cálculos de punto fijo mínimo, uno para prefijos y otro para períodos, manipulando conjuntos finitos de estados.

Nuestro segundo algoritmo para la inclusión entre autómatas Büchi se llama FORKLIFT. Su novedad es que utiliza una familia de cuasiórdenes. Introducimos FORQs (family of right quasiorders) que obtenemos adaptando la noción de familia de congruencias derechas propuesta por Maler y Staiger en 1993. FORKLIFT consta de dos puntos fijos para los prefijos y un número ilimitado de puntos fijos para los períodos. A pesar de que calcula más puntos fijos, se adapta mejor que BAIT y el estado de la técnica en una variedad de puntos de referencia, incluyendo puntos de referencia de verificación de programas y demostración de teoremas para la combinatoria de palabras.

Nuestro tercer algoritmo, llamado ωVPLInc, decide la inclusión entre autómatas Büchi Visibly Pushdown. Utiliza un par de cuasi-órdenes para podar la búsqueda de contraejemplos a la inclusión. También implementamos nuestro algoritmo y realizamos una evaluación empírica en puntos de referencia de verificación de software.

Además, en esta tesis establecemos un teorema similar al de Myhill- Nerode para subclases específicas de lenguajes temporizados aceptados por autómatas temporizados de un reloj. La conocida equivalencia de Nerode para palabras finitas desempeña un papel fundamental en nuestra comprensión de la clase de lenguajes regulares. La equivalencia conduce a una forma canónica que, a su vez, es la base de los algoritmos de aprendizaje de autómatas. Dicha equivalencia se ha estudiado para varias subclases de lenguajes temporizados, por ejemplo, los lenguajes temporizados deterministas y los lenguajes en tiempo real (que se describen mediante un autómata temporizado de un reloj que se reinicia en cada transición). En este trabajo, consideramos la subclase de autómatas temporizados con reinicios enteros. Se sabe que esta clase tiene buenas propiedades autómatas-teóricas y también es útil para el modelado práctico. Presentamos una equivalencia al estilo de Nerode para esta clase que depende de una constante K y conduce a la construcción de un autómata temporizado canónico de un reloj con restablecimiento entero y constante máxima K.

Table of Contents

List of Figures

Abbreviations and acronyms

UPM	Universidad Politécnica de Madrid
FA	Finite Automaton
BA	Büchi Automaton
PDA	Pushdown Automaton
BPDA	Büchi Pushdown Automaton
CFG	Context-Free Grammar
VPA	Visibly Pushdown Automaton
1**-TA**	One-clock Timed Automaton
1**-IRTA**	One-clock Integer Reset Timed Automaton
1**-IRDTA**	One-clock Integer Reset Deterministic Timed Automaton
FORQ	Family of Right Quasiorders

Chapter 1

INTRODUCTION

In this thesis we focus on the language inclusion problem, a fundamental problem in computer science [48] with diverse applications ranging from automata-based verification to compiler construction [51, 34, 53, 81] and theorem proving [68]. Additionally, as a secondary result, we establish a Myhill-Nerode theorem for certain subclasses of timed languages [4]. Here, a language is a collection of words, where a word is simply a sequence of letters from a finite set called an alphabet. For instance, 01001 is a word over the alphabet $\{0, 1\}$ consisting of the two letters 0 and 1. The kind of languages we are interested in are the *formal languages*.

1.1 Formal Languages

Formal languages are possibly infinite sets of words that admit a *finite description* i.e., a finite set of rules that dictate how letters can be combined to form valid words. An example of such finite descriptions are *finite automata* which represent *regular languages*. Two examples of finite automata are given in Figure 1.1.

Figure 1.1: Automata $\mathcal{E}$ and $\mathcal{O}$ with alphabet $\Sigma = \{0, 1\}$.

A finite automaton has a finite number of states and transitions between these states. It accepts or rejects words as follows. It starts in the initial state and reads letters from an input word one at a time. Based on the current state and the letter it reads, the automaton transitions to another state. The process continues until the entire input word is read. If, after reading the entire input word, the automaton is in one of its designated accepting states, it accepts the word as part of the regular language. If not, it rejects the word.

Example 1. *The automaton $\mathcal{E}$ depicted in Figure 1.1 has two states: the initial state q_I and the state q which is accepting. It transitions from state q_I to state q when it reads the letter* 0*. At state q it can loop by reading either* 0 *or* 1*. It accepts the words* 0 *and* 0101 *but rejects*

the word 1*. The language accepted by this automaton consists of all the binary words that represent even positive integers, with least significant digit first. Similarly, the automaton* $\mathcal{O}$ *in Figure 1.1 accepts binary words corresponding to odd positive integers.*

Another class of formal languages are the *context free languages* which extend the expressiveness of finite automata by adding a stack for memory. Such automata are called *pushdown automata*. Figure 1.2 illustrates one. Like a finite automaton, a pushdown automaton has a finite set of

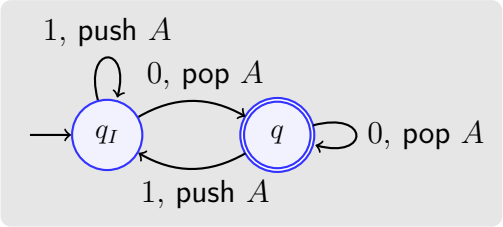


Figure 1.2: Automaton equipped with a stack, with alphabet $\Sigma = \{0, 1\}$ and stack alphabet $\{A\}$.

states, including an initial state and one or more accepting states. Additionally, a pushdown automaton has a stack alphabet, a finite set of symbols that can be pushed onto the stack. It can perform three primary operations on the stack: It can push a symbol onto the top of the stack. It can remove the symbol from the top of the stack. It can choose not to modify the stack. When a pushdown automaton reads an input letter, its behavior depends on both its current state and the symbol at the top of the stack. This combination dictates the next state, the input letter to be consumed, and any stack operations.

Example 2. *The pushdown automaton of Figure 1.2 has two states: the initial state* q_I *and the accepting state* q*. It starts at the initial state* q_I *with an empty stack. It reads the letter* 1*, pushes the symbol* A *onto the stack and remains in state* q_I*. At this point, it has the option to either loop by reading another* 1 *or transition to state* q *by reading* 0 *and popping* A *from the stack. However, the transition to state* q *can only occur if the automaton has previously read a letter* 1 *to ensure an* A *on top of the stack. Subsequently, from state* q*, it can return to state* q_I *by reading* 1*, pushing* A *onto the stack, and repeating the process. As a result, the automaton accepts the word* 1010*. However, it does not accept the word* 000*.*

The stack allows to recognize more complex languages. While it adds expressive power, it also introduces challenges. For example, although the inclusion between regular languages is a decidable problem, the inclusion problem between context-free languages is undecidable. Notably, the inclusion of a context-free language into a regular language remains decidable.

1.2 The Language Inclusion Problem

The language inclusion problem asks whether a language contains all the words of another language. Formally, given a language L and a language M, the problem asks whether every word $w \in L$ is also contained in M. It can be expressed as:

$$L \subseteq^{?} M$$

Solving the inclusion problem efficiently is crucial as it has various applications in computer science. Questions like asking whether a system complies with a specification naturally reduce to a language inclusion problem and so does proving a theorem of the form $\forall x\, \exists y,\ P(x) \Rightarrow Q(y)$ [68] as illustrated by the following example.

Example 3. *Consider the theorem "Every positive integer is either odd or even". The set of positive integers is represented by the language* $\{0,1\}^*$ *while the subsets of even integers and odd integers are given by the languages* $L(\mathcal{E})$ *and* $L(\mathcal{O})$ *of the automata in Figure 1.1. Hence, proving this theorem reduces to checking that the inclusion problem* $\{0,1\}^* \subseteq L(\mathcal{E}) \cup L(\mathcal{O})$ *holds.*

We distinguish two general approaches to solve the language inclusion problem $L \subseteq^? M$: (i) complement M, intersect with L and check for emptiness of the result; and (ii) reduce the inclusion check to finitely many *membership queries* asking whether $w \in M$ holds where $w \in L$ and each query aims at finding a counterexample to inclusion.

In this thesis we focus on the second approach. Previous works in that space leverage relations between words to select a finite subset of words in L for which membership queries are executed. Such relations are equivalence relations and more generally, *quasiorders*.

1.2.1 Quasiorders for the Inclusion

Quasiorders on words are binary relations that satisfy the properties of reflexivity and transitivity, but not necessarily antisymmetry (as opposed to equivalence relations). They are a versatile heuristic that has been applied to inclusion problems for languages such as languages of finite words [2, 24, 31] (including visibly pushdown languages [15]) or languages of infinite words [1, 3, 28, 29, 36, 64] and even tree languages [2, 13]. Quasiorders constitute a class of relations that yields good results in practice. Algorithms leveraging quasiorders are commonly referred to as *antichains algorithms* [24].

The key idea in using quasiorders is to discard words subsumed (for the quasiorder) by others, so as to end up with a finite subset S of L. We then perform the membership queries $w \in M$ for every $w \in S$. To provide a concrete example, consider a quasiorder that compares words based on their length. A word u would be subsumed by a word v if the length of v is less than or equal to the length of u. In this case, the resulting subset S would be the set of words with the smallest lengths.

However, not every quasiorder is suitable. The quasiorder we use, in addition to being decidable, needs to satisfy three requirements. Firstly, it must be a *well-quasiorder*, ensuring the selection of S as a finite subset of minimal words. The second requirement is *M-preservation* which intuitively says that a word inside M cannot subsume a word outside of M. This condition ensures that a subset of minimal words will include a counterexample to inclusion if one exists, so that $L \subseteq M$ holds iff $S \subseteq M$ holds. Furthermore, the quasiorder needs to satisfy certain *monotonicity* conditions w.r.t. word concatenation, which are needed for the computation of S. Notably, these conditions depend on the class of the language L. For example when L is a regular language a quasiorder $\leqslant$ on finite words should be *right-monotonic*, meaning that $u \leqslant v$ implies $ua \leqslant va$ for all letters a and all finite words u and v. On the other hand, when

L is context-free, stronger monotonicity conditions are required. In this case, the quasiorder should be both right-monotonic and left-monotonic.

Example 4 illustrates a suitable quasiorder to be used for the inclusion of a regular language into the language of the automaton of Figure 1.3. Example 5 demonstrates the selection of a subset S of $\{0,1\}^*$ using this quasiorder.

Example 4. *A M-preserving, right-monotonic, decidable well-quasiorder can be defined from an automaton representing the language M. For instance, say that M is given by the automaton of Figure 1.3. We define a quasiorder by comparing two words based on the states reachable by the words from the initial state of the automaton. For instance, the word 1 reaches states q_I and p, while the word 01 can only reach the state q_I. Since the set of reachable states by 01 is a subset of the set of reachable states by 1, 1 is subsumed by 01. More generally, one can show that every word in $\{0,1\}^*$ ending with a 1 is subsumed by 01. Similarly, every word ending with a 0 is subsumed by 0. As prescribed by the M-preservation, all the words ending by 1 are rejected by the automaton, while those ending by 0 are accepted.*

Example 5. *Consider the inclusion problem $\{0,1\}^* \subseteq^? M$ where M is given by the automaton of Figure 1.3. A possible subset S of $\{0,1\}^*$ selected with the quasiorder defined in Example 4 is the set $\{0, 01\}$. This is because any binary word is subsumed by either 0 or 01. By checking membership in M on every word of S we find the counterexample $01 \notin M$ showing that the inclusion $\{0,1\}^* \subseteq M$ does not hold.*

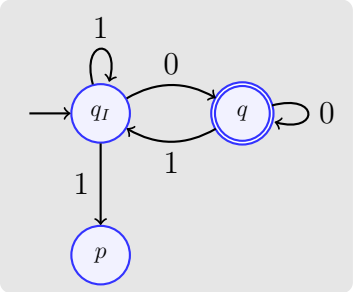


Figure 1.3: Automaton with alphabet $\Sigma = \{0,1\}$.

1.2.2 Starting Point: A Quasiorder-Based Framework

Our starting point is the framework by Ganty et al. [40] which tackles the inclusion problem $L \subseteq^? M$ for languages of words of finite length (finite words). The framework by Ganty et al. handles different decidable cases including the inclusion between regular languages, the inclusion of a context-free language into a regular language and even the inclusion between one-counter net languages.

The algorithm by Ganty et al. uses the so-called antichain heuristics to solve the problem $L \subseteq^? M$ where L is regular or context-free and M is regular. It computes, through a fixpoint computation, a finite subset of words of L. Subsequently, it executes the membership queries $w \in M$ for each word w of this subset. At each step of the fixpoint computation the algorithm discards words using a quasiorder: when two potential counterexamples are comparable

based on the quasiorder, the algorithm can safely discard the 'higher' of the two without compromising the completeness of the search.

As discussed in Section 1.2.1, the chosen quasiorder must satisfy specific conditions for the algorithm to effectively decide the inclusion problem $L \subseteq^? M$. Specifically, it must (1) be decidable; (2) be a well-quasiorder; (3) be M-preserving and (4) satisfy certain monotonicity conditions depending on the class of the language L.

1.2.3 Extending the Framework

In this thesis we want to extend the framework by Ganty et al. to languages of infinite words, often referred to as ω-languages, which consist of words of infinite length. Moreover, we want to handle the inclusion between visibly pushdown languages [5]. More precisely, we focus on the inclusion between two ω-regular languages, the inclusion of a ω-context free language into a ω-regular language and the inclusion between two ω-visibly pushdown languages.

Regular languages of infinite words, or ω-regular languages, are recognized by *Büchi automata*. A Büchi automaton is essentially a finite automaton, but with an altered acceptance condition. This acceptance condition, known as the *Büchi acceptance condition*, is designed to accept or reject infinite words as follows: If the automaton visits certain accepting states infinitely often, then it accepts the infinite word; otherwise, it rejects it. Similarly, ω-context free languages are accepted by *Büchi pushdown automata*.

Example 6. *For example, the automaton in Figure 1.3, seen as a Büchi automaton accepts the infinite word* $0000\ldots$ *consisting only of the letter* 0 *because it can read this word by staying on the accepting state q forever. It does not accept the infinite word* $0111\ldots$ *because it visits the accepting state only once (after reading the letter* 0*), thus finitely many times.*

Visibly pushdown languages extend regular languages by incorporating a stack into the automaton. However, unlike for context-free languages this stack is subject to specific restrictions ensuring decidability properties like solving the inclusion problem for visibly pushdown languages. These restrictions are imposed by partitioning the alphabet into different classes where each class determines the stack operations allowed for letters belonging to that class. To better understand this concept, let's consider a partition of the alphabet $\{0, 1\}$ into two classes: one for the letter 0 and the other for the letter 1. We will specify the stack operations for each class as follows: When processing a letter 0, a pop operation is performed on the stack, while push operations are not allowed. Conversely, when processing a letter 1, no pop operation is permitted, and in this case, a push operation must be executed on the stack. This partition and its associated stack operations represent a simplified example of how visibly pushdown languages operate under alphabet constraints. With this partition of the alphabet, the pushdown automaton of Figure 1.2 falls into the category of visibly pushdown automata (VPA).

Visibly pushdown languages [5] of infinite words are recognized by visibly pushdown automata with a Büchi acceptance condition as illustrated by Example 7 bellow.

Example 7. *The Büchi VPA of Figure 1.2 accepts the infinite word* $1010\ldots$*. It processes this word by alternating between popping and pushing A while transitioning between states* q_I *and*

q. As a result, it passes infinitely often by the accepting state q.

Extending the framework by Ganty et al. [40] to ω-languages and to visibly pushdown languages presents notable challenges.

Challenge 1. *To tackle these inclusion problems, we need a quasiorder that can effectively compare infinite words.* Such a quasiorder is not readily available in the literature. This raises the question of whether one exists at all.

Moreover, for a quasiorder to be suitable for deciding the inclusion problem, it must adhere to conditions similar to the conditions (1)–(4) in the finite word case. The challenge posed by the visibly pushdown languages is further complicated since, even in the context of finite words, it is not immediately evident whether a suitable quasiorder exists. This brings us to our second challenge.

Challenge 2. *Is there a suitable quasiorder for visibly pushdown languages?* Consider the quasiorder described in Example 4, which compares words based on the reachable states in the automaton. This quasiorder can be defined in the VPA case in a similar manner and is a natural starting point in our quest for a quasiorder in the visibly pushdown case. It fulfills the requirements (1)–(3) but it falls short when it comes to satisfying the right-monotonicity condition. To see this, consider the words 110 and 10 and the VPA of Figure 1.2. Both of these words lead the VPA to the same reachable state, q. Therefore, 110 subsumes 10. However, there is a significant distinction in how the VPA processes these two words: With the word 110, the VPA reaches state q with the symbol A on top of its stack, while it has an empty stack with the word 10. As a result, after reading 110, the VPA can transition by reading the letter 0. This transition is not possible with the word 10. Consequently, when we concatenate the letter 0 to the right of these two words, the quasiorder fails to preserve the order. Specifically, 1100 can reach state q, while 100 reaches no state. This suggests that somehow the stack should be taken into account in the definition of our order. However, this jeopardize its well-quasiorder property and raises the question of whether there exists a quasiorder that satisfies all the requirements in the visibly pushdown case.

Perhaps we should question whether right-monotonicity is an appropriate monotonicity condition to require in the visibly pushdown case.

Challenge 3. *What form of monotonicity characterizes visibly pushdown languages?* While right-monotonicity characterizes regular languages (in essence, each transition in a finite automaton corresponds to a right concatenation), this is not the case for visibly pushdown languages. Recall the example involving the two words, 110 and 10, and the VPA in Figure 1.2. As we discussed earlier the VPA handles these two words differently when it comes to concatenating by a letter 0 on the right. This illustrates that the right-monotonicity, which appropriately characterizes regular languages, does not hold the same for visibly pushdown languages.

In summary, our main challenges so far involve finding decidable, M-preserving, monotonic, well-quasiorders for language inclusion problems with infinite words. The specific monotonicity

properties required depend on the language class of L. While right-monotonicity is expected for ω-regular languages, this isn't the case for ω-visibly pushdown languages, which add an extra layer of complexity to our task.

The next challenge apply similarly to all inclusion problems we address.

Challenge 4. *The finite word case crucially relies on least fixpoint characterizations of languages.* Such characterizations are not readily available for languages of infinite words. Therefore, we need to define a fixpoint characterization for ω-regular, ω-context free and ω-visibly pushdown languages.

Challenge 5. *Different quasiorders yield different algorithms.* There are various quasiorders that meet the conditions (1)–(4), offering flexibility within the framework [40]. For instance, quasiorders can be derived from automata representations of M, as explained in Example 4, and enhanced with simulation relations, as demonstrated in [40]. It is also possible to define *syntactic* quasiorders whose definitions are independent of any automaton representation of M. We want to investigate different quasiorders for the inclusion problems we are addressing and assess their trade-offs.

1.3 A Myhill-Nerode Theorem for Timed Languages

A cornerstone in our understanding of regular languages is the Myhill-Nerode theorem. This theorem characterizes regular languages in terms of the Nerode equivalence $\sim_L$: for a word w we write $w^{-1}L = \{z \mid wz \in L\}$ for the *residual language* of w w.r.t. L; and for two words u, v we say $u \sim_L v$ if $u^{-1}L = v^{-1}L$.

Theorem 1 (Myhill-Nerode theorem)**.** *Let L be a language of finite words.*

- *L is regular iff the Nerode equivalence has a finite index.*
- *The Nerode equivalence is coarser than any right-monotonic L-preserving equivalence.*

An equivalence over words being right-monotonic makes it possible to construct a natural automaton with states being the equivalence classes. The Nerode equivalence being the coarsest makes the associated automaton the minimal (and a canonical) deterministic machine for the regular language. Our goal in this work is to obtain a similar characterization for certain subclasses of timed languages.

Timed languages and timed automata were introduced by Alur and Dill [4] as a model for systems with real-time constraints between actions. Ever since its inception, the model has been extensively studied for its theoretical aspects [76, 11, 58, 75, 14, 8] and practical applications [16, 49]. In this setting, words are decorated with a delay between consecutive letters. A *timed word* is a finite sequence $(t_1 \cdot a_1)(t_2 \cdot a_2)\dots(t_n \cdot a_n)$ where each $t_i \in \mathbb{R}_{\geq 0}$ and each a_i is a letter taken from a finite alphabet Σ. A timed word associates a time delay between letters: a_1 was seen after a delay of t_1 from the start, the next letter a_2 appears t_2 time units after a_1, and so on. Naturally, a timed language is a set of timed words. A timed automaton is an automaton model that recognizes timed languages. Figures 1.4 and

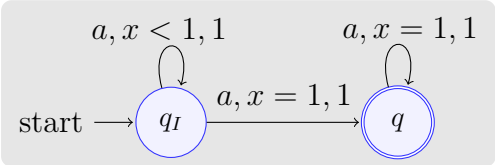


Figure 1.4: Timed Automaton accepting $L_1 = \{(t_1 \cdot a) \dots (t_n \cdot a) \mid t_1 + \dots + t_n = 1\}$ with alphabet $\Sigma = \{a\}$.

1.5 present some examples. In essence, a timed automaton makes use of *clocks* to constrain time between the occurrence of transitions. In Figure 1.4, the variable x denotes a clock. The transition labels are given by triples comprising a letter (e.g. a), a clock constraint (e.g. $x = 1$), and a multiplicative factor (0 or 1) for the clock update. Intuitively, the semantics of the transition from q_I to q is as follows: the automaton reads the letter a when the value of the clock held in x is exactly 1 and updates the clock value to $1 \times x$. Imagining the third element of transition label is 0 then the transition updates the value of x to $0 \times x = 0$. The second element of the transition label is often referred to as the transition *guard* whereas the third element is the *reset*. It is worth mentioning that guards feature constants given by integer values meaning that a guard like $x = 0.33$ is not allowed.

It is challenging to attempt a Nerode-style equivalence for timed languages.

Challenge 1. *The Nerode equivalence lifted as it is has infinitely many classes.* For example, the timed automaton of Figure 1.4 accepts a timed word $(t_1 \cdot a) \dots (t_n \cdot a)$ as long as $t_1 + \dots + t_n = 1$. The timed language L_1 of that automaton has infinitely many quotients. Indeed let $0 < t_1 < 1$, we have that $(t_1 \cdot a)^{-1} L_1 = \{(t_2 \cdot a) \dots (t_n \cdot a) \mid t_2 + \dots + t_n = 1 - t_1\}$. It is easy to see that different values for t_1 yield different quotients, hence L_1 has uncountably many quotients.

Challenge 2. *Two words with the same residual languages may never go to the same control state in any timed automaton.* Figure 1.5 gives an example of a timed language that exhibits this challenge. Consider the words $u = (0.5 \cdot a)$ and $v = (1.5 \cdot a)$. The residual of both these words is the singleton language $\{(0.5 \cdot b)\}$. Suppose both u and v go to the same control state q in the timed automaton. After reading u (resp. v), clocks which are possibly reset will be 0, whereas the others will be 0.5 (resp. 1.5). Suppose v is accepted via a transition sequence $q_I \to q \to q_F$. Since guards contain only integer constants, the guard on $q \to q_F$ should necessarily be of the form $x = 2$ for some clock x which reaches q with value 1.5. The same transition can then be taken from u to give $u(2 \cdot b)$ or $u(1.5 \cdot b)$ depending on the value of x after reading u. A contradiction. This challenge simply means that we cannot hope to identify states of a timed automaton through quotients of a Nerode-type equivalence. The equivalence that we are aiming for needs to be stronger, and further divide words based on some past history.

Challenge 3. *The Nerode-style equivalence should be amenable to a timed automaton construction.* In the case of languages of untimed words, monotonicity of the Nerode equiv-

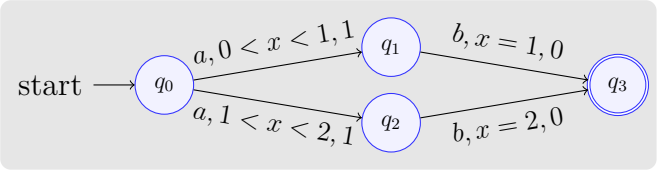


Figure 1.5: Automaton accepting $L_2 = \{(t_1 \cdot a)(t_2 \cdot b) \mid \text{either } 0 < t_1 < 1 \text{ and } t_1 + t_2 = 1, \text{ or } 1 < t_1 < 2 \text{ and } t_1 + t_2 = 2\}$ with alphabet $\Sigma = \{a, b\}$

alence immediately led to an automaton construction. We need to find the right notion of monotonicity for the class of automata that we want to build from the equivalence.

1.4 Thesis Contributions

Next, we outline our contributions in the same order as they are presented in this thesis.

1.4.1 Inclusion for Infinite Words

We define antichain algorithms for the inclusion problem $L \subseteq^? M$ where L and M are two regular languages of infinite words [29], a PSPACE-complete problem [54]. To do so, we build upon the quasiorder-based approach. We further demonstrate the generality of our algorithmic framework by instantiating it to the inclusion problem of ω-context free languages into ω-regular languages, which is known to be EXPTIME-complete [50, 63].

To overcome the first challenge outlined in Section 1.2, we reduce the inclusion problem between ω-regular languages into an equivalent inclusion problem between their *ultimately periodic words*. These are infinite words of the form $uv^\omega \in L$, where u is a prefix and v a period of uv^ω respectively. Once this reduction is accomplished, we compare two ultimately periodic words using a pair of quasiorders on finite words, denoted as $\leqslant, \preccurlyeq$. The first quasiorder, $\leqslant$, compares the prefixes of ultimately periodic words, while the second quasiorder, $\preccurlyeq$, compares their periods. In other words, we compare the decompositions (u, v) of ultimately periodic words uv^ω using the ordering $\leqslant \times \preccurlyeq$ to filter out a finite number of them.

As expected, for the inclusion between ω-regular languages both quasiorders $\leqslant$ and $\preccurlyeq$ need to be right-monotonic. Furthermore, it turns out that a Büchi automaton enables a fixpoint characterization of the set of ultimately periodic words it accepts. Hence, by adapting certain quasiorders from the literature we are able to obtain a suitable pair of quasiorders for the inclusion between ω-regular languages. Consequently, we are able to compute a finite subset of ultimately periodic words, which is sufficient for solving the inclusion problem.

We propose different pairs of quasiorders to be used in our framework and discuss how they compare. Of particular interest are the *state-based* pairs of quasiorders derived from a Büchi automaton because they allow to formulate an inclusion algorithm that exclusively operates on automata states. We call such an algorithm a *state-based algorithm*. We also define a syntactic pair whose definition is based solely on M. This syntactic pair of orders is "ideal" in the sense it is the coarsest one suitable to apply in our framework. Hence, it selects the

"smallest" subset of L. However, the quasiorders of the syntactic pair are as hard to decide as the language inclusion problem itself.

We implemented our state-based algorithm in a tool called BAIT (Büchi Automata Inclusion Tester) [9]. We conducted an experimental comparison of BAIT against some state-of-the-art language inclusion tools. We also compare with the FORQ-based approach [28] of the next contribution.

1.4.2 Improved Inclusion with Families of Quasiorders

Our second contribution is also an antichain algorithm for the inclusion problem between regular languages of infinite words. The novelty here is that instead of two quasiorders (as in the previous contribution), we use an unbounded number of quasiorders: one for the prefixes and a family of quasiorders for the periods each of them depending on a distinct prefix. Our motivation for doing so is to obtain more efficient algorithms. Resorting to a family of quasiorders, instead of just a pair, allows more pruning when searching for a counterexample, thus lesser membership queries at the end.

We formalize the notion of family of right quasiorders by relaxing and generalizing the notion of *family of right congruences* introduced by Maler and Staiger [61]. Using families of quasiorders leads to significant algorithmic differences compared to the two quasiorders approach [29]. More precisely, the algorithm with two quasiorders computes exactly two fixpoints (one for the prefixes and one for the periods) independently whereas the algorithm using a family of quasiorders computes two fixpoints for the prefixes and unboundedly many fixpoints for the periods (depending on the number of prefixes that belong to the first two fixpoints). Even though we lose the prefix/period independence and we compute more fixpoints, in practice, the use of families of quasiorders scales up better than the approach based on two quasiorders.

We define the FORQ of a Büchi automaton and study the algorithmic complexity of our algorithm instantiated with this specific FORQ.

Finally, we implemented our FORQ-based algorithm in a tool called FORKLIFT [28]. We conducted an experimental evaluation of both FORKLIFT and BAIT, comparing them with state-of-the-art language inclusion checking tools, including SPOT [33, 32], GOAL [77], RABIT [71, 18] and ROLL [56]. We conducted our experimental evaluation on an extensive benchmark suite. This suite covers verification tasks as defined by the RABIT tool [1, 3], logical implication tasks in word combinatorics as defined by the Pecan theorem prover [67], and termination tasks as defined by Ultimate Automizer [44, 43].

1.4.3 Inclusion for Visibly Pushdown Languages

We introduce antichain algorithms for the inclusion problem $L \subseteq^? M$ between visibly pushdown languages of infinite words [26], an EXPTIME-complete problem.

In this context as well, the selection of words from L is limited to the ultimately periodic words. We define a specific type of decompositions of ultimately periodic words called *legitimate decompositions*. By reducing the problem to these legitimate decompositions, we ensure that we only compare words that a VPA handles in a similar manner. We also define

monotonicity conditions w.r.t. word concatenations that are characteristic to visibly pushdown languages. We establish a fixpoint characterization for a subset of legitimate decompositions of L. Hence, we are able to define an antichain algorithm which leverages a subset of legitimate decompositions, and is parameterized by a pair of quasiorders.

We put forward different quasiorders to be used in our algorithm: state-based quasiorders derived from a visibly pushdown automaton underlying M and, also syntactic quasiorders. We further prove that when instantiated with the state-based quasiorders the resulting state-based algorithm, has a runtime that matches the corresponding problem complexity.

Finally, we implemented the state-based algorithm and evaluated it on various benchmarks collected from Friedmann et al. [38] and from SV-COMP[1], the Software Verification competition. Our empirical evaluation was carried out against Ultimate [44] which follows a complement, intersect and check for emptiness approach. The preliminary conclusion of the empirical results is in favor of our approach as it scales up better.

1.4.4 A Myhill-Nerode Theorem for Timed Automata with Integer Resets

In this work, we look at languages recognized by timed automata with integer resets (IRTA) [76]. These are automata where clock resets are restricted to transitions that contain a guard of the form $x = c$ for some clock x and some integer c. It is known that IRTA can be reduced to 1-clock-deterministic IRTA [62].

We define a notion of monotonicity for an equivalence on timed words. This monotonicity is parametrised by a constant K, and is called K-monotonicity. Given an IRTA language L we define a Nerode-style equivalence $\approx^{L,K}$ that has a finite number of classes and is K-monotonic. This equivalence leads to the following Myhill-Nerode style characterization for IRTA languages.

Theorem 2. *Let L be a timed language.*

- *L is accepted by a timed automaton with integer resets iff there exists a constant K such that $\approx^{L,K}$ is K-monotonic and has a finite index.*
- *The $\approx^{L,K}$ equivalence is coarser than any K-monotonic L-preserving equivalence.*

This characterization leads to the construction of a canonical 1-clock-deterministic IRTA with maximum constant K.

Finally, we also present a similar characterization for the class of languages obtained by timed automata with no resets.

[1] https://sv-comp.sosy-lab.org

Chapter 2

RELATED WORKS

In this section, we present existing work related to this thesis.

Inclusion for Regular Languages. Significant effort has been devoted to the discovery of algorithms for inclusion that behave well in practice [3, 18, 30, 36, 52, 55]. Each proposed algorithm is characterized by a set of techniques (e.g. Ramsey-based, rank-based) and heuristics (e.g. antichains, simulation relations). The algorithm we propose falls into the category of Ramsey-based algorithms and uses the antichain [24] heuristics.

In the work of Abdulla et al. [1, 3] which was further refined by Clemente et al. [18] they use a single quasiorder to compare both prefixes and periods. Their effort has been focused on refining that single quasiorder by enhancing it with simulation relations.

Kuperberg et al. [52] also reduce the language equivalence problem over Büchi automata to that of their ultimately periodic subsets. A further commonality is that the algorithm of [52] handles prefixes and periods differently: for the prefixes they leverage a state-of-the-art up-to congruence algorithm [12], while up-to congruences are not used for the periods[1]. Fogarty and Vardi [36] for the universality problem, and later Abdulla et al. [1, 3] for the inclusion problem between languages accepted by Büchi automata, all reduce their decision problems to the ultimately periodic subsets. Their approach is based on a partition of nonempty words whose blocks are represented and manipulated through so-called supergraphs. The equivalence relation underlying their partition can be obtained from one of our quasiorders. Moreover, by equipping their supergraphs with a subsumption order [3, Def. 6], they define a relation which coincides with one of our quasiorders. Hofmann and Chen [47], whose approach based on abstract interpretation inspired our work, also tackle the inclusion problem for ω-languages. They construct an abstract (finite) lattice using the same equivalence relation which is derived from a given Büchi atomaton, and define a Galois connection between it and the (infinite) lattice of languages of infinite words. However, they do not relax this relation into a quasiorder.

Finally, the complementation-based approaches reduce language inclusion to a language

[1] In the technical report thereof, the authors work out up-to union and up-to equivalence reasoning for periods but not their combination (up-to congruence).

emptiness check by using intersection and an explicit complementation of a Büchi automaton. Despite that there are Büchi automata of size n whose complement cannot be represented with less than $n!$ states [65], algorithms to complement Büchi automata have been defined, implemented and are effective in practice [78]. In our approach, explicit complementation is avoided altogether.

Families of Quasiorders. The notion of family of right quasiorders was inspired by the notion of *family of right congruences* (FORC) introduced by Maler and Staiger [61] to advance the theory of recognizability of ω-regular languages and, in particular, questions related to minimal-state automata. Recently, families of right congruences have been used in other contexts like the learning of ω-regular languages (see [7] and references therein) and Büchi automata complementation [57].

Inclusion for Visibly Pushdown Languages. Bruyere et al. [15] proposed an antichain algorithm for the inclusion of VPL but they only tackle the problem for languages of finite words. The same limitation applies to Ganty et al. [40] where, moreover, they do not tackle the inclusion problem of VPL into VPL (the closest they tackle is CFL into regular). The extension from the finite to the infinite case was tackled in Doveri et al. [29] but they do not cover the case ω-VPL into ω-VPL (the closest they tackle is ω-CFL into ω-regular). Friedmann et al. [37, 38] do tackle the ω-VPL into ω-VPL problem. However they do not leverage the full power of quasiorders (they use equivalence instead); they do not use distinct pruning techniques for prefix and periods; and they do not put forward syntactic quasiorders. In the work by Henzinger et al. [45] the authors define antichains algorithms for solving the inclusion problem between operator-precedence languages [35]. These languages, which also enjoy an EXPTime-complete inclusion problem, fall within a class that is strictly contained in deterministic context-free languages and, in turn, strictly contains VPL [22].

Canonical Form for Timed Languages. A machine independent characterization for deterministic timed languages has been studied by Bojańczyk and Lasota [11]. The challenges presented in Section 1.3 are circumvented by considering a new automaton model *timed register automata* that generalizes timed automata. This automaton model makes use of registers to store useful information, for instance for the language in Figure 1.5, a register stores the value 0.5 after reading $(0.5 \cdot a)$ and $(1.5 \cdot a)$. This feature helps in resolving Challenge 2. For the question of finiteness mentioned in Challenge 1, timed register automata are further viewed as a restriction of a more general model of automata that uses the abstract concept of *Frankel-Mostowski* sets in its definition. Finiteness is relaxed to a notion of *orbit-finiteness*.

The work of An et al. [6] takes another approach to these challenges by considering a subclass of timed languages which are called *real-time languages*. These are languages that can be recognized using timed automata with a single clock that is reset in every transition. Therefore, after reading a letter, the value of the clock is always 0. This helps in solving the challenges, resulting in a canonical form for real-time languages.

The work [58] also focus on a machine independent characterization for deterministic timed languages, whereas the works [42], [6], [80] extend the study of the canonical forms to an

active learning algorithm.

Our work is in the same spirit as [11], but for a subclass of deterministic timed languages. Languages accepted by event-recording automata are a class of languages where the value of the clocks is determined by the input word. This helps in coming up with a canonical form [42]. In [80], the author presents a Myhill-Nerode style characterization for deterministic timed languages by making use symbolic words, rather than timed words directly. The author shows that the equivalence has finite index iff the language is recognizable (under the notion of recognizability using right-morphisms from timed words to a bounded subset of itself, given by Maler and Pnueli [58]). Further, Maler and Pnueli have given an algorithm to convert recognizable timed languages to deterministic timed automata which resets a fresh clock in every transition and makes use of clock-copy updates $x := y$ in the transitions. It is known that automata with such updates can be translated to classical timed automata [75, 14].

We have considered a subclass of deterministic timed languages. Therefore, our class does fall under the purview of the [80, 58] work — however, the fundamental difference is that we continue to work with timed words, and not symbolic timed words. This gives an alternate perspective and a direct and simpler 1-clock IRTA construction.

Chapter 3

PRELIMINARIES

In this section, we introduce the concepts and notations that will be used throughout the remainder of the thesis.

3.1 Words and Languages

An *alphabet* is a nonempty finite set of symbols, generally denoted by Σ. A *word* is a finite sequence $a_1 \cdots a_n$ of letters $a_i \in \Sigma$. The set of finite words and the set of infinite words over Σ are denoted by Σ^* and Σ^ω respectively. We denote by ϵ the *empty word* and define $\Sigma^+ \triangleq \Sigma^* \backslash \{\epsilon\}$. An *ultimately periodic word* is an infinite word $\xi \in \Sigma^\omega$ such that $\xi = uv^\omega$ for some finite *prefix* $u \in \Sigma^*$ and some finite *period* $v \in \Sigma^+$. We call such a couple $(u, v) \in \Sigma^* \times \Sigma^+$ a *decomposition* of ξ. For a word $u \in \Sigma^* \cup \Sigma^\omega$ we denote by $|u| \in \mathbb{N} \cup \{\omega\}$ its length and for $i \in \{1, \ldots, |u|\}$ we write $u[i]$ for the i-th letter of u. A *language of finite words* over Σ is a subset of Σ^*. A *language of infinite words* or ω-*language* over Σ is a subset of Σ^ω.

A *timestamp* is a finite sequence of non-negative real numbers. We denote the latter set by $\mathbb{R}_{\geq 0}$ and the set of all timestamps by $\mathbb{T}$. A *timed word* is a finite sequence $(t_1 \cdot a_1) \cdots (t_n \cdot a_n)$ where $a_1 \cdots a_n \in \Sigma^*$ and $t_1 \cdots t_n \in \mathbb{T}$. We denote the set of timed words by $\mathbb{T}\Sigma^*$. Given a timed word $u = (t_1 \cdot a_1) \ldots (t_n \cdot a_n) \in \mathbb{T}\Sigma^*$ we denote by $\sigma(u)$ the sum $t_1 + \cdots + t_n$ of its timestamps. A *timed language* is a set of timed words. As usual, we denote the empty (un)timed word by ϵ. The *residual language* of an (un)timed language L with regard to a (un)timed word u is defined as $u^{-1}L = \{w \mid uw \in L\}$. Therefore it is easy to see that $\epsilon^{-1}L = L$ for every (un)timed language L. Also we have that $\sigma(\epsilon) = 0$.

3.2 Well-Quasiorders, Kleene Iterates and Monotonicity

Well-Quasiorders. A *quasiorder* on a set E, is a binary relation $\ltimes \subseteq E \times E$ that is reflexive ($x \ltimes x$) and transitive ($x \ltimes y \wedge y \ltimes z \implies x \ltimes z$). A quasiorder $\ltimes$ is a *partial order* when $\ltimes$ is antisymetric ($x \ltimes y \wedge y \ltimes x \implies x = y$). Given two subsets $X, Y \subseteq E$ the set Y is

said to be a *basis* for X w.r.t. $\ltimes$ whenever $Y \subseteq X$ and $\forall x \in X, \exists y \in Y, y \ltimes x$. A basis Y for X is said to be a *finite basis* if it is a finite set. A quasiorder is a *well-quasiorder* if every subset of E admits a finite basis. An *antichain* is a subset of E such that any two distinct elements in the subset are incomparable.

Equivalence Relation. A binary relation $\sim \subseteq S \times S$ on a set S is an *equivalence* if it is reflexive (i.e. $x \sim x$), transitive (i.e. $x \sim y \wedge y \sim z \implies x \sim z$) and symmetric (i.e. $x \sim y \implies y \sim x$). The equivalence class of $s \in S$ w.r.t. $\sim$ is the subset $[s]_\sim = \{s' \in S \mid s \sim s'\}$. A *representative* of the class $[s]_\sim$ is any element $s' \in [s]_\sim$. Given a subset D of S we define $[D]_\sim = \{[d]_\sim \mid d \in D\}$. We say that $\sim$ has *finite index* when $[S]_\sim$ is a finite set.

Kleene Iterates. A *complete lattice* is a partially ordered set $(E, \ltimes)$ in which every subset has a least upper bound (the supremum) in E. A sequence $\{s_n\}_{n\in\mathbb{N}} \in E^{\mathbb{N}}$ on quasiordered set $(E, \ltimes)$ is *increasing* if for every $n \in \mathbb{N}$ we have $s_n \ltimes s_{n+1}$. For a function $f\colon E \to E$ on a quasiordered set $(E, \ltimes)$ and for all $n \in \mathbb{N}$, we define the n-th iterate $f^n : E \to E$ of f inductively as follows: $f^0 \triangleq \lambda x.x$; $f^{n+1} \triangleq f \circ f^n$. The denumerable sequence of *Kleene iterates* of f starting from the bottom value $\bot \in E$ is given by $\{f^n(\bot)\}_{n\in\mathbb{N}}$. Recall that when $(E, \ltimes)$ is a complete lattice and $f\colon E \to E$ is an increasing function (i.e. $d \ltimes d' \implies f(d) \ltimes f(d')$) then by the Knaster–Tarski theorem, f has a least fixpoint $\mathrm{lfp}\, f$ given by the supremum of the increasing sequence of Kleene iterates of f i.e., $\mathrm{lfp}\, f = \bigcup_{n\in\mathbb{N}} f^n(\bot)$.

Monotonicity. A quasiorder $\ltimes \subseteq \Sigma^* \times \Sigma^*$ is said to be *right-monotonic* when $\forall u, v \in \Sigma^*, \forall a \in \Sigma, u \ltimes v \Rightarrow ua \ltimes va$. We define symmetrically a quasiorder to be *left-monotonic* when it is preserved by left concatenations. We say that a quasiorder is *monotonic* if it is both left and right-monotonic.

3.3 Finite Automata

A *Finite Automaton* (FA for short) defined over an alphabet Σ is a tuple $\mathcal{A} = (Q, q_I, \delta, F)$, where:

- Q is a finite set of states, including a unique initial state $q_I \in Q$.
- $\delta \subseteq Q \times \Sigma \times Q$ is a finite set of transitions.
- $F \subseteq Q$ is a subset of final states.

We denote transitions as $q \xrightarrow{a} q'$ when $(q, a, q') \in \delta$ and extend this relation to finite words using transitive and reflexive closure, writing $q \xrightarrow{u}{}^* q'$ for $u \in \Sigma^*$. Furthermore, we use $q \xrightarrow{u}{}^{\circledast} q'$ to indicate the existence of $q_f \in F$ and $u_1, u_2 \in \Sigma^*$ such that $q \xrightarrow{u_1}{}^* q_f$, $q_f \xrightarrow{u_2}{}^* q'$, and $u = u_1 u_2$.

A run of $\mathcal{A}$ on a word $u = a_0 a_1 \cdots \in \Sigma^* \cup \Sigma^\omega$ is a sequence:

$$e = q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \cdots .$$

When u is a finite word, e is a finite sequence $q \xrightarrow{u}{}^* q'$. When u is an infinite word, e is an infinite sequence. In this case, e is called an *accepting run* when $q_0 = q_I$ and $q_j \in F$ for infinitely many j's.

The language of finite words accepted by $\mathcal{A}$ is defined as:

$$L^*(\mathcal{A}) \triangleq \bigcup_{q \in F} \{u \in \Sigma^* \mid q_I \xrightarrow{u}{}^* q\}.$$

A language $L \subseteq \Sigma^*$ is *regular* if $L = L^*(\mathcal{A})$ for some FA $\mathcal{A}$.

We refer to $\mathcal{A}$ as a *Büchi Automaton* (BA) when considering it as an acceptor of infinite words. The ω-language accepted by $\mathcal{A}$ is defined as:

$$L^\omega(\mathcal{A}) \triangleq \{\xi \in \Sigma^\omega \mid \text{ there is an accepting run of } \mathcal{A} \text{ on } \xi\}.$$

A language $L \subseteq \Sigma^\omega$ is *ω-regular* if $L = L^\omega(\mathcal{A})$ for some BA $\mathcal{A}$.

We provide two examples of automata in Figure 3.1:

1. FA $\mathcal{C}$ accepts every finite word over the alphabet $\{a, b\}$ and, when viewed as a BA, it accepts every infinite word over the same alphabet.
2. A finite word is accepted by $\mathcal{D}$ if and only if it ends with an a. An infinite word is accepted by $\mathcal{D}$ if and only if it contains infinitely many a's.

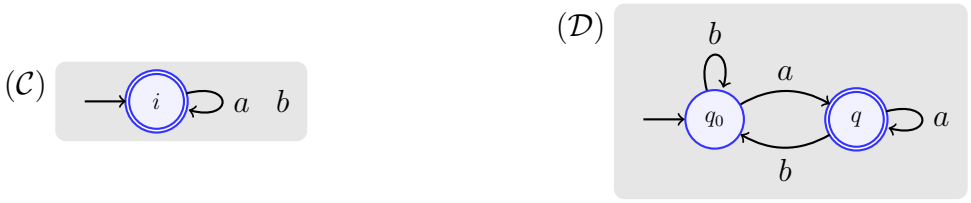


Figure 3.1: Automata $\mathcal{C}$ and $\mathcal{D}$ with alphabet $\Sigma = \{a, b\}$.

3.4 Pushdown Automata

A *Pushdown Automaton* (PDA) over Σ is defined by the tuple $\mathcal{P} = (Q, q_I, \Gamma, \delta, F)$, where:

- Q is a finite set of states, including an initial state $q_I \in Q$.
- Γ is the stack alphabet, including an initial stack symbol $\perp$.
- δ is a finite set of transitions, $\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \times Q \times \Gamma^*$.
- $F \subseteq Q$ is a subset of final states.

Configurations of the PDA $\mathcal{P}$ are pairs in $Q \times \Gamma^*$. For each $a \in \Sigma$, the transition relation $\vdash^a$ between configurations is defined by $(q, \gamma w) \vdash^a (p, \beta w)$, for some $w \in \Gamma^*$, when $(q, a, \gamma, p, \beta) \in \delta$. This relation is extended to words using reflexivity and transitivity. Specifically, for all $u \in \Sigma^*$, $(q, w) \vdash^{*u} (p, w')$ when the configurations (q, w) and (p, w') are related by a sequence of transitions such that the concatenation of their labels forms the word u. We denote $(q, w) \vdash^{\circledast u} (p, w')$ when such a sequence includes a configuration with a final state. The language of finite words accepted by a PDA $\mathcal{P}$ is defined as:

$$L^*(\mathcal{P}) \triangleq \bigcup_{q \in F} \{u \in \Sigma^* \mid (q_I, \perp) \vdash^{*u} (q, w),\ w \in \Gamma^*\}.$$

A language $L \subseteq \Sigma^*$ is *context-free* if $L = L^*(\mathcal{P})$ for some PDA $\mathcal{P}$ on Σ.

A run of $\mathcal{P}$ on an infinite word $\xi = a_0 a_1 \cdots \in \Sigma^\omega$ is an infinite sequence:

$$(q_0, w_0) \vdash^{a_0} (q_1, w_1) \vdash^{a_1} \cdots .$$

It is an *accepting run* when $(q_0, w_0) = (q_I, \bot)$ and $q_j \in F$ for infinitely many j's.

We refer to $\mathcal{P}$ as a *Büchi Pushdown Automaton* (BPDA) when considering it as an acceptor of infinite words. The ω-language accepted by $\mathcal{P}$ is defined as:

$$L^\omega(\mathcal{P}) \triangleq \{\xi \in \Sigma^\omega \mid \text{ there is an accepting run of } \mathcal{P} \text{ for } \xi\}.$$

A language $L \subseteq \Sigma^\omega$ is *ω-context-free* if $L = L^\omega(\mathcal{P})$ for some BPDA $\mathcal{P}$ on Σ.

3.5 Context-free Grammar

Context-free languages can also be generated by grammars. A *Context-Free Grammar* (CFG) on Σ is a tuple $\mathcal{G} = (V, P)$ where $V = \{X_1, \cdots, X_n\}$ is the set of variables including the *start variable* X_1, and P is the set of productions $X_i \to \beta$ where $\beta \in (V \cup \Sigma)^*$. We assume, for simplicity and without loss of generality, that grammars are always given in *Chomsky Normal Form*, that is, every rule $X_j \to \beta \in P$ is such that $\beta \in (V \times V) \cup \Sigma \cup \{\epsilon\}$ and if $\beta = \epsilon$ then $i = 1$. We also assume that for all $X_j \in V$ there exists a rule $X_j \to \beta \in P$, otherwise X_j can be safely removed from V. Given two strings $w, w' \in (V \cup \Sigma)^*$ we write $w \Rightarrow w'$ if there exist two strings $u, v \in (V \cup \Sigma)^*$ and a grammar rule $X \Rightarrow \beta \in P$ such that $w = uXv$ and $w' = u\beta v$. We denote by $\Rightarrow^*$ the reflexive-transitive closure of $\Rightarrow$. The language generated by G is $L^*(G) \triangleq \{w \in \Sigma^* \mid X_1 \Rightarrow^* w\}$.

3.6 Visibly Pushdown Automata

Let $\Sigma = \Sigma_i \cup \Sigma_c \cup \Sigma_r$ be an alphabet comprising three disjoint alphabets. Given a word $u = u_0 u_1 \cdots \in \Sigma^* \cup \Sigma^\omega$, we define a position j (where $j \in \mathbb{N}$ and $j < |u|$) to be an *internal* position if $u_j \in \Sigma_i$, a *call* position if $u_j \in \Sigma_c$, and a *return* position if $u_j \in \Sigma_r$.

A *Visibly Pushdown Automaton* (VPA) over Σ is described by the tuple $\mathcal{A} = (Q, q_I, \Gamma, \delta, F)$. It is akin to a PDA, with the distinction being in the transition relation, defined as: $\delta = \delta_i \cup \delta_c \cup \delta_r$, comprising three transition relations:

- $\delta_i \subseteq Q \times \Sigma_i \times Q$
- $\delta_c \subseteq Q \times \Sigma_c \times Q \times \Gamma \backslash \{\bot\}$
- $\delta_r \subseteq Q \times \Sigma_r \times \Gamma \times Q$

For $a \in \Sigma$, the relation $\vdash^a$ between configurations is defined as follows:

- If $a \in \Sigma_i$ and $w \in \Gamma^*$, $(p, w) \vdash^a (q, w)$ if $(p, a, q) \in \delta_i$.
- If $a \in \Sigma_c$ and $w \in \Gamma^*$, $(p, w) \vdash^a (q, w\gamma)$ if $(p, a, q, \gamma) \in \delta_c$.

- If $a \in \Sigma_r$, $\gamma \in \Gamma \backslash \{\bot\}$, and $w \in \Gamma^*$, $(p, w\gamma) \vdash^a (q, w)$ if $(p, a, \gamma, q) \in \delta_r$.
- If $a \in \Sigma_r$, $(p, \bot) \vdash^a (q, \bot)$ if $(p, a, \bot, q) \in \delta_r$.

We lift the relation $\vdash$ to words by transitivity and reflexivity. Similar to the BPDA, we define the criteria for accepting runs and the ω-language of a VPA. A language $L \subseteq \Sigma^\omega$ is ω-VPL if $L = L^\omega(\mathcal{A})$ for some VPA $\mathcal{A}$.

Two examples of VPA are provided in Figure 3.2, where $\mathcal{A}$ has an accepting run on $c\,cr\,cr\,cr\ldots$ and so does $\mathcal{B}$ on $crr\,crr\ldots$.

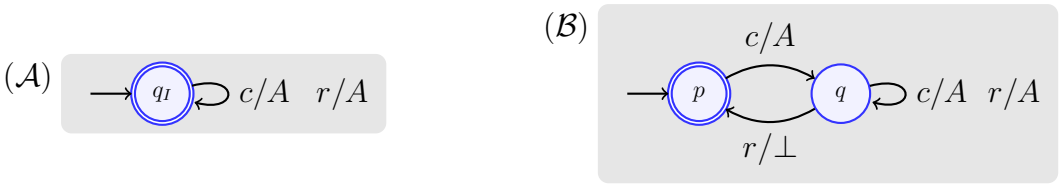


Figure 3.2: Two ω-VPA with $\Gamma = \{A, \bot\}$, $\Sigma_i = \emptyset$, $\Sigma_c = \{c\}$ and $\Sigma_r = \{r\}$.

Matching Relation. The partition of the alphabet $\Sigma = \Sigma_i \cup \Sigma_c \cup \Sigma_r$ induces a unique matching relation between a word's *call* and *return* positions (see [38]).

For any $u \in \Sigma^* \cup \Sigma^\omega$, we define the *matching relation* of u, denoted $\curvearrowright_u$, as the unique relation on its call and return positions such that for every $j \curvearrowright_u k$ we have $0 \leq j < k < |u|$, $u_j \in \Sigma_c$, $u_k \in \Sigma_r$, $|\{n \mid j \curvearrowright_u n\}| \leq 1$, $|\{n \mid n \curvearrowright_u k\}| \leq 1$ and there are no j', k' with $j' \curvearrowright_u k'$ and $j < j' < k < k'$.

When $j \curvearrowright_u k$, we refer to j and k as *matched* positions. A call (resp. return) position j in u is *unmatched* if there exists no k such that $j \curvearrowright_u k$ (resp. $k \curvearrowright_u j$). Furthermore, for every unmatched position n in u there is no $j \curvearrowright_u k$ such that $j < n < k$, and if $u_n \in \Sigma_c$ (resp. $u_n \in \Sigma_r$), then there is no unmatched return (resp. call) position k with $n < k$ (resp. $k < n$). A word is said to be *well-matched* if it has no unmatched position.

Definition 3.6.1. We define the subsets of words $\mathtt{W}$, $\mathtt{C}$, $\mathtt{R}$, and $\mathtt{U_c}$ as the least solution to the following system of equations:

$$
\begin{aligned}
W &= \Sigma_i \cup \{\epsilon\} \cup \Sigma_c\, W\, \Sigma_r \cup W\, W \\
R &= \Sigma_c \cup W \cup R\, R \\
C &= \Sigma_r \cup W \cup C\, C \\
U_c &= \Sigma_c \cup C\, U_c\, R
\end{aligned}
$$

The subset $\mathtt{W}$ is the set of well-matched finite words, $\mathtt{C}$ (resp. $\mathtt{R}$) is the set of finite words where all call (resp. return) positions are matched and $\mathtt{U_c}$ is the set of finite words with at least one unmatched call position.

3.7 Timed Automata

Timed automata are recognizers of timed languages introduced in the seminal work [4]. Since we focus on subclasses of timed automata with a single clock, we will not present the definition of general timed automata. Instead, we give a modified presentation of one clock timed automata that will be convenient for our work.

One Clock Timed Automata. A *One-clock Timed Automaton* (1-TA) over Σ is a tuple $\mathcal{A} = (Q, q_I, T, F)$ where Q is a finite set of states, $q_I \in Q$ is the initial state, $F \subseteq Q$ is the set of final states and $T \subseteq Q \times Q \times \Sigma \times \Phi \times \{0, 1\}$ is a finite set of transitions where Φ is the set of clock constraints given by

$$\phi ::= x < m \quad | \quad m < x \quad | \quad x = m \quad | \quad \phi \wedge \phi \ , \text{ where } m \in \mathbb{N}.$$

For a clock constraint ϕ, we write $[\![\phi]\!]$ for the set of values of x that satisfies the constraint. Notice that we have disallowed guards of the form $x \leq m$ which appear in standard timed automata literature, since its effect can be captured using two transitions, one with $x = m$ and another with $x < m$. In our syntax, a transition looks like (q, q', a, ϕ, r) where ϕ is a clock constraint called the *guard* of the transition and $r \in \{0, 1\}$ denotes whether the single clock x is *reset* in the transition: 0 denotes that it is reset, whereas 1 denotes otherwise.

We say that a 1-TA with transitions T is *deterministic* whenever for every pair $\theta = (q, q', a, \phi, r)$ and $\theta_1 = (q_1, q_1', a_1, \phi_1, r_1)$ of transitions in T such that $\theta \neq \theta_1$ we have that either $q \neq q_1$, $a \neq a_1$ or $[\![\phi]\!] \cap [\![\phi_1]\!] = \emptyset$.

A *run* of $\mathcal{A}$ on a timed word $(t_1 \cdot a_1) \dots (t_k \cdot a_k) \in \mathbb{T}\Sigma^*$ is a finite sequence

$$e = (q_0, \nu_0) \xrightarrow{t_1, \theta_1} (q_1, \nu_1) \xrightarrow{t_2, \theta_2} \cdots \xrightarrow{t_k, \theta_k} (q_k, \nu_k) \ ,$$

where $q_j \in Q$, $\nu_j \in \mathbb{R}_{\geq 0}$ for all $j \in \{0, \dots, k\}$ and, moreover for each $i \in \{1, \dots, k\}$ the following hold:

- $\theta_i \in T$ and θ_i is of the form $(q_{i-1}, q_i, a_i, \phi_i, r_i)$,
- $\nu_{i-1} + t_i \in [\![\phi_i]\!]$, and
- $\nu_i = r_i(\nu_{i-1} + t_i)$.

Therefore if $r_i = 0$, we have $\nu_i = 0$ and if $r_i = 1$ we have $\nu_i = \nu_{i-1} + t_i$. A pair $(q, \nu) \in Q \times \mathbb{R}_{\geq 0}$ like the ones occurring in the run e is called a *configuration* of $\mathcal{A}$ and the configuration $(q_I, 0)$ is called *initial*. The run e is deemed *accepting* if $q_k \in F$.

For $w \in \mathbb{T}\Sigma^*$ we write $(q, \nu) \rightsquigarrow^w (q', \nu')$ if there is a run of $\mathcal{A}$ on w from (q, ν) to (q', ν'). Observe that if $\mathcal{A}$ is deterministic then for every timed word w there is at most one run on w starting from the initial configuration. Finally, given a configuration (q, x) define $\mathcal{L}(q, x) = \{w \in \mathbb{T}\Sigma^* \mid (q, x) \rightsquigarrow^w (q, \nu), q \in F, \nu \in \mathbb{R}_{\geq 0}\}$, hence define $L(\mathcal{A})$ as $\mathcal{L}(q_I, 0)$.

An important technical tool in the analysis of timed automata is the *region equivalence* [4]. We recall this equivalence in the setting of one-clock timed automata.

Region Equivalence. Given a constant $K \in \mathbb{N}$ define the equivalence $\equiv^K \subseteq \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}$ by

$$x \equiv^K y \iff \Big(\lfloor x \rfloor = \lfloor y \rfloor \wedge (\mathit{frac}(x) = 0 \Leftrightarrow \mathit{frac}(y) = 0)\Big) \vee (x, y > K) \ ,$$

where given $x \in \mathbb{R}_{\geq 0}$ we denote by $\lfloor x \rfloor$ its integral part and by $\mathit{frac}(x)$ its fractional part.

Chapter 4

INCLUSION CHECKING OF LANGUAGES OF FINITE WORDS

In this chapter, we present the quasiorder-based framework by Ganty et al. [39, 40, 79] designed to address the language inclusion problem between regular languages of finite words. The original presentation of this framework was rooted in the context of abstract interpretation [20, 21]. Here, to simplify the exposition, we have removed any reliance on abstract interpretation.

4.1 Overview

We consider the inclusion problem $L^*(\mathcal{A}) \subseteq M$ where $\mathcal{A} = (Q, q_I, \delta, F)$ is a FA and M is a regular language. We derive an algorithm for solving this problem in three steps:

1. **Reduction to a Finite Basis:** we employ a well-quasiorder on finite words to derive from $L^*(\mathcal{A})$ a finite basis, denoted as S_{finite}. The key objective here is to establish the following equivalence, which reduces the inclusion problem to a finite number of membership queries in the language M:

$$L^*(\mathcal{A}) \subseteq M \iff \forall u \in S_{\text{finite}},\ u \in M \ . \tag{$\dagger$}$$

2. **Fixpoint Characterization:** we give a fixpoint characterization for $L^*(\mathcal{A})$ to iteratively produce the words of $L^*(\mathcal{A})$.
3. **Basis Detection:** we define a check to identify bases for $L^*(\mathcal{A})$ among the Kleene iterates of the fixpoint established in Step 2. Once we have successfully identified a basis, we can designate it as the set S_{finite} mentioned in Step 1.

Notice that a subset S_{finite} such that ($\dagger$) holds always exists: if the inclusion holds take S_{finite} to be any finite subset of $L^*(\mathcal{A})$ (empty set included); else take S_{finite} to contain some counterexample to inclusion.

Expectedly, the well-quasiorder we use in Step 1 needs to satisfy certain properties for S_{finite} to be computable and for ($\dagger$) to hold (in particular the left to right implication).

Definition 4.1.1. Given a regular language $M \subseteq \Sigma^*$ we say that a quasiorder $\leqslant$ on Σ^* is *M-preserving* when for every $u, v \in \Sigma^*$ if $u \in M$ and $u \leqslant v$ then $v \in M$. We say that $\leqslant$ is *M-suitable* if it is a M-preserving, right-monotonic and decidable well-quasiorder.

Intuitively, the "well"-property on the quasiorder ensures the finiteness of the basis. The preservation property ensures completeness: if the inclusion $L^*(\mathcal{A}) \subseteq M$ does not hold then a basis of $L^*(\mathcal{A})$ contains a counterexample to inclusion because a counterexample can only be discarded (that is, not included in S_{finite}) if it is subsumed by another counterexample in S_{finite}. Finally, the property of right-monotonicity is needed to compute S_{finite} as a terminating fixpoint computation.

4.2 An Algorithmic Framework for Checking Inclusion

In the following we detail the three steps of the framework.

4.2.1 Reduction to a Finite Basis

We fix a M-preserving well-quasiorder $\leqslant$ on Σ^*. We reduce the inclusion problem $L^*(\mathcal{A}) \subseteq M$ to an inclusion problem $S_{\text{finite}} \subseteq M$ where S_{finite} is obtained as a finite basis for $L^*(\mathcal{A})$ w.r.t. $\leqslant$. The direction $\Rightarrow$ of (†) trivially holds since S_{finite} is a subset of $L^*(\mathcal{A})$. For the reverse direction, assume that $\forall s \in S_{\text{finite}},\ s \in M$ and let $u \in L^*(\mathcal{A})$. Since S_{finite} is a basis for $L^*(\mathcal{A})$ there is $s \in S_{\text{finite}}$ such that $s \leqslant u$. Since $s \in M$, $s \leqslant u$ and $\leqslant$ is M-preserving we have $u \in M$.

4.2.2 Fixpoint Characterization

We work with the complete lattice $(\wp(\Sigma^*)^{|Q|}, \subseteq \times \cdots \times \subseteq)$, where each Cartesian product consists of $|Q|$ factors. Given a $|Q|$-dimensional vector X on $\wp(\Sigma^*)$ we write X_q for the q-component of X. Given $X \in \wp(\Sigma^*)^Q$, we define

$$\mathrm{Post}_{\mathcal{A}}(X) \triangleq \langle \textstyle\bigcup_{a \in \Sigma, (q', a, q) \in \delta} X_{q'} a \rangle_{q \in Q} \in \wp(\Sigma^*)^Q \ .$$

In turn, we define the map $f_{\mathcal{A}} \triangleq \lambda X. \langle \{\epsilon \mid q = i_{\mathcal{A}}\} \cup (\mathrm{Post}_{\mathcal{A}}(X))_q \rangle_{q \in Q}$, which allows us to give the following least fixpoint characterization of $L^*(\mathcal{A})$.

Example 8. *Consider the BA $\mathcal{C}$ in Figure 3.1. Since $\mathcal{C}$ has only one state, vectors have dimension one. We have $f_{\mathcal{C}} = \lambda X. \{\epsilon\} \cup Xa \cup Xb$. Its Kleene iterates are ${f_{\mathcal{C}}}^n(\emptyset) = \{u \in \{a, b\}^* \mid |u| \leq n - 1\}$ for every $n \in \mathbb{N}$.*

Proposition 1. $L^*(\mathcal{A}) = \bigcup_{p \in F} (\mathrm{lfp}\, f_{\mathcal{A}})_p$.

Proof. The function $f_{\mathcal{A}}$ is increasing and the supremum of the increasing sequence of its Kleene iterates starting at the bottom value $\vec{\emptyset} \triangleq (\emptyset, \ldots, \emptyset)$ of dimension $|Q|$ is the vector $\{u \in \Sigma^* \mid q_I \xrightarrow{u}{}^* q\}_{q \in Q}$. Therefore, by the Knaster–Tarski theorem $\mathrm{lfp}\, f_{\mathcal{A}} = \{u \in \Sigma^* \mid q_I \xrightarrow{u}{}^* q\}_{q \in Q}$. Thus, $\bigcup_{p \in F} (\mathrm{lfp}\, f_{\mathcal{A}})_p = L^*(\mathcal{A})$. □

Notice that each Kleene iterate of $f_{\mathcal{A}}$ is computable. Next we show how to detect if it is a basis for $L^*(\mathcal{A})$.

4.2.3 Basis Detection

In order to detect finite bases among the Kleene iterates of $f_{\mathcal{A}}$ we replace the set inclusion on $\wp(\Sigma^*)$, used so far, with the quasiorder $\sqsubseteq_\leqslant \subseteq \wp(\Sigma^*) \times \wp(\Sigma^*)$ defined by $X \sqsubseteq_\leqslant Y \stackrel{\triangle}{\iff} \forall x \in X, \exists y \in Y, y \leqslant x$. The quasiorder $\sqsubseteq_\leqslant$ leverage the notion of basis: given $X \in \wp(\Sigma^*)$ a subset $Y \subseteq X$ is a basis for X w.r.t. $\leqslant$ whenever $X \sqsubseteq_\leqslant Y$. In the following we lift the notion of basis to $|Q|$-dimensional vectors component wise and work with the quasiordered sets $(\wp(\Sigma^*)^{|Q|},\ \sqsubseteq_\leqslant^{|Q|})$, where the ordering $\sqsubseteq_\leqslant^{|Q|}$ is given by the product $\sqsubseteq_\leqslant \times \cdots \times \sqsubseteq_\leqslant$ of $|Q|$ factors.

As shown by the following lemma when the quasiorder $\leqslant$ is right-monotonic then $f_{\mathcal{A}}$ preserves bases w.r.t. $\leqslant$.

Lemma 1. *Let $\leqslant$ be a right-monotonic quasiorder. If B is a basis for $X \in \wp(\Sigma^*)^{|Q|}$ w.r.t. $\leqslant^{|Q|}$, then $f_{\mathcal{A}}(B)$ is a basis for $f_{\mathcal{A}}(X)$ w.r.t. $\leqslant^{|Q|}$.*

Proof. Since $B \subseteq^{|Q|} X$ and $f_{\mathcal{A}}$ is an increasing function, we immediately have $f_{\mathcal{A}}(B) \subseteq^{|Q|} f_{\mathcal{A}}(X)$. Now, we need to show that $f_{\mathcal{A}}(X) \sqsubseteq_\leqslant^{|Q|} f_{\mathcal{A}}(B)$. Let $q \in Q$ and $y \in (f_{\mathcal{A}}(X))_q$. We consider the case where y is in $(\mathrm{Post}_{\mathcal{A}}(X))_q$ as the case $y = \epsilon$ is straightforward. By definition of $f_{\mathcal{A}}$, $y = xa$ for some $x \in X_p$, $p \in Q$ and $a \in \Sigma$. Since $X_p \sqsubseteq_\leqslant B_p$ there is $b \in B_p$ such that $b \leqslant x$. Given that $\leqslant$ is right-monotonic, from $b \leqslant x$ we can deduce that $ba \leqslant y$. Finally, by definition of $f_{\mathcal{A}}$ we have $ba \in (\mathrm{Post}_{\mathcal{A}}(B))_q$. So, we have established that for every component $q \in Q$ of $f_{\mathcal{A}}$ we have $(f_{\mathcal{A}}(X))_q \sqsubseteq_\leqslant (f_{\mathcal{A}}(B))_q$, which concludes the proof. □

Proposition below shows that if $\leqslant$ is a right-monotonic well-quasiorder then to detect a finite basis it suffices to check whether ${f_{\mathcal{A}}}^{n+1}(\vec{\emptyset}) \sqsubseteq_\leqslant^{|Q|} {f_{\mathcal{A}}}^n(\vec{\emptyset})$ holds for two consecutive Kleene iterates of $f_{\mathcal{A}}$.

Proposition 2. *Let $\leqslant$ be a well-quasiorder. There is a positive integer n such that ${f_{\mathcal{A}}}^{n+1}(\vec{\emptyset}) \sqsubseteq_\leqslant^{|Q|} {f_{\mathcal{A}}}^n(\vec{\emptyset})$; and, if $\leqslant$ is right-monotonic then $\mathrm{lfp}\, f_{\mathcal{A}} \sqsubseteq_\leqslant^{|Q|} {f_{\mathcal{A}}}^n(\vec{\emptyset})$.*

Proof. The sequence of Kleene iterates of $f_{\mathcal{A}}$ is increasing w.r.t. $\sqsubseteq_\leqslant^{|Q|}$ and since $\leqslant$ is a well-quasiorder the quasiordered set $(\wp(\Sigma^*)^{|Q|},\ \sqsubseteq_\leqslant^{|Q|})$ satisfies the ascending chain condition [23]. Therefore, there is a positive integer n such that ${f_{\mathcal{A}}}^{n+1}(\vec{\emptyset}) \sqsubseteq_\leqslant^{|Q|} {f_{\mathcal{A}}}^n(\vec{\emptyset})$.

An easy induction that uses Lemma 1 shows that for every $k \geq n$ we have ${f_{\mathcal{A}}}^{k+1}(\vec{\emptyset}) \sqsubseteq_\leqslant^{|Q|} {f_{\mathcal{A}}}^k(\vec{\emptyset})$. Hence, by transitivity of $\sqsubseteq_\leqslant^{|Q|}$ we deduce that for every $k \geq n$ we have ${f_{\mathcal{A}}}^k(\vec{\emptyset}) \sqsubseteq_\leqslant^{|Q|} {f_{\mathcal{A}}}^n(\vec{\emptyset})$. Since the sequence of Kleene iterates of $f_{\mathcal{A}}$ is increasing we also have ${f_{\mathcal{A}}}^n(\vec{\emptyset}) \sqsubseteq_\leqslant^{|Q|} {f_{\mathcal{A}}}^k(\vec{\emptyset})$ for every $k \geq n$. We have ${f_{\mathcal{A}}}^n(\vec{\emptyset}) \subseteq^{|Q|} \mathrm{lfp}\, f_{\mathcal{A}}$ and since $\mathrm{lfp}\, f_{\mathcal{A}}$ is the supremum of the sequence of Kleene iterates of $f_{\mathcal{A}}$ we deduce that $\mathrm{lfp}\, f_{\mathcal{A}} \sqsubseteq_\leqslant^{|Q|} {f_{\mathcal{A}}}^n(\vec{\emptyset})$. Thus, ${f_{\mathcal{A}}}^n(\vec{\emptyset})$ is a basis for $\mathrm{lfp}\, f_{\mathcal{A}}$.

□

Finally, if the quasiorder $\leqslant$ is decidable i.e., given $x, y \in \Sigma^*$ we can decide whether $x \leq y$, then given two finite sets $X, Y \subseteq \Sigma^*$ it is also decidable whether $X \sqsubseteq_{\leqslant} Y$ holds. Thus, given a M-suitable quasiorder $\leqslant$ on Σ^* by the steps 1–3 we can compute a finite basis for $\operatorname{lfp} f_{\mathcal{A}}$ w.r.t. $\leqslant$.

4.3 Algorithm

In this section, we present Ganty et al.'s algorithm [40] for $L^*(\mathcal{A}) \subseteq M$. We also discuss an optimization of the algorithm that aims to reduce the number of words involved in the fixpoint computation by keeping antichains at every iteration. Lastly, we compare the algorithms derived by instantiating the framework with different quasiorders.

Algorithm 1: Algorithm for deciding $L^*(\mathcal{A}) \subseteq M$

Data: FA $\mathcal{A} = (Q, q_I, \delta, F)$.
Data: M-suitable quasiorder $\leqslant$.
Data: Procedure deciding $u \in M$ given u.

```
1 Compute f_A^m(∅⃗) with least m s.t. f_A^{m+1}(∅⃗) ⊑_≤^{|Q|} f_A^m(∅⃗);
2 foreach p ∈ F do
3     foreach u ∈ (f_A^m(∅⃗))_p do
4         if u ∉ M then return false;
5 return true;
```

Theorem 3. *Given the required inputs, Algorithm 1 decides the inclusion problem $L^*(\mathcal{A}) \subseteq M$.*

Proof. Given a M-suitable quasiorder $\leqslant$ and a procedure deciding membership in M, Algorithm 1 computes in line 1 a finite basis ${f_{\mathcal{A}}}^m(\vec{\emptyset})$ for $\operatorname{lfp} f_{\mathcal{A}}$ w.r.t. $\leqslant^{|Q|}$ as prescribed by Proposition 2 (each Kleene iterate is computable, given two iterates the check ${f_{\mathcal{A}}}^{k+1}(\vec{\emptyset}) \sqsubseteq_{\leqslant} {f_{\mathcal{A}}}^k(\vec{\emptyset})$ is decidable and the computation of line 1 terminates because $\leqslant$ is a well-quasiorder). Hence, by Proposition 1, $\bigcup_{p \in F}({f_{\mathcal{A}}}^m(\vec{\emptyset}))_p$ is a finite basis for $L^*(\mathcal{A})$ w.r.t. $\leqslant$. Finally, Algorithm 1 checks in lines 2-4 whether this finite basis is included in M. By M-preservation this basis satisfies Equation (†) and so we have $L^*(\mathcal{A}) \subseteq M \iff \forall u \in \bigcup_{p \in F}({f_{\mathcal{A}}}^m(\vec{\emptyset}))_p,\ u \in M$.

□

4.3.1 Antichains Optimization

Algorithm 1 remains correct if, at each Kleene iteration we first select, for each q-component, a basis w.r.t. $\leqslant$ and then apply $f_{\mathcal{A}}$. As shown by Lemma 1, if B is a basis for ${f_{\mathcal{A}}}^n(\vec{\emptyset})$ w.r.t. $\leqslant^{|Q|}$ then $f_{\mathcal{A}}(B)$ is a basis for ${f_{\mathcal{A}}}^{n+1}(\vec{\emptyset})$ w.r.t. $\leqslant^{|Q|}$. Thus, at each step $n \leq m$ of the iterations of line 1, we can replace ${f_{\mathcal{A}}}^n(\vec{\emptyset})$ by a basis $B \subseteq^{|Q|} {f_{\mathcal{A}}}^n(\vec{\emptyset})$ and then continue applying $f_{\mathcal{A}}$ from B. In particular, we can select basis that keep *antichains* for each q-component, that is, bases of incomparable words.

4.3.2 The Coarser the Better

Algorithm 1 is parametrized by a M-suitable quasiorder $\leqslant$ on Σ^*, so that each such quasiorder yields a slightly different algorithm deciding $L^*(\mathcal{A}) \subseteq M$. Let us discuss how the inclusion algorithms provided by different quasiorders can be related to each other. Consider two well-quasiorders $\leqslant, \leqslant' \subseteq \Sigma^* \times \Sigma^+$ such that $\leqslant$ is *coarser* than $\leqslant'$ i.e., $\leqslant' \subseteq \leqslant$ holds. It turns out that $X \sqsubseteq_{\leq'} Y$ implies $X \sqsubseteq_{\leq} Y$, so that if the Kleene iterate of $f_{\mathcal{A}}$ converge in N' steps w.r.t. $\leqslant'$, then they converge in $N \leq N'$ steps w.r.t. $\leqslant$, namely, convergence can be "faster" with a coarser quasiorder. Also, if $\leqslant$ is coarser than $\leqslant'$ and $X \in \wp(\Sigma^*)$ is a a nonempty set then any finite basis for X w.r.t. $\leqslant$ that is an antichain has at most as many elements as any finite basis for X w.r.t. $\leq'$ that is an antichain. Thus, a coarser well-quasiorder may achieve a smaller finite basis on which to perform the membership queries of Algorithm 1 (see Example 10 in the following section).

In the following section we provide different M-suitable quasiorders to instantiate Algorithm 1.

4.4 Quasiorders for Regular Languages

We now present two kinds of quasiorders to be used in Algorithm 1. The first kind of quasiorders, called *state-based quasiorders*, are derived from an automaton representation of M. The second kind, called *syntactic*, refers to quasiorders independent of any specific representation of M.

The quasiorders we introduce in this section will be also employed in the following sections, where we address the inclusion problem for languages of infinite words.

4.4.1 State-based Quasiorders

Given an automaton $\mathcal{B} = (\hat{Q}, \hat{q}_I, \hat{\delta}, \hat{F})$ and a word $u \in \Sigma^*$ we define the set

$$post^{\mathcal{B}}(u) \triangleq \{q \in \hat{Q} \mid \hat{q}_I \xrightarrow{u}{}^* q\}$$

of all the states that u can reach from the initial state $\hat{q}_I$, the set

$$ctx^{\mathcal{B}}(u) \triangleq \{(q, q') \in \hat{Q}^2 \mid q \xrightarrow{u}{}^* q'\}$$

of all the pairs of states that u "connects" in $\mathcal{B}$ and, the subset

$$ctx^{\mathcal{B}}_{\circledast}(u) \triangleq \{(q, q') \in \hat{Q}^2 \mid q \xrightarrow{u}{}^{\circledast} q'\}$$

of $ctx^{\mathcal{B}}(u)$ which keeps only the pairs of states connected by a sequence of transitions that pass by a final state. Using these sets, we define three quasiorders on Σ^* that compare words based on the sets of states that the words can reach or connect in $\mathcal{B}$.

Definition 4.4.1. Given an automaton $\mathcal{B} = (\hat{Q}, \hat{q}_I, \hat{\delta}, \hat{F})$ we define the following quasiorders on Σ^*, called the *state-based* quasiorders:

$$u \leqslant^{\mathcal{B}} v \overset{\triangle}{\iff} post^{\mathcal{B}}(u) \subseteq post^{\mathcal{B}}(v) \ ,$$
$$u \lesssim^{\mathcal{B}} v \overset{\triangle}{\iff} ctx^{\mathcal{B}}(u) \subseteq ctx^{\mathcal{B}}(v) \ ,$$
$$u \preccurlyeq^{\mathcal{B}} v \overset{\triangle}{\iff} u \lesssim^{\mathcal{B}} v \wedge ctx^{\mathcal{B}}_{\circledast}(u) \subseteq ctx^{\mathcal{B}}_{\circledast}(v) \ .$$

Note that we have $\preccurlyeq^{\mathcal{B}} \subseteq \lesssim^{\mathcal{B}} \subseteq \leqslant^{\mathcal{B}}$.

It's worth noting that $\preccurlyeq^{\mathcal{B}}$ is primarily designed for handling the infinite word case (although it can also be applied in the framework of Ganty et al.), but we include it here for clarity and organization.

Example 9. *Consider the automaton $\mathcal{D}$ in Figure 3.1 (b). Since for all $u \in \Sigma^*$, $post^{\mathcal{D}}(ua) = \{q\}$ and $post^{\mathcal{D}}(ub) = post^{\mathcal{D}}(\epsilon) = \{q_0\}$, we have that $u \leqslant^{\mathcal{D}} v$ iff either $u, v \in \Sigma^* a$ or $u, v \in \Sigma^* b \cup \{\epsilon\}$. Similarly, we find that $u \lesssim^{\mathcal{D}} v$ iff either $u, v \notin \{\epsilon\}$ and $u \leqslant^{\mathcal{D}} v$, or $u, v \in \{\epsilon\}$. For $u \in \Sigma^*$ we have $ctx^{\mathcal{D}}_{\circledast}(ua) = \{(q_0, q), (q, q)\}$. For $u \in \Sigma^* \setminus b^*$ we have $ctx^{\mathcal{D}}_{\circledast}(ub) = \{(q_0, q_0), (q, q_0)\}$ and $ctx^{\mathcal{D}}_{\circledast}(b^k) = \{(q, q_0)\}$, for any $k \geq 1$. As for the empty word, $ctx^{\mathcal{D}}_{\circledast}(\epsilon) = \{(q, q)\}$. Hence, for all $u, v \in \Sigma^*$, it turns out that $u \preccurlyeq^{\mathcal{D}} v$ holds iff one of the following four cases holds: (i) $u, v \in \Sigma^* a$; (ii) $u \in \Sigma^* b$ and $v \in \Sigma^* b \backslash b^*$; (iii) $u, v \in b^+$; (iv) $u, v \in \{\epsilon\}$.*

Proposition 3. *The state-based quasiorders are $L^*(\mathcal{B})$-suitable. Additionally, the quasiorders $\lesssim^{\mathcal{B}}$ and $\preccurlyeq^{\mathcal{B}}$ are monotonic.*

Proof. From the equivalences $u \in L^*(\mathcal{B}) \iff post^{\mathcal{B}}(u) \cap F \neq \emptyset$ and $u \in L^*(\mathcal{B}) \iff ctx^{\mathcal{B}}(u) \cap (\{\hat{q}_I\} \times F) \neq \emptyset$ we deduce that the state-based quasiorders are $L^*(\mathcal{B})$-preserving. Since Q is finite they are all well quasiorders. The proof of decidability is trivial by Definition 4.4.1. Next we show that $\leqslant^{\mathcal{B}}$ is right-monotonic. Let $u \leqslant^{\mathcal{B}} v$ and $a \in \Sigma$. Let $q \in post^{\mathcal{B}}(ua)$. There is $p \in post^{\mathcal{B}}(u)$ such that $p \xrightarrow{a} q$. Since $u \leqslant^{\mathcal{B}} v$ and $p \in post^{\mathcal{B}}(u)$ we have $p \in post^{\mathcal{B}}(v)$, thus $q \in post^{\mathcal{B}}(va)$. An analogue reasoning shows that $\lesssim^{\mathcal{B}}$ and $\preccurlyeq^{\mathcal{B}}$ are monotonic. □

By Proposition 3, Algorithm 1 can be instantiated with any of the state-based quasiorders. Since $\leqslant^{\mathcal{B}}$ is the coarsest among them, its corresponding algorithm yields more pruning when searching for a counterexample to inclusion as shown by Example 10 (see also discussion in Section 4.3.2). On the other hand, the quasiorders $\leqslant^{\mathcal{B}}$ and $\preccurlyeq^{\mathcal{B}}$ verify the stronger property of both left and right-monotonicity, which as we will see in Section 5.5, is needed to handle the inclusion problem of context free languages into regular traces. The additional condition on final states required by $\preccurlyeq^{\mathcal{B}}$ will come in handy when deciding the inclusion in (infinite) traces of Büchi automata (Chapter 5).

Example 10 (The Coarser the Better)**.** *We show the benefits of using the coarsest state-based quasiorder $\leqslant^{\mathcal{B}}$ on the family of inclusion problems between the automata $\{\mathcal{A}_n\}_{n\geq 2}$ and $\{\mathcal{B}_n\}_{n\geq 2}$ such that $L^*(\mathcal{A}_n) \subseteq L^*(\mathcal{B}_n)$ for all n, depicted in Figure 4.1. Let $X_n \triangleq \{a^i b a^{j+1} \in \Sigma^* \mid i, j \geq 0, i + j \leq n - 1\}$ such that $L^*(\mathcal{A}_n) = X_n\{b\}^*$. For any $w \in L^*(\mathcal{A}_n)$ we have that $q_n \in post^{\mathcal{B}_n}(w)$, and, since $post^{\mathcal{B}_n}(aba) = \{q_n\}$, it holds that $aba \leqslant^{\mathcal{B}_n} w$. Thus, $\{aba\}$ is a basis for $L^*(\mathcal{A}_n)$ w.r.t. $\leqslant^{\mathcal{B}_n}$. Since $c^{\mathcal{B}_n}[a^i b a^{j+1}] = \{(n - i, j + 2), (0, q_n), (q_n, q_n)\}$ we deduce that $w \preccurlyeq^{\mathcal{B}_n} w'$ implies $w = w'$ for every $w, w' \in X_n$. Thus, as X_n has size $\frac{n(n+1)}{2}$, any basis for $L^*(\mathcal{A}_n)$ w.r.t. $\lesssim^{\mathcal{B}_n}$ or $\preccurlyeq^{\mathcal{B}_n}$ has at least $\frac{n(n+1)}{2}$ elements. Hence, using the quasiorder $\leqslant^{\mathcal{B}_n}$, a*

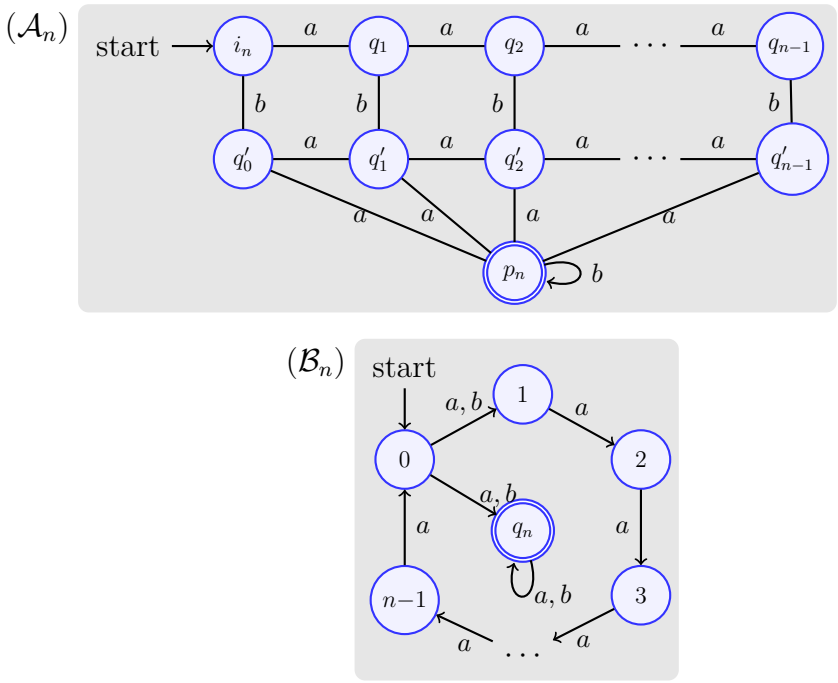


Figure 4.1: The families $\{\mathcal{A}_n\}_{n\geq 2}$ and $\{\mathcal{B}_n\}_{n\geq 2}$.

single membership query (i.e., $u \in L^*(\mathcal{B}_n)$) is needed to decide the inclusion $L^*(\mathcal{A}_n) \subseteq L^*(\mathcal{B}_n)$, as opposed to no less than $\frac{n(n+1)}{2}$ membership queries for the other quasiorders.

4.4.2 A Syntactic Quasiorder

Given a language $M \subseteq \Sigma^*$ we define the following syntactic quasiorder on Σ^*:

$$u \leqslant^M v \stackrel{\triangle}{\iff} \forall w \in \Sigma^*, uw \in M \implies vw \in M \ .$$

Proposition 4. *If M is a regular language then $\leqslant^M$ is M-suitable. Moreover, $\leqslant^M$ is the coarsest M-suitable quasiorder.*

Proof. First we show that any M-suitable quasiorder $\leqslant$ is finer than $\leqslant^M$. Assume $u \leqslant u'$ and let $w \in \Sigma^*$ such that $uw \in M$. By right-monotonicity we have $uw \leqslant u'w$. Hence, by M-preservation $u'w \in M$. Thus, $u \leqslant^M u'$.

If M is accepted by a FA $\mathcal{B}$ then by Proposition 3, $\leqslant^{\mathcal{B}}$ is M-suitable, thus $\leqslant^{\mathcal{B}} \subseteq \leqslant^M$. Since $\leqslant^{\mathcal{B}}$ is a well-quasiorder and $\leqslant^{\mathcal{B}} \subseteq \leqslant^M$, $\leqslant^M$ is a well-quasiorder. Decidability comes from the decidability of the inclusion problem between regular languages. Right-monotonicity and M-preservation are an easy exercise. □

4.5 State-based Algorithms

The state-based quasiorders enable us to derive *state-based* inclusion algorithms from Algorithm 1, namely, algorithms that, given two FA $\mathcal{A} = (Q, q_I, \delta, F)$ and $\mathcal{B} = (\hat{Q}, \hat{q}_I, \hat{\delta}, \hat{F})$, decide whether $L^*(\mathcal{A}) \subseteq L^*(\mathcal{B})$ by operating on the states of $\mathcal{A}$ and $\mathcal{B}$ only, without the need to store and manipulate words at all. In this section we explain how to obtain such state-based algorithms using the quasiorders of Definition 4.4.1. We then formally define the state-based algorithm derived by instantiating Algorithm 1 with the quasiorder $\preccurlyeq^{\mathcal{B}}$.

4.5.1 Data Structures

Comparing two words given a state-based quasiorder requires to compute the corresponding sets of $post^{\mathcal{B}}$, $ctx^{\mathcal{B}}$ and $ctx^{\mathcal{B}}_{\circledast}$. In the following we show that instead of computing these sets every time we need to compare two words we can reason directly on the $post^{\mathcal{B}}$, $ctx^{\mathcal{B}}$ and $ctx^{\mathcal{B}}_{\circledast}$ information of the words, for comparisons, for applying the fixpoint function, for convergence check and for membership check. Assume we are computing a new word during the fixpoint computation, for instance the word ua that is obtained by concatenating the letter a to the right of u. As shown next $post^{\mathcal{B}}(ua)$, $ctx^{\mathcal{B}}(ua)$ and $ctx^{\mathcal{B}}_{\circledast}(ua)$ can be computed directly from $post^{\mathcal{B}}(u)$, $ctx^{\mathcal{B}}(u)$ and $ctx^{\mathcal{B}}_{\circledast}(u)$ instead of computing it from "scratch" :

$$post^{\mathcal{B}}(ua) = \{q \in \hat{Q} \mid \exists p \in \hat{Q}, p \in post^{\mathcal{B}}(u), (p, q) \in ctx^{\mathcal{B}}(a)\} \ , \tag{4.1}$$

and analogously,

$$ctx^{\mathcal{B}}(ua) = \{(p, q) \in \hat{Q}^2 \mid \exists p_0 \in \hat{Q}, (p, p_0) \in ctx^{\mathcal{B}}(u), (p_0, q) \in ctx^{\mathcal{B}}(a)\} \ , \tag{4.2}$$

and

$$\begin{aligned} ctx^{\mathcal{B}}_{\circledast}(ua) = \{(p, q) \in \hat{Q}^2 \mid \exists p_0 \in \hat{Q}, (p, p_0) \in ctx^{\mathcal{B}}_{\circledast}(u), (p_0, q) \in ctx^{\mathcal{B}}(a)\} \\ \cup \{(p, q) \in \hat{Q}^2 \mid \exists p_0 \in \hat{Q}, (p, p_0) \in ctx^{\mathcal{B}}(u), (p_0, q) \in ctx^{\mathcal{B}}_{\circledast}(a)\} \ . \end{aligned} \tag{4.3}$$

Example 11. *Consider the BA $\mathcal{D}$ in Figure 3.1. Using the above definition it is straightforward to verify that $post^{\mathcal{D}}(ab) = \{q_0\}$ because $post^{\mathcal{D}}(a) = \{q\}$ and $ctx^{\mathcal{D}}(b) = \{(q_0, q_0), (q, q_0)\}$. Similarly, we find that $ctx^{\mathcal{D}}(ab) = \{(q_0, q_0), (q, q_0)\}$ because $ctx^{\mathcal{D}}(a) = \{(q_0, q), (q, q)\}$ and $ctx^{\mathcal{D}}(b) = \{(q_0, q_0), (q, q_0)\}$.*

In the following section, we provide a detailed definition of the state-based algorithm derived using the quasiorder $\preccurlyeq^{\mathcal{B}}$, building upon the ideas explained here.

4.5.2 Detailed Algorithm for $\preccurlyeq^{\mathcal{B}}$

To establish the state-based algorithm with $\preccurlyeq^{\mathcal{B}}$ we first define a 'state-based version' of the function $f_{\mathcal{A}}$. To do so, we work with the complete lattice $(\wp(\wp(\hat{Q}))^{|Q|}, \subseteq \times \cdots \times \subseteq)$, where each Cartesian product consists of $|Q|$ factors. Given $X \in \wp(\wp(\hat{Q}))^Q$, we define

$$\mathrm{Post}^{\preccurlyeq^{\mathcal{B}}}_{\mathcal{A}}(X) \triangleq \langle \textstyle\bigcup_{a \in \Sigma, (q,a,q') \in \delta} \{y \star a \in \wp(\hat{Q}) \mid \exists y \in X_{q'}\} \rangle_{q \in Q} \in \wp(\wp(\hat{Q}))^Q \ ,$$

where for $y \in \wp(\hat{Q})$ and $a \in \Sigma$ we define $y \star a \triangleq \{q \in \hat{Q} \mid \exists q' \in \hat{Q}, q' \in y, (q', q) \in ctx^{\mathcal{B}}(a)\}$. In turn we define the map $\hat{f}_{\mathcal{A}} : \wp(\wp(\hat{Q}))^{|Q|} \to \wp(\wp(\hat{Q}))^{|Q|}$ by

$$\hat{f}_{\mathcal{A}} \triangleq \lambda X.\langle\{\{\hat{q}_I\} \mid q = \hat{q}_I\} \cup \ (\mathrm{Post}_{\mathcal{A}}^{\leqslant^{\mathcal{B}}}(X))_q\rangle_{q \in Q} \ .$$

Next, we extend the function $post^{\mathcal{B}}$ to sets by defining for $X \in \wp(\Sigma^*)$, $post^{\mathcal{B}}(X) = \cup_{x \in X} post^{\mathcal{B}}(x)$. We also extend it to vectors component-wise.

Lemma 2. *For every $X \in \wp(\Sigma^*)^{|Q|}$ we have $\hat{f}_{\mathcal{A}} \circ post^{\mathcal{B}}(X) = post^{\mathcal{B}} \circ f_{\mathcal{A}}(X)$.*

Proof. By Equation (4.1) if $y = post^{\mathcal{B}}(u)$, for some $u \in \Sigma^*$, then $y \star a = post^{\mathcal{B}}(ua)$. The proof then follows directly from the definitions of the functions $f_{\mathcal{A}}$ and $\hat{f}_{\mathcal{A}}$. □

We check convergence of the fixpoint computation in line 1 of Algorithm 1 by reasoning on the function $\hat{f}_{\mathcal{A}}$ and by using the quasiorder $\sqsubseteq_{\subseteq}$ on $\wp(\wp(\hat{Q}))$. Using Lemma 2 it is an easy exercise to establish that for every $n \in \mathbb{N}$, $\hat{f}_{\mathcal{A}}^n(\vec{\emptyset}) = post^{\mathcal{B}}({f_{\mathcal{A}}}^n(\vec{\emptyset})$ and, consequently,

$${f_{\mathcal{A}}}^{m+1}(\vec{\emptyset}) \sqsubseteq_{\leqslant^{\mathcal{B}}}^{|Q|} {f_{\mathcal{A}}}^m(\vec{\emptyset}) \iff \hat{f}_{\mathcal{A}}^{m+1}(\vec{\emptyset}) \sqsubseteq_{\subseteq}^{|Q|} \hat{f}_{\mathcal{A}}^m(\vec{\emptyset}) \ .$$

Incidentally, we perform the membership querries of line 4 (asking whether $u \in L^*(\mathcal{B})$ given u) by checking whether the $post^{\mathcal{B}}$ associated to the word u contains a final state of $\mathcal{B}$.

Below we present the state-based algorithm obtained by instantiating Algorithm 1 with the quasiorder $\leqslant^{\mathcal{B}}$.

Algorithm 2: Algorithm for deciding $L^*(\mathcal{A}) \subseteq L^*(\mathcal{B})$

Data: Büchi automata $\mathcal{A} = (Q, q_I, \delta, F)$ and $\mathcal{B} = (\hat{Q}, \hat{q}_I, \hat{\delta}, \hat{F})$

1 Compute $\hat{f}_{\mathcal{A}}^m(\vec{\emptyset})$ with least m s.t. $\hat{f}_{\mathcal{A}}^{m+1}(\vec{\emptyset}) \sqsubseteq_{\subseteq}^{|Q|} \hat{f}_{\mathcal{A}}^m(\vec{\emptyset})$;
2 **foreach** $p \in F$ **do**
3 **foreach** $x \in (\hat{f}_{\mathcal{A}}^m(\vec{\emptyset}))_p$ **do**
4 **if** $x \cap \hat{F} = \emptyset$ **then return false**;
5 **return true**;

Theorem 4. *Given the required inputs, Algorithm 2 decides the inclusion problem $L^*(\mathcal{A}) \subseteq L^*(\mathcal{B})$.*

Proof. Follows from Theorem 3. □

Proposition 5. *Let $n \triangleq |Q|$ and $\hat{n} \triangleq |\hat{Q}|$. The running time of the state-based algorithm is $|\Sigma| \cdot n^2 \cdot 2^{O(\hat{n})}$.*

Proof. At worst the computation of line 1 of Algorithm 2 adds exactly one element of $\wp(\hat{Q})$ at each iteration step k to some component of the n-dimensional vector $\hat{f}_{\mathcal{A}}^k(\vec{\emptyset})$, so that $n \cdot 2^{\hat{n}}$ is an upper bound on the number of iterations needed to compute $\hat{f}_{\mathcal{A}}^m(\vec{\emptyset})$. For $X, Y \subseteq \wp(\hat{Q})$ the time to check $X \sqsubseteq_{\subseteq} Y$ is bounded by $2^{O(\hat{n})}$. To infer an upper bound on the runtime of

line 1 we also need to multiply the above expression by a factor $|\Sigma| \cdot n$ since the number of concatenations in the fixpoint function depends on the size of the alphabet and on n.

The number of iterations of the loops of lines 2 and 3 is n and $2^{\hat{n}}$ respectively. Since all loops are nested, we multiply these bounds to end up with $n \cdot 2^{\hat{n}}$ as an upper bound on the number of times the check in line 4 is performed. The runtime for each membership query is upper bounded by $2^{\mathcal{O}(\hat{n})}$. We conclude from the above that the runtime of Algorithm 2 is at most $|\Sigma| \cdot n^2 \cdot 2^{\mathcal{O}(\hat{n})}$. □

To conclude this chapter, we present an illustrative example of an execution of Algorithm 2.

4.5.3 Illustrative Example

We show the execution of a run of the state-based algorithm obtained on the FA $\mathcal{C}$ and $\mathcal{D}$ depicted in Figure 3.1. As a result, the algorithm will correctly decide that $L^*(\mathcal{C})$ is not included in $L^*(\mathcal{D})$ since, for example $b \in L^*(\mathcal{C})$ but $b \notin L^*(\mathcal{D})$. Observe that since $\mathcal{C}$ consists of a single state, vectors are of dimension one. First, the algorithm evaluates the sequence $\{\hat{f}^n_{\mathcal{C}}(\emptyset)\}_{n\in\mathbb{N}}$ where $\hat{f}^n_{\mathcal{C}}(\emptyset) \in \wp(\wp(\{q_0, q\}))$ for every $n \in \mathbb{N}$ and $\hat{f}_{\mathcal{C}}$ is defined for $X \in \wp(\wp(\{q_0, q\}))$ by

$$\hat{f}_{\mathcal{C}}(X) = \{\{q_0\}\} \cup \bigcup_{c\in\{a,b\}} \{s \in \{q_0, q\} \mid \exists x \in X, p \in x, (p, s) \in ctx^{\mathcal{D}}(c)\} \ .$$

We have:

- $\hat{f}^1_{\mathcal{C}}(\emptyset) = \{\{q_0\}\}$,
- $\hat{f}^2_{\mathcal{C}}(\emptyset) = \{\{q_0\}, \{q\}\}$,
- $\hat{f}^3_{\mathcal{C}}(\emptyset) = \{\{q_0\}, \{q\}\}$.

Hence, the computations for the prefix iterates stop at the third iteration and the algorithm continues with the membership check of line 4 on the elements of $\hat{f}^2_{\mathcal{C}}(\emptyset)$. Since $q_0 \notin \hat{F}$ the membership check fails for the element $\{q_0\}$ of $\hat{f}^2_{\mathcal{C}}(\emptyset)$, which intuitively, corresponds to the counterexample b that belongs to $L^*(\mathcal{C})$ but not $L^*(\mathcal{D})$. Hence, the inclusion $L^*(\mathcal{C}) \subseteq L^*(\mathcal{D})$ does not hold.

Chapter 5

INCLUSION FOR INFINITE WORDS

In this chapter, we address the problem of deciding inclusion between regular languages of infinite words. We then adapt our framework to the inclusion problem of ω-context free languages into ω-regular languages.

5.1 Overview

In this section we outline our framework for solving the inclusion problem $L^\omega(\mathcal{A}) \subseteq M$, where $\mathcal{A} = (Q, q_I, \delta, F)$ is a BA and M is a ω-regular language. We follow the same principles as in the finite word case (Chapter 4): using an ordering on words we reduce the inclusion check to a finite number of membership queries in M applied to words from $L^\omega(\mathcal{A})$. To obtain this reduction, we leverage a well-known theorem:

Theorem 5 ([17])**.** *For $\mathcal{A}$ a BA and M a ω-regular language we have $L^\omega(\mathcal{A}) \subseteq M \iff \forall uv^\omega \in L^\omega(\mathcal{A}), uv^\omega \in M$.*

Our goal is to define a finite subset $S_{\text{finite}} \subseteq \Sigma^* \times \Sigma^+$ of decompositions of ultimately periodic words in $L^\omega(\mathcal{A})$ such that the following equivalence holds.

$$L^\omega(\mathcal{A}) \subseteq M \iff \forall (u,v) \in S_{\text{finite}},\ uv^\omega \in M \ . \tag{††}$$

A Sufficient Subset of Decompositions We begin by defining the following subset

$$S \triangleq \bigcup_{p \in F} L_{q_I,p} \times (L_{p,p} \backslash \{\epsilon\})$$

of decompositions of ultimately periodic words in $L^\omega(\mathcal{A})$ where for every pair $p, q \in Q$ of states of $\mathcal{A}$, $L_{p,q} \triangleq \{u \in \Sigma^* \mid p \xrightarrow{u}{}^* q\}$ is the language containing all the words that can lead from state p to q according to $\mathcal{A}$. This subset is sufficient for the inclusion problem $L^\omega(\mathcal{A}) \subseteq M$ as shown by the next lemma.

Lemma 3. $L^\omega(\mathcal{A}) \subseteq M \iff \forall (u,v) \in S,\ uv^\omega \in M.$

Proof. By Theorem 5 it suffices to show that for every $(u,v) \in \Sigma^* \times \Sigma^+$ we have $uv^\omega \in L^\omega(\mathcal{A}) \iff \exists (u',v') \in S,\ uv^\omega = u'v'^\omega$. By definition of the languages $L_{p,q}$ it is immediate

that for every $(u, v) \in S$ there is an accepting run of $\mathcal{A}$ on uv^ω, thus $uv^\omega \in L^\omega(\mathcal{A})$. For the converse implication, we notice that every accepting run of $\mathcal{A}$ on uv^ω has a final state $p \in F$ that appears infinitely often, thus such a run can be factorized as $q_I \xrightarrow{u'}{}^* p \xrightarrow{v'}{}^* p$ where $uv^\omega = u'v'^\omega$. □

Reduction to a Finite Basis We employ a pair $\leqslant, \preccurlyeq$ of quasiorders on Σ^* in order to derive a finite basis $S_{\text{finite}} \subseteq \Sigma^* \times \Sigma^+$ for S w.r.t. $\leqslant \times \preccurlyeq$. Expectedly, this pair of quasiorders needs to verify certain requirements for (††) to hold and yield a correct algorithm.

Definition 5.1.1. Given a ω-regular language $M \subseteq \Sigma^*$ we say that a pair $\leqslant, \preccurlyeq$ of quasiorders on Σ^* is *M-preserving* when for every $(u, v), (u', v') \in \Sigma^* \times \Sigma^+$, if $uv^\omega \in M, u \leqslant u'$ and $v \preccurlyeq v'$ then $u'v'^\omega \in M$. We say that $\leqslant, \preccurlyeq$ is *M-suitable* if it is a M-preserving, right-monotonic and decidable pair of well-quasiorders.

The intuition behind these requirements is the same as in Section 4.1: The "well" property is for finiteness of the basis[1], the preservation property for completeness, and the right-monotonicity for computation.

5.2 Framework

In this section we give a fixpoint characterization for S and show that we can compute a finite basis for it, provided that we have a suitable pair of quasiorders. We then present our algorithm for the inclusion problem $L^\omega(\mathcal{A}) \subseteq M$.

5.2.1 Fixpoint Characterization

To obtain a least fixpoint characterization for S we work with the complete lattice $(\wp(\Sigma^*)^{|Q|}, \subseteq \times \cdots \times \subseteq)$ and follow the notations from Section 4.2.2. We recall the definition of $X \in \wp(\Sigma^*)^{|Q|} \mapsto \mathrm{Post}_{\mathcal{A}}(X) \triangleq \langle \bigcup_{a \in \Sigma, q \in \delta(q', a)} X_{q'} a \rangle_{q \in Q} \in \wp(\Sigma^*)^Q$ from Section 4.2.2. We obtain a fixpoint characterization of the prefixes of S using the function

$$f_{\mathcal{A}} = \lambda X. \langle \{\epsilon \mid q = i_{\mathcal{A}}\} \cup \ (\mathrm{Post}_{\mathcal{A}}(X))_q \rangle_{q \in Q}$$

and, a fixpoint characterization of the periods using for every $p \in F$ the functions

$$r_{\mathcal{A},p} = \lambda X. \langle \{a \in \Sigma \mid q \in \delta(p, a)\} \cup \ (\mathrm{Post}_{\mathcal{A}}(X))_q \rangle_{q \in Q} \ .$$

In turn, we achieve the fixpoint characterization of S given in the following proposition.

Proposition 6. $S = \bigcup_{p \in F} (\mathrm{lfp}\, f_{\mathcal{A}})_p \times (\mathrm{lfp}\, r_{\mathcal{A},p})_p$.

Proof. By an analogue reasoning as in the proof of Proposition 1 we find that $L_{q_I,p} = (\mathrm{lfp}\, f_{\mathcal{A}})_p$ and $(L_{p,p} \backslash \{\epsilon\}) = (\mathrm{lfp}\, r_{\mathcal{A},p})_p$. Thus, $S = \bigcup_{p \in F} (\mathrm{lfp}\, f_{\mathcal{A}})_p \times (\mathrm{lfp}\, r_{\mathcal{A},p})_p$.

□

[1]The quasiorder $\leqslant \times \preccurlyeq$ is a well-quasiorder when both $\leqslant$ and $\preccurlyeq$ are well-quasiorders.

Example 12. *Consider the BA $\mathcal{C}$ in Figure 3.1. Since $\mathcal{C}$ has only one state, vectors have dimension one. We have that $f_{\mathcal{C}} = \lambda X.\{\epsilon\} \cup Xa \cup Xb$ and $r_{\mathcal{C}} = \lambda X.\{a, b\} \cup Xa \cup Xb$, so that $\{a, b\}^* \times \{a, b\}^+ = \operatorname{lfp} f_{\mathcal{C}} \times \operatorname{lfp} r_{\mathcal{C}}$.*

5.2.2 Basis Detection

We fix a M-suitable pair $\leqslant, \preccurlyeq$ of quasiorders on Σ^*. We detect a basis for the least fixed point of $f_{\mathcal{A}}$ and $r_{\mathcal{A},p}$ among their Kleene iterates by using the quasiorders $\sqsubseteq_{\leqslant}^{|Q|}$ and $\sqsubseteq_{\preccurlyeq}^{|Q|}$ on $\wp(\Sigma^*)^{|Q|}$ defined by $X \sqsubseteq_{\ltimes} Y \iff \forall q \in Q, \forall x \in X_q, \exists y \in Y_q, y \ltimes x$ for $\ltimes \in \{\leqslant, \preccurlyeq\}$, as explained in Section 4.2.3 (Lemma 1 and Proposition 2 hold for the functions $r_{\mathcal{A},p}$ as well).

5.2.3 Algorithm

We now present our algorithm which given a M-suitable pair of quasiorders $\leqslant, \preccurlyeq$ along with a procedure that can determine membership in the language M, solves the inclusion problem $L^\omega(\mathcal{A}) \subseteq M$.

In line 1, Algorithm 3 computes a finite basis ${f_{\mathcal{A}}}^m(\vec{\emptyset})$ for the least fixed point $\operatorname{lfp} f_{\mathcal{A}}$ w.r.t. $\leqslant$. Subsequently, for each $p \in F$, Algorithm 3, at line 3, computes a finite basis ${r_{\mathcal{A},p}}^{m'}(\vec{\emptyset})$ for the least fixed point $\operatorname{lfp} r_{\mathcal{A},p}$ w.r.t. $\preccurlyeq$. Lastly, at lines 4–5, the algorithm proceeds to verify whether each ultimately periodic word within $S_{\text{finite}} = \bigcup_{p \in F} ({f_{\mathcal{A}}}^m(\vec{\emptyset}))_p \times ({r_{\mathcal{A},p}}^{m'}(\vec{\emptyset}))_p$ belongs to M.

Algorithm 3: Algorithm for deciding $L^\omega(\mathcal{A}) \subseteq M$

Data: BA $\mathcal{A} = (Q, q_I, \delta, F)$.
Data: M-suitable pair $\leqslant, \preccurlyeq$.
Data: Procedure deciding $uv^\omega \in M$ given (u, v).

1 Compute ${f_{\mathcal{A}}}^m(\vec{\emptyset})$ with least m s.t. ${f_{\mathcal{A}}}^{m+1}(\vec{\emptyset}) \sqsubseteq_{\leqslant}^{|Q|} {f_{\mathcal{A}}}^m(\vec{\emptyset})$;
2 **foreach** $p \in F$ **do**
3 Compute ${r_{\mathcal{A},p}}^{m'}(\vec{\emptyset})$ with least m' s.t. ${r_{\mathcal{A},p}}^{m'+1}(\vec{\emptyset}) \sqsubseteq_{\preccurlyeq}^{|Q|} {r_{\mathcal{A},p}}^{m'}(\vec{\emptyset})$;
4 **foreach** $u \in ({f_{\mathcal{A}}}^m(\vec{\emptyset}))_p$, $v \in ({r_{\mathcal{A},p}}^{m'}(\vec{\emptyset}))_p$ **do**
5 **if** $uv^\omega \notin M$ **then return** false;
6 **return** true;

Notice that by omitting the fixpoint computations for the periods at line 3 we obtain Algorithm 3.

Theorem 6. *Given the required inputs, Algorithm 3 decides the inclusion problem $L^\omega(\mathcal{A}) \subseteq M$.*

Proof. The subset $S_{\text{finite}} = \bigcup_{p \in F} ({f_{\mathcal{A}}}^m(\vec{\emptyset}))_p \times ({r_{\mathcal{A},p}}^{m'}(\vec{\emptyset}))_p$ is a finite basis for S w.r.t. $\leqslant \times \preccurlyeq$. Hence, since the pair $\leqslant, \preccurlyeq$ is M-preserving, by Lemma 3 we deduce that S_{finite} satisfies Equation (††). Thus, we have

$$L^\omega(\mathcal{A}) \subseteq M \iff \forall p \in F, \forall u \in ({f_{\mathcal{A}}}^m(\vec{\emptyset}))_p, \forall v \in ({r_{\mathcal{A},p}}^{m'}(\vec{\emptyset}))_p,\ uv^\omega \in M \ .$$

Each Kleene iterate of $f_{\mathcal{A}}$ and $r_{\mathcal{A},p}$ is computable, the checks ${f_{\mathcal{A}}}^{n+1}(\vec{\emptyset}) \sqsubseteq_{\preccurlyeq}^{|Q|} {f_{\mathcal{A}}}^{n}(\vec{\emptyset})$ and ${r_{\mathcal{A},p}}^{n+1}(\vec{\emptyset}) \sqsubseteq_{\preccurlyeq}^{|Q|} {r_{\mathcal{A},p}}^{n}(\vec{\emptyset})$ are decidable and since $\leqslant$ and $\preccurlyeq$ are well-quasiorders the computations at line 1 and line 3 terminate. □

Antichain Optimization The analogue arguments as in Section 4.3.1 show that Algorithm 3 remains correct if at each iteration we discard from a component of ${f_{\mathcal{A}}}^{m}(\vec{\emptyset})$ or ${r_{\mathcal{A},p}}^{m'}(\vec{\emptyset})$ words that are subsumed by others, and that in particular we can keep antichains. We refer to the optimization that keeps antichains for all the sets as the *antichain optimization*.

The Coarser the Better The discussion in Section 4.3.2 applies similarly to Algorithm 3: a coarser pair of quasiorders may achieve a smaller finite basis on which to perform the membership queries of line 5 (see Example 14 in the following section).

5.3 Suitable Pairs of Quasiorders

In the preceding section, we presented Algorithm 3 which is parameterized by a pair of quasiorders. Notably, each distinct pair of quasiorders leads to a variation of the algorithm. In this section we discuss different pairs of quasiorders that can be used in Algorithm 3.

5.3.1 State-based Pairs

We start with the state-based quasiorders of Definition 4.4.1.

Proposition 7. *Let $\mathcal{B}$ be a BA, the pairs $\leqslant^{\mathcal{B}}, \preccurlyeq^{\mathcal{B}}$ and $\lesssim^{\mathcal{B}}, \preccurlyeq^{\mathcal{B}}$ and $\preccurlyeq^{\mathcal{B}}, \preccurlyeq^{\mathcal{B}}$ are $L(\mathcal{B})$-suitable.*

Proof. All the state-based quasiorders are right-monotonic, decidable and well-quasiorders as shown by Proposition 3. Next we show that the pairs in the statement are $L(\mathcal{B})$-preserving, starting with $\leqslant^{\mathcal{B}}, \preccurlyeq^{\mathcal{B}}$. Let $(u, v), (u', v') \in \Sigma^* \times \Sigma^+$ such that $uv^\omega \in M$, $u \leqslant^{\mathcal{B}} u'$ and $v \preccurlyeq^{\mathcal{B}} v'$. Since $uv^\omega \in M$ there is an accepting run e of $\mathcal{B}$ of the form $e: \; q_I \xrightarrow{u} q_1 \xrightarrow{v} q_2 \xrightarrow{v} q_3 \cdots$ Since $post^{\mathcal{B}}(u) \subseteq post^{\mathcal{B}}(u')$ and $ctx^{\mathcal{B}}(u) \subseteq ctx^{\mathcal{B}}(u')$ from e we deduce a run $e': \; q_I \xrightarrow{u'} q_1 \xrightarrow{v'} q_2 \xrightarrow{v'} q_3 \cdots$ of $\mathcal{B}$ on $u'v'^\omega$. Since e is accepting $(q_j, q_{j+1}) \in ctx^{\mathcal{B}}_{\circledast}(v)$ for infinitely many j's. Since $ctx^{\mathcal{B}}_{\circledast}(v) \subseteq ctx^{\mathcal{B}}_{\circledast}(v')$ we thus deduce that e' is accepting. Thus, $u'v'^\omega \in M$. Since $\preccurlyeq^{\mathcal{B}} \subseteq \lesssim^{\mathcal{B}} \subseteq \leqslant^{\mathcal{B}}$ and $\leqslant^{\mathcal{B}}, \preccurlyeq^{\mathcal{B}}$ is $L(\mathcal{B})$-preserving we deduce that $\lesssim^{\mathcal{B}}, \preccurlyeq^{\mathcal{B}}$ and $\preccurlyeq^{\mathcal{B}}, \preccurlyeq^{\mathcal{B}}$ are also $L(\mathcal{B})$-preserving. □

Remark 7. The supergraphs of [3, Def. 6] endowed with their subsumption orders coincide with our qo $\preccurlyeq^{\mathcal{B}}$. Without the subsumption order they coincide with $\preccurlyeq^{\mathcal{B}} \cap \preccurlyeq^{\mathcal{B}^{-1}}$.

Example 13 shows a run of Algorithm 3 instantiated with the pair $\leqslant^{\mathcal{D}}, \preccurlyeq^{\mathcal{D}}$ on the BA $\mathcal{C}$ and $\mathcal{D}$ depicted in Figure 3.1. Example 14 shows the benefits of using the coarser pair $\leqslant^{\mathcal{B}}, \preccurlyeq^{\mathcal{B}}$ of state-based quasiorders.

Example 13 (Run of Algorithm 3)**.** *We consider the BA of Figure 3.1 for which we want to check whether $L(\mathcal{C}) \subseteq L(\mathcal{D})$ holds using the pair $\leqslant^{\mathcal{D}}, \preccurlyeq^{\mathcal{D}}$ of state-based quasiorders. From Example 12 we have that $f_{\mathcal{C}}(\emptyset) = \{\epsilon\}$, ${f_{\mathcal{C}}}^2(\emptyset) = \{\epsilon, a, b\}$ and ${f_{\mathcal{C}}}^3(\emptyset) = \{\epsilon, a, b, aa, ab, ba, bb\}$. From Example 9, for $u \in \{aa, ba\}$ and $v \in \{ab, bb\}$, we have that $a \leqslant^{\mathcal{D}} u$ and $b \leqslant^{\mathcal{D}} v$, while*

a and ϵ are incomparable for $\leqslant^{\mathcal{D}}$. Hence, ${f_{\mathcal{C}}}^3(\emptyset) \sqsubseteq_{\leqslant^{\mathcal{D}}} {f_{\mathcal{C}}}^2(\emptyset)$ so that a finite basis of $\operatorname{lfp} f_{\mathcal{C}}$ is achieved by ${f_{\mathcal{C}}}^2(\emptyset)$. Similarly we have ${r_{\mathcal{C}}}^2(\emptyset) \sqsubseteq_{\preccurlyeq^{\mathcal{D}}} r_{\mathcal{C}}(\emptyset)$. Thus, the membership check is performed on the elements of ${f_{\mathcal{C}}}^2(\emptyset) \times r_{\mathcal{C}}(\emptyset) = \{\epsilon, a, b\} \times \{a, b\}$, and for $(a, b) \in {f_{\mathcal{C}}}^2(\emptyset) \times r_{\mathcal{C}}(\emptyset)$, the word ab^ω is a witness that $L^\omega(\mathcal{C}) \not\subseteq L^\omega(\mathcal{D})$.

Example 14 (The Coarser the Better)**.** *The family of inclusion problems $L^\omega(\mathcal{A}_n) \subseteq L^\omega(\mathcal{B}_n)$ in Figure 4.1 shows the benefits of using the coarser pair $\leqslant^{\mathcal{B}}, \preccurlyeq^{\mathcal{B}}$ of state-based quasiorders. From Example 10 in Section 4.4.1 we deduce that a basis for L_{i_n,p_n} w.r.t. $\leqslant^{\mathcal{B}_n}$ has size one and likewise, a basis for $L^*(\mathcal{A}_{n\,p_n}^{\,p_n})\backslash\{\epsilon\} = b^+$ w.r.t. $\preccurlyeq^{\mathcal{B}_n}$. Thus, using the pair of quasiorders $\leqslant^{\mathcal{B}_n}, \preccurlyeq^{\mathcal{B}_n}$, a single membership query (i.e., $u \in L^\omega(\mathcal{B}_n)$) is needed to decide the inclusion $L^\omega(\mathcal{A}_n) \subseteq L^\omega(\mathcal{B}_n)$, as opposed to no less than $\frac{n(n+1)}{2}$ membership queries for the other pairs of Proposition 7.*

In the work by Parolini [70], various suitable pairs of quasiorders are defined based on simulation relations on the states of the BA. These simulation-based quasiorders are coarser than the state-based pairs introduced in this section, while still being finer than the syntactic pair, which will be discussed next.

5.3.2 A Syntactic Pair

Given a ω-regular language $M \subseteq \Sigma^\omega$ we define the following syntactic quasiorders:

$$u \leqslant_\omega^M v \overset{\triangle}{\iff} \forall \xi \in \Sigma^\omega, u\xi \in M \implies v\xi \in M \ ,$$
$$u \preccurlyeq^M v \overset{\triangle}{\iff} u \leqslant_\omega^M v \wedge (\forall s, t \in \Sigma^*, s(ut)^\omega \in M \implies s(vt)^\omega \in M) \ .$$

Deciding these syntactic quasiorders is as hard as the inclusion problem between ω-regular languages. Nevertheless, the syntactic quasiorders act as a gold standard for quasiorders in the sense formalized in the following proposition.

Proposition 8. *If M is a ω-regular language then $\leqslant_\omega^M, \preccurlyeq^M$ is M-suitable. Moreover, if $\leqslant, \preccurlyeq$ is a pair of M-suitable quasiorders such that $\preccurlyeq \subseteq \leqslant$ then $\leqslant \subseteq \leqslant_\omega^M$ and $\preccurlyeq \subseteq \preccurlyeq^M$.*

Proof. Let $\leqslant, \preccurlyeq$ be a pair of M-suitable quasiorders such that $\preccurlyeq \subseteq \leqslant$. First we show that $\leqslant \subseteq \leqslant_\omega^M$ and $\preccurlyeq \subseteq \preccurlyeq^M$. Let $u \leqslant u'$. Since M is ω-regular for every $w \in \Sigma^*$ the language $w^{-1}M$ is ω-regular. Thus, by Theorem 5 to show that $u \leqslant_\omega^M u'$ it suffices to show that $\forall st^\omega \in u^{-1}M, st^\omega \in {u'}^{-1}M$. Let $st^\omega \in u^{-1}M$. Since $\leqslant$ is right-monotonic from $u \leqslant u'$ we deduce $us \leqslant u's$. Since $ust^\omega \in M$, $us \leqslant u's$, $t \preccurlyeq t$ and $\leqslant, \preccurlyeq$ is M-preserving we have $u'st^\omega \in M$, thus $st^\omega \in {u'}^{-1}M$. Hence, $u \leqslant_\omega^M u'$. Let $u \preccurlyeq u'$. Since $\preccurlyeq \subseteq \leqslant \subseteq \leqslant_\omega^M$ we have $u \leqslant_\omega^M u'$. Let $(s, t) \in s, t \in \Sigma^*$ such that $s(ut)^\omega \in M$. Since $\preccurlyeq$ is right-monotonic from $u \preccurlyeq u'$ we deduce $ut \preccurlyeq u't$. Since $s(ut)^\omega \in M$, $s \leqslant s$, $ut \preccurlyeq u't$ and $\leqslant, \preccurlyeq$ is M-preserving we have $s(u't)^\omega \in M$, thus $u \preccurlyeq^M u'$.

Next we show that $\leqslant_\omega^M, \preccurlyeq^M$ is M-suitable. Let $\mathcal{B}$ be a BA accepting M. Since $\lesssim^{\mathcal{B}}, \preccurlyeq^{\mathcal{B}}$ is M-suitable by what was previously shown we have $\lesssim^{\mathcal{B}} \subseteq \leqslant_\omega^M$ and $\preccurlyeq^{\mathcal{B}} \subseteq \preccurlyeq^M$ and since $\lesssim^{\mathcal{B}}$ and $\preccurlyeq^{\mathcal{B}}$ are well-quasiorders (Proposition 7) we deduce that $\leqslant_\omega^M$ and $\preccurlyeq^M$ are well-quasiorders.

Let $uv^\omega \in M$, $u \leqslant_\omega^M u'$ and $v \preccurlyeq^M v'$. Form $uv^\omega \in M$ and $u \leqslant_\omega^M u'$ we deduce $u'v^\omega \in M$. From $v \preccurlyeq^M v'$ and $u'(v\epsilon)^\omega \in M$ we deduce $u'v'^\omega \in M$. Thus, $\leqslant_\omega^M, \preccurlyeq^M$ is M-preserving.

It is an easy exercise to show that $\leqslant_\omega^M$ and $\preccurlyeq^M$ are right-monotonic and that given a BA for M deciding the quasiorders $\leqslant_\omega^M$ and $\preccurlyeq^M$ reduces to deciding an inclusion between two BA derived from the BA for M. □

5.4 State-based Algorithm

In this section we instantiate Algorithm 3 with the pair $\leqslant^{\mathcal{B}}, \preccurlyeq^{\mathcal{B}}$ of state-based quasiorders of a BA $\mathcal{B} = (\hat{Q}, \hat{q}_I, \hat{\delta}, \hat{F})$ accepting M. From this instantiation, we derive a state-based inclusion algorithm deciding $L^\omega(\mathcal{A}) \subseteq L^\omega(\mathcal{B})$.

Data Structures. Thanks to the equalities (4.1), (4.2) and (4.3) in Section 4.5.1 the $post^{\mathcal{B}}$, $ctx^{\mathcal{B}}$ and $ctx^{\mathcal{B}}_{\circledast}$ of newly added words in the functions $f_{\mathcal{A}}$ and $r_{\mathcal{A},p}$ can be inductively computed. Hence, the fixpoint computations at line 1 and line 3 of Algorithm 3 instantiated with the state-based pair of quasiorders $\leqslant^{\mathcal{B}}, \preccurlyeq^{\mathcal{B}}$ can be implemented by manipulating directly subsets of $\wp(\hat{Q})$ (for the prefixes) and pairs of subsets of $\wp(\hat{Q}^2)$ (for the periods). Next, we provide a detailed definition of this implementation.

5.4.1 Fixpoint Computation

In Section 4.5.2 we defined the state-based version $\hat{f}_{\mathcal{A}} : \wp(\wp(\hat{Q}))^{|Q|} \to \wp(\wp(\hat{Q}))^{|Q|}$ of $f_{\mathcal{A}}$ on the complete lattice $(\wp(\wp(\hat{Q}))^{|Q|}, \subseteq \times \cdots \times \subseteq)$[2]. Next, we do the analogue with the functions $r_{\mathcal{A},p}$ i.e., we define functions $\hat{r}_{\mathcal{A},p} : \wp(\wp(\hat{Q}^2) \times \wp(\hat{Q}^2))^{|Q|} \to \wp(\wp(\hat{Q}^2) \times \wp(\hat{Q}^2))^{|Q|}$ such that for every $X \in \wp(\Sigma^*)^Q$ we have

$$\hat{r}_{\mathcal{A},p}(\langle\{(ctx^{\mathcal{B}}(u), ctx^{\mathcal{B}}_{\circledast}(u)) \mid u \in X_q\}\rangle_{q\in Q}) = \langle\{(ctx^{\mathcal{B}}(y), ctx^{\mathcal{B}}_{\circledast}(y)) \mid y \in (r_{\mathcal{A},p}(X))_q\}\rangle_{q\in Q} \ .$$

To do so, we work with the complete lattice $(\wp(\wp(\hat{Q}^2) \times \wp(\hat{Q}^2))^{|Q|}, \subseteq^2 \times \cdots \times \subseteq^2)$ and $\subseteq^2 \triangleq \subseteq \times \subseteq$ and given $X \in \wp(\wp(\hat{Q}^2) \times \wp(\hat{Q}^2))^Q$, we define $\mathrm{Post}^{\preccurlyeq^{\mathcal{B}}}_{\mathcal{A}}(X) \in \wp(\wp(\hat{Q}^2) \times \wp(\hat{Q}^2))^Q$ by

$$\mathrm{Post}^{\preccurlyeq^{\mathcal{B}}}_{\mathcal{A}}(X) \triangleq \langle \bigcup_{a\in\Sigma, q\in\delta(q',a)} \{\Big(x_1 \circ ctx^{\mathcal{B}}(a), x_1 \circ ctx^{\mathcal{B}}_{\circledast}(a) \cup x_2 \circ ctx^{\mathcal{B}}(a)\Big) \mid (x_1, x_2) \in X_{q'}\}\rangle_{q\in Q} \ ,$$

where, given two binary relations $x, x' \in \wp(\hat{Q}^2)$ on states of $\mathcal{B}$, the notation $x \circ x'$ denotes their composition.

In turn for every $p \in F$ we define the maps

$$\hat{r}_{\mathcal{A},p} = \lambda X.\langle\{(ctx^{\mathcal{B}}(a), ctx^{\mathcal{B}}_{\circledast}(a)) \mid q \in \delta_{\mathcal{A}}(p, a)\} \cup \ (\mathrm{Post}^{\preccurlyeq^{\mathcal{B}}}_{\mathcal{A}}(X))_{q\in Q}\rangle_{q\in Q} \ .$$

We check convergence of the fixpoint computations (lines 1–3 of Algorithm 3) by reasoning on the functions $\hat{f}_{\mathcal{A}}$ and $\hat{r}_{\mathcal{A},p}$ and by using the quasiorders $\sqsubseteq_{\subseteq}$ on $\wp(\wp(\hat{Q}))$ for prefixes and $\sqsubseteq_{\subseteq\times\subseteq}$ on $\wp(\wp(\hat{Q}^2) \times \wp(\hat{Q}^2))$ for periods. Incidentally, as we show below, we can perform the membership checks of line 5 (asking whether $uv^\omega \in L^\omega(\mathcal{B})$ given u and v) using the $post^{\mathcal{B}}$ associated to the prefix u and $ctx^{\mathcal{B}}$-$ctx^{\mathcal{B}}_{\circledast}$ associated to the period v and nothing else.

[2] each Cartesian product consists of $|Q|$ factors

5.4.2 Membership Check

To decide membership in $L^\omega(\mathcal{B})$ we use the membership predicate $\mathrm{Inc}^{\mathcal{B}}$ defined for $x \in \wp(\hat{Q})$ and $y_1, y_2 \in \wp(\hat{Q}^2)$ as follows:

$$\mathrm{Inc}^{\mathcal{B}}(x, y_1, y_2) \triangleq \exists q, p \in \hat{Q}, q \in x \wedge (q,p) \in y_1^* \wedge (p,p) \in y_1^* \circ y_2 \circ y_1^* \ ,$$

where, given two binary relations $y, y' \in \wp(\hat{Q}^2)$ on states of $\mathcal{B}$, the notation $y \circ y'$ denotes their composition, and y^* denotes the Kleene closure of y.

Proposition 9. *For all* $(u,v) \in \Sigma^* \times \Sigma^+$, $Inc^{\mathcal{B}}(post^{\mathcal{B}}(u), ctx^{\mathcal{B}}(v), ctx^{\mathcal{B}}_{\circledast}(v)) \iff uv^\omega \in L^\omega(\mathcal{B})$,

Proof. Let $(u,v) \in \Sigma^* \times \Sigma^+$. We have $(p,q) \in \mathrm{ctx}^{\mathcal{B}}(v)^* \iff \exists n, (p,q) \in \mathrm{ctx}^{\mathcal{B}}(v^n)$. Therefore, $\mathrm{Inc}^{\mathcal{B}}(\mathrm{post}^{\mathcal{B}}(u), \mathrm{ctx}^{\mathcal{B}}(v), \mathrm{ctx}^{\mathcal{B}}_{\circledast}(v))$ holds iff there are $q, p \in \hat{Q}$ and two positive integers n, m such that $q \in \mathrm{post}^{\mathcal{B}}(u)$, $(q,p) \in \mathrm{ctx}^{\mathcal{B}}(v^n)$ and $(p,p) \in \mathrm{ctx}^{\mathcal{B}}_{\circledast}(v^m)$. Thus, $\mathrm{Inc}^{\mathcal{B}}(\mathrm{post}^{\mathcal{B}}(u), \mathrm{ctx}^{\mathcal{B}}(v), \mathrm{ctx}^{\mathcal{B}}_{\circledast}(v))$ holds iff there is an accepting run of $\mathcal{B}$ on uv^ω of the form $\hat{q_I} \xrightarrow{u}{}^* q \xrightarrow{v^n}{}^* p \xrightarrow{v^m}{}_F{}^* p$.

□

5.4.3 Algorithm and Complexity

Our state-based algorithm is Algorithm 4 below.

Algorithm 4: Algorithm for deciding $L^\omega(\mathcal{A}) \subseteq L^\omega(\mathcal{B})$

Data: Büchi automata $\mathcal{A} = (Q, q_I, \delta, F)$ and $\mathcal{B} = (\hat{Q}, \hat{q}_I, \hat{\delta}, \hat{F})$

1 Compute $\hat{f}^m_{\mathcal{A}}(\vec{\emptyset})$ with least m s.t. $\hat{f}^{m+1}_{\mathcal{A}}(\vec{\emptyset}) \sqsubseteq^{|Q|}_{\subseteq} \hat{f}^m_{\mathcal{A}}(\vec{\emptyset})$;
2 **foreach** $p \in F$ **do**
3 Compute $\hat{r}^{m'}_{\mathcal{A},p}(\vec{\emptyset})$ with least m' s.t. $\hat{r}^{m'+1}_{\mathcal{A},p}(\vec{\emptyset}) \sqsubseteq^{|Q|}_{\subseteq\times\subseteq} \hat{r}^{m'}_{\mathcal{A},p}(\vec{\emptyset})$;
4 **foreach** $x \in (\hat{f}^m_{\mathcal{A}}(\vec{\emptyset}))_p$, $(y_1, y_2) \in (\hat{r}^{m'}_{\mathcal{A},p}(\vec{\emptyset}))_p$ **do**
5 **if** $\neg\mathrm{Inc}^{\mathcal{B}}(x, (y_1, y_2))$ **then return false**;
6 **return true**;

Theorem 8. *Given the required inputs, Algorithm 4 decides the inclusion problem* $L^\omega(\mathcal{A}) \subseteq L^\omega(\mathcal{B})$.

Proof. Follows from Theorem 6. □

Proposition 10. *Let* $n \triangleq |Q|$ *and* $\hat{n} \triangleq |\hat{Q}|$. *The running time of the state-based algorithm is* $|\Sigma| \cdot n^3 \cdot 2^{O(\hat{n}^2)}$.

Proof. By similar arguments as in the proof of Proposition 5 we find that an upper bound on the runtime of lines 1 and 3 is $|\Sigma| \cdot n^2 \cdot 2^{O(\hat{n}^2)}$. The runtime for the state-based membership query at line 5 is upper bounded by $2^{O(\hat{n}^2)}$. Finally, the number of iterations of the loops of lines 2 and 4 is n and $2^{\hat{n}} \cdot 2^{2\hat{n}^2}$ respectively. Since these loops are nested, we multiply these bounds to end up with an upper bound $|\Sigma| \cdot n^3 \cdot 2^{O(\hat{n}^2)}$ on the runtime of Algorithm 4. □

We implemented Algorithm 4 with the antichain optimization in a tool called BAIT[9]. In Section 6.5, we provide more details on BAIT and present our experiments.

We conclude this section with an illustrative example of a run of Algorithm 4.

5.4.4 Illustrative Example

We show the execution of a run of Algorithm 4 obtained on the BA $\mathcal{C}$ and $\mathcal{D}$ depicted in Figure 3.1. As a result, the algorithm will correctly decide that $L^\omega(\mathcal{C})$ is not included in $L^\omega(\mathcal{D})$ (e.g., $ab^\omega \in L^\omega(\mathcal{C})$ but $ab^\omega \notin L^\omega(\mathcal{D})$).

As shown in Section 4.5.3 the fixpoint computation at line 1 computes $\hat{f}^2_{\mathcal{C}}(\emptyset) = \{\{q_0\}, \{q\}\}$. Next, the algorithm evaluates the sequence $\{\hat{r}^n_{\mathcal{C}}(\emptyset)\}_{n\in\mathbb{N}}$ where $\hat{r}^n_{\mathcal{C}}(\emptyset) \in \wp(\wp(\{q_0, q\}^2) \times \wp(\{q_0, q\}^2))$ for every $n \in \mathbb{N}$ and $\hat{r}_{\mathcal{C}}$ is defined for $X \in \wp(\wp(\{q_0, q\}^2) \times \wp(\{q_0, q\}^2))$ by

$$\hat{r}_{\mathcal{C}}(X) = \{(y, y), (z_1, z_2)\} \cup \mathrm{Post}^{\preccurlyeq^{\mathcal{D}}}_{\mathcal{C}}(X)$$

where, $y \triangleq \{(q_0, q), (q, q)\}$, $z_1 \triangleq \{(q_0, q_0), (q, q_0)\}$, $z_2 \triangleq \{(q, q_0)\}$ and

$$\mathrm{Post}^{\preccurlyeq^{\mathcal{D}}}_{\mathcal{C}}(X) \triangleq \{(p_1 \circ ctx^{\mathcal{D}}(c), p_1 \circ \mathrm{ctx}^{\mathcal{D}}_{\circledast}(c) \cup p_2 \circ ctx^{\mathcal{D}}(c)) \mid c \in \{a, b\} \wedge (p_1, p_2) \in X\} \ .$$

Note that $y = ctx^{\mathcal{D}}(a) = \mathrm{ctx}^{\mathcal{D}}_{\circledast}(a)$, $z_1 = ctx^{\mathcal{D}}(b)$ and $z_2 = \mathrm{ctx}^{\mathcal{D}}_{\circledast}(b)$. We then have:

- $\hat{r}^1_{\mathcal{C}}(\emptyset) = \{(y, y), (z_1, z_2)\}$.
- $\hat{r}^2_{\mathcal{C}}(\emptyset) = \{(y, y), (z_1, z_2), (z_1, z_1)\}$.

Since $(z_1, z_2) \subseteq^2 (z_1, z_1)$ we have $\hat{r}^2_{\mathcal{C}}(\emptyset) \sqsubseteq_{\subseteq\times\subseteq} \hat{r}^1_{\mathcal{C}}(\emptyset)$. Thus, the computations for the period iterates stop at the second iteration. It turns out that $\neg\mathrm{Inc}^{\mathcal{D}}(\{q\}, (z_1, z_2))$: this, intuitively, corresponds to the counterexample ab^ω that belongs to $L^\omega(\mathcal{C})$ but not $L^\omega(\mathcal{D})$. Hence, the inclusion $L^\omega(\mathcal{C}) \subseteq L^\omega(\mathcal{D})$ does not hold.

5.5 Extension: ω-context free $\subseteq$ ω-regular

In this section, we extend the framework presented previously in this chapter to address the inclusion problem $L^\omega(\mathcal{P}) \subseteq M$, where $\mathcal{P} = (Q, q_I, \Gamma, \delta, F)$ is a BPDA and M is a ω-regular language.

First, we observe that ultimately periodic words suffice once again for the inclusion problem.

Theorem 9. *Let $\mathcal{P}$ be a BPDA and M a ω-regular language we have $L^\omega(\mathcal{P}) \subseteq M \iff \forall uv^\omega \in L^\omega(\mathcal{P}), uv^\omega \in M$.*

Proof. Suppose $\forall uv^\omega \in L^\omega(\mathcal{P}), uv^\omega \in M$ and $L^\omega(\mathcal{P}) \cap (\Sigma^\omega \backslash M)$ non empty. The language $L^\omega(\mathcal{P}) \cap (\Sigma^\omega \backslash M)$ is an ω-context free language [19] and since it is not empty one can show that it contains an ultimately periodic word. We thus obtain a contradiction. □

Next, we want to derive a finite subset $S_{\text{finite}} \subseteq \Sigma^* \times \Sigma^+$ of decompositions of ultimately periodic words of $L^\omega(\mathcal{P})$ such that

$$L^\omega(\mathcal{P}) \subseteq M \iff \forall (u, v) \in S_{\text{finite}},\ uv^\omega \in M \ . \qquad (\dagger\dagger^{cf})$$

To achieve this, the idea remains the same as before. We first define a sufficient subset of decompositions of ultimately periodic words of $L^\omega(\mathcal{P})$ and then, given a pair of M-preserving well-quasiorders $\leqslant, \preccurlyeq$ we derive S_{finite} as a finite basis for this subset w.r.t. $\leqslant \times \preccurlyeq$.

5.5.1 A Sufficient Subset of Decompositions

For every pair of configurations $(q,\alpha),(p,\beta) \in Q \times \Gamma$ we define the following languages of finite words:

$$L_{(q,\alpha),(p,\beta)} \triangleq \{u \in \Sigma^* \mid (q,\alpha) \vdash^u (p,\beta w) \text{ for some } w \in \Gamma^*\} \ ,$$
$$L^{\circledast}_{(q,\alpha),(p,\beta)} \triangleq \{u \in \Sigma^* \mid (q,\alpha) \vdash^{\circledast u} (p,\beta w) \text{ for some } w \in \Gamma^*\} \ .$$

In turn, we define $S^{cf} \triangleq \bigcup_{(q,\gamma)\in Q\times\Gamma} L_{(q_I,\bot),(q,\gamma)} \times (L^{\circledast}_{(q,\gamma),(q,\gamma)} \backslash \{\epsilon\})$ which as shown by the next lemma satisfies Equation $(\dagger\dagger^{cf})$.

Lemma 4. $L^\omega(\mathcal{P}) \subseteq M \iff \forall (u,v) \in S^{cf},\ uv^\omega \in M$.

Proof. It suffices to show that for every $(u,v) \in \Sigma^* \times \Sigma^+$ we have $uv^\omega \in L^\omega(\mathcal{P}) \iff \exists (u',v') \in S^{cf},\ uv^\omega = u'v'^\omega$. The direction $\Leftarrow$ is straightforward. For the reverse direction let $uv^\omega \in L^\omega(\mathcal{P})$ and consider an accepting run $e:\ (q_I,\bot) \vdash^{a_1} (q_1,\alpha_1) \vdash^{a_2} (q_2,\alpha_2) \vdash^{a_3} \dots$ where $uv^\omega = a_1a_2\dots$ and for every $n \geq 1$, $\alpha_n = \gamma_n w_n$ for some $\gamma_n \in \Gamma$ and $w_n \in \Gamma^*$. First we observe that there is a subsequence $\{(q_{s_n},\alpha_{s_n})\}_{n\in\mathbb{N}}$ of configurations of e such that

1. $\forall n,\ (q_{s_n},\alpha_{s_n}) = (q,\gamma w_n)$ for some $(q,\gamma) \in Q \times \Gamma$,
2. $\forall n, \forall m \geq s_n,\ |\alpha_{s_n}| \leq |\alpha_m|$.

Since e is accepting, we can assume that every fragment

$$(q_{s_n},\alpha_{s_n}) \vdash^{* a_{s_n+1}\dots a_{s_{n+1}}} (q_{s_{n+1}},\alpha_{s_{n+1}})$$

of e includes a configuration whose state is final i.e.,

$$(q_{s_n},\alpha_{s_n}) \vdash^{\circledast a_{s_n+1}\dots a_{s_{n+1}}} (q_{s_{n+1}},\alpha_{s_{n+1}}) \ .$$

Let $\Delta_v = \{v[i] \mid i = 1,\dots,|v|\} \cup \{\epsilon\}$ be the alphabet where each letter $v[i]$ of v is seen as a distinct symbol. We can assume that $a_{s_0}a_{s_0+1}\dots$ only contains letters in Δ_v, that every fragment $(q_{s_n},\alpha_{s_n}) \vdash^{\circledast a_{s_n+1}\dots a_{s_{n+1}}} (q_{s_{n+1}},\alpha_{s_{n+1}})$ always starts with the same letter and that $a_{s_n+1}\dots a_{s_{n+1}}$ contains all letters in $\Delta_v\backslash\{\epsilon\}$. Hence, for $u' = a_1\dots a_{s_0}$ and $v' = a_{s_0+1}\dots a_{s_{n+1}}$ we have $uv^\omega = u'v'^\omega$, $u' \in L_{(q_I,\bot),(q,\gamma)}$ and thanks to item 2, $v' \in L^{\circledast}_{(q,\gamma),(q,\gamma)}\backslash\{\epsilon\}$. Thus, $(u',v') \in S^{cf}$. □

5.5.2 Fixpoint Computation of a Finite Basis

First, we establish a characterization of the subset S^{cf}. As for the ω-regular case, we show that S^{cf} can be described using least fixpoint operators.

The languages $L_{(q_I,\bot),(q,\gamma)}$ and $(L^{\circledast}_{(q,\gamma),(q,\gamma)}\backslash\{\epsilon\})$ defining S^{cf} are context-free, and thus, they can be recognized by CFGs. Consequently, as demonstrated by the first part of the proposition

below, there exist two functions $f_{(q,\gamma)}$ and $r_{(q,\gamma)}$ over vectors of sets of words, such that $L_{(q_I,\bot),(q,\gamma)} = (\mathrm{lfp}\, f_{(q,\gamma)})_1$ and $L^{\circledast}_{(q,\gamma),(q,\gamma)} \backslash \{\epsilon\} = (\mathrm{lfp}\, r_{(q,\gamma)})_1$. We thus have

$$S^{cf} = \bigcup_{(q,\gamma)\in Q\times\Gamma} (\mathrm{lfp}\, f_{(q,\gamma)})_1 \times (\mathrm{lfp}\, r_{(q,\gamma)})_1 \ .$$

Proposition 11. *Let $\mathcal{G}$ be a CFG and k the number of its variables.*

1. *$\mathcal{G}$ induces an increasing function $f : \wp(\Sigma^*)^k \to \wp(\Sigma^*)^k$ such that $L(\mathcal{G}) = (\mathrm{lfp}\, f)_1$. Moreover, every Kleene iterate of f is computable.*
2. *If $\ltimes$ is a well-quasiorder on $\wp(\Sigma^*)$ then there is a positive integer n such that $f^{n+1}(\vec{\emptyset}) \sqsubseteq^k_{\ltimes} f^n(\vec{\emptyset})$; and, if $\ltimes$ is monotonic then $\mathrm{lfp}\, f \sqsubseteq^k_{\ltimes} f^n(\vec{\emptyset})$.*

Proof. 1. We define the function f over $\wp(\Sigma^*)^k$ i.e., over the k-dimensional vectors of sets of words, where k is the number of variables of our grammar $\mathcal{G}$. Let $Y = (Y_1, \ldots, Y_k) \in \wp(\Sigma^*)^k$. For each $j \in [1,k]$, the j-th component of $f(Y)$ is defined as

$$(f(Y))_j \triangleq \bigcup_{X_j \to X_k X_{k'} \in P} Y_k Y_{k'} \cup \bigcup_{a\in\Sigma\cup\{\epsilon\},\ X_j\to a\in P} a$$

Notice that f is increasing in the complete lattice of set $\wp(\Sigma^*)^k$ equipped with the cartesian product of k set inclusions $\subseteq \times \cdots \times \subseteq$. Therefore, by the Knaster–Tarski theorem $\mathrm{lfp}\, f = (\{u \in \Sigma^* \mid X_j \to^* u\})_{j\in[1,k]}$. Thus, $L(\mathcal{G}) = (\mathrm{lfp}\, f)_1$.

2. Analogue to the proof of Proposition 2 and by adapting Lemma 1.

□

By Proposition 11 each Kleene iterate of $f_{(q,\gamma)}$ and $r_{(q,\gamma)}$ is computable. Furthermore, when we have a pair of M-suitable quasiorders $\leqslant, \preccurlyeq$, and both of these quasiorders are monotonic, we can compute a finite basis for $\mathrm{lfp}\, f_{(q,\gamma)}$ and $\mathrm{lfp}\, r_{(q,\gamma)}$. Thus, we can compute a finite basis for S^{cf} w.r.t. $\leqslant \times \preccurlyeq$.

Quasiorders for the Context-Free Case

Given a BA $\mathcal{B}$ representing M the pairs $\lesssim^{\mathcal{B}}, \preccurlyeq^{\mathcal{B}}$ and $\preccurlyeq^{\mathcal{B}}, \preccurlyeq^{\mathcal{B}}$ are M-suitable as shown by Proposition 7. Furthermore, by Proposition 3 the quasiorders $\lesssim^{\mathcal{B}}$ and $\preccurlyeq^{\mathcal{B}}$ are monotonic. Thus, we can use them in our framework to decide the inclusion $L^{\omega}(\mathcal{P}) \subseteq M$. Just as in Section 5.4, we can employ these quasiorders to formulate a purely "state-based" algorithm deciding $L^{\omega}(\mathcal{P}) \subseteq M$.

Chapter 6

FORQ-BASED INCLUSION

In this chapter we focus once again on the inclusion problem $L^\omega(\mathcal{A}) \subseteq M$, where $\mathcal{A}$ is a BA and M is a ω-regular language. The difference with respect to the previous chapter is that now we use an unbounded number of quasiorders to filter the ultimately periodic words of $L^\omega(\mathcal{A})$. The motivation for doing so is to obtain more pruning when searching for a counterexample to inclusion.

6.1 Foundations

We consider the inclusion problem $L^\omega(\mathcal{A}) \subseteq M$, where $\mathcal{A} = (Q, q_I, \delta, F)$ is a BA and M is a ω-regular language.

To tackle this problem, our goal is to derive a finite subset $T_{\text{finite}} \subseteq \Sigma^* \times \Sigma^+$ of decompositions of ultimately periodic words from $L^\omega(\mathcal{A})$ that should satisfy Equivalence (††), as shown below.

$$L^\omega(\mathcal{A}) \subseteq M \iff \forall (u,v) \in T_{\text{finite}},\ uv^\omega \in M \ . \tag{††}$$

The difference with the approach in Section 5.1, is that now we will use a *family of right quasiorders* to derive T_{finite}, a notion introduced next.

Definition 6.1.1. A Family of Right Quasiorders (FORQ) is a pair $\langle \leqslant, \{\preccurlyeq_u\}_{u\in\Sigma^*}\rangle$ where $\leqslant$ is a right-monotonic quasiorder on Σ^* as well as every $\preccurlyeq_u$ where $u \in \Sigma^*$. Additionally, we require the following constraint, called the FORQ-constraint: $\forall u, u' \in \Sigma^*,\ u \leqslant u' \Rightarrow \preccurlyeq_{u'} \subseteq \preccurlyeq_u$.

Given a FORQ $\langle \leqslant, \{\preccurlyeq_u\}_{u\in\Sigma^*}\rangle$ the quasiorder $\leqslant$ is used to compare the prefixes of ultimately periodic words and the quasiorders $\preccurlyeq_u$, where u is a prefix, to compare their periods. Observe that the ordering used for the periods depends on the prefixes so that a period may or may not be discarded depending on the prefix under consideration. The FORQ constraint tells us that if the periods v and w compare for a prefix u', that is $v \preccurlyeq_{u'} w$, then they also compare for every prefix u subsuming u', that is $v \preccurlyeq_u w$ if $u \leqslant u'$.

The property of right-monotonicity of the quasiorders of a FORQ is needed to iteratively compute T_{finite} via the fixpoint characterizations of Section 5.2.1. Next we adapt the notion of suitability (Definition 5.1.1) to a family of quasiorders. It is worth pointing out that for

a FORQ to be suitable the inverse of the order on the prefixes needs also to satisfy the "well"-property.

Definition 6.1.2. We say that a FORQ $\langle \leqslant, \{\preccurlyeq_u\}_{u\in\Sigma^*}\rangle$ is *M-preserving* when $\forall u, \hat{u} \in \Sigma^*, v, \hat{v} \in \Sigma^+$ if $uv^\omega \in L$, $u \leqslant \hat{u}$, $v \preccurlyeq_{\hat{u}} \hat{v}$ and $\hat{u}\,\hat{v} \leqslant \hat{u}$ then $\hat{u}\hat{v}^\omega \in L$. A FORQ $\langle \leqslant, \{\preccurlyeq_u\}_{u\in\Sigma^*}\rangle$ is said to be *M-suitable* when it is M-preserving and when $\leqslant$, its inverse $\leqslant^{-1}$, and $\preccurlyeq_u$ for every $u \in \Sigma^*$ are all decidable well-quasiorders.

In the following, we derive T_{finite} using a M-suitable FORQ $\langle \leqslant, \{\preccurlyeq_u\}_{u\in\Sigma^*}\rangle$. To achieve this, we rely on some definitions and results from Section 5.1. Recalling that for any pair of states $p, q \in Q$ the language $L_{p,q} = \{u \in \Sigma^* \mid p \xrightarrow{u}{}^* q\}$ contains all the words that can lead from state p to q in $\mathcal{A}$, and that $S = \bigcup_{p\in F} L_{q_I,p} \times (L_{p,p}\backslash\{\epsilon\})$ is a subset of decompositions of ultimately periodic words in $L^\omega(\mathcal{A})$ that satisfies Equivalence (††) (as per Lemma 3), we proceed to derive T_{finite} as a subset of S. To do this, for each $p \in F$ we fix a basis U_p for $L_{q_I,p}$ w.r.t. $\leqslant$, a basis W_p for $L_{q_I,p}$ w.r.t. $\leqslant^{-1}$, and a basis V_p^w for $L_{p,p}\backslash\{\epsilon\}$ w.r.t. $\preccurlyeq_w$ for every $w \in W_p$. In turn we define T_{finite} as follows.

$$T_{\text{finite}} = \bigcup_{p\in F} \left\{(u, v) \in \Sigma^* \times \Sigma^+ \mid u \in U_p, v \in V_p^w \text{ for some } w \in W_p \text{ with } u \leqslant w\right\} . \qquad (‡)$$

Note that the quasiorder $\preccurlyeq_w$ used to prune the periods of $L_{p,p}$ depends on a maximal w.r.t. $\leqslant$ prefix w of $L_{q_I,p}$ since w belongs to the basis W_p for $\leqslant^{-1}$. The correctness argument for choosing $\preccurlyeq_w$ essentially relies on the FORQ constraint as the proof of Proposition 12 shows. In Section 6.3 (Example 18) we will show, that when w is not "maximal" the quasiorder $\preccurlyeq_w$ yields a set T_{finite} for which (††) does not hold.

Next we prove that T_{finite} satisfies the equivalence (††). The proof crucially relies on the preservation property of the FORQ which allows discarding candidate counterexamples without loosing completeness, that is, if inclusion does not hold a counterexample will be returned.

Proposition 12. *The subset T_{finite} satisfies Equivalence* (††).

Proof. We show that $uv^\omega \in L^\omega(\mathcal{A}) \iff \exists (u', v') \in T_{\text{finite}},\ uv^\omega = u'v'^\omega$. The direction $\Leftarrow$ holds because T_{finite} is a subset of decompositions of ultimately periodic words of $L^\omega(\mathcal{A})$. For the direction $\Rightarrow$ we first show that for every $(u, v) \in S = \bigcup_{p\in F} L_{q_I,p} \times (L_{p,p}\backslash\{\epsilon\})$ there is $(u', v') \in S$ such that $u'v' \leqslant v'$ and $u'v'^\omega = uv^\omega$. Let $u \in L_{q_I,p}$ and $v \in L_{p,p}$. If $uv \leqslant u$ then we are done for otherwise consider the sequence $\{uv^i\}_{i\in\mathbb{N}}$. Since $\leqslant^{-1}$ is a well-quasiorder, there exists $x, y \in \mathbb{N}$ such that $x < y$ and $uv^x \leqslant^{-1} uv^y$ (viz. $uv^y \leqslant uv^x$). Therefore we have $(uv^x)(v^{y-x})^\omega = uv^\omega$, $(uv^x) \in L_{q_I,p}$, $(v^{y-x}) \in L_{p,p}$, and $(uv^x)(v^{y-x}) \leqslant (uv^x)$. For every $uv^\omega \in L^\omega(\mathcal{A})$ there is $(u', v') \in S$ such that $uv^\omega = u'v'^\omega$ (Lemma 3) and by previously we can assume that $u'v' \leqslant u'$. By definition of bases there are $u_0 \in U_p$, $w_0 \in W_p$ and $v_0 \in V_p^{w_0}$ for some $p \in F$ such that $u_0 \leqslant u' \leqslant w_0$ and $v_0 \preccurlyeq_{w_0} v'$. Since $u' \leqslant w_0$, the FORQ constraint shows that $\preccurlyeq_{w_0} \subseteq \preccurlyeq_{u'}$ which, in turn, implies that $v_0 \preccurlyeq_u v'$ holds. Finally, we deduce from $u_0v_0^\omega \in M$, $u_0 \leqslant u'$, $v_0 \preccurlyeq_u v'$, $u'v' \leqslant u'$ and the M-preservation of the FORQ that $u'v'^\omega \in M$. Thus, $uv^\omega \in M$. □

6.2 The FORQ of a BA

In this section we derive a FORQ from a BA $\mathcal{B} = (\hat{Q}, \hat{q}_I, \hat{\delta}, \hat{F})$ such that it is a $L^\omega(\mathcal{B})$-suitable FORQ. For $X \subseteq \hat{Q}$ and $v \in \Sigma^*$ we define $Ctx^{\mathcal{B}}(X, v) \triangleq \{(q, q', k) \mid q \in X, q \xrightarrow{v}{}^* q', (k = \top \implies q \xrightarrow{v}{}_F{}^* q')\}$. A "context" (q, q', k) returned by $Ctx^{\mathcal{B}}$, consists in a source state $q \in Q$, a sink state $q' \in Q$ and a boolean $k \in \{\top, \bot\}$ that keeps track whether an accepting state is visited. Note that, having $\bot$ as last component of a context does *not* mean that no accepting state is visited.

Definition 6.2.1 (FORQ of a BA)**.** Given a BA $\mathcal{B} = (\hat{Q}, \hat{q}_I, \hat{\delta}, \hat{F})$ we define the family of quasiorders $\langle \preccurlyeq^{\mathcal{B}}, \{\preccurlyeq^{\mathcal{B}}_u\}_{u \in \Sigma^*} \rangle$ where $\preccurlyeq^{\mathcal{B}}$ is defined as in Definition 4.4.1 and for every $u \in \Sigma^*$ the quasiorder $\preccurlyeq^{\mathcal{B}}_u$ is defined by: $v \preccurlyeq^{\mathcal{B}}_u v' \overset{\triangle}{\iff} Ctx^{\mathcal{B}}(post^{\mathcal{B}}(u), v) \subseteq Ctx^{\mathcal{B}}(post^{\mathcal{B}}(u), v')$.

Note that for every $u \in \Sigma^*$ the quasiorder $\preccurlyeq^{\mathcal{B}}_u$ is coarser than the quasiorder $\preccurlyeq^{\mathcal{B}}$ used to prune the periods in the algorithm of Chapter Section 5. Thus, the FORQ $\langle \preccurlyeq^{\mathcal{B}}, \{\preccurlyeq^{\mathcal{B}}_u\}_{u \in \Sigma^*} \rangle$ yields more pruning than the pair of quasiorders $\preccurlyeq^{\mathcal{B}}, \preccurlyeq^{\mathcal{B}}$.

Lemma 5. *Given a BA $\mathcal{B}$, the pair $\langle \preccurlyeq^{\mathcal{B}}, \{\preccurlyeq^{\mathcal{B}}_u\}_{u \in \Sigma^*} \rangle$ of Definition 6.2.1 is a FORQ.*

Proof. Let $\mathcal{B} = (\hat{Q}, \hat{q}_I, \hat{\delta}, \hat{F})$ be a BA. The proof of right monotonicity is similar to the proof of Proposition 3. Next we prove that the FORQ constraint holds: $u \preccurlyeq^{\mathcal{B}} u' \implies \preccurlyeq^{\mathcal{B}}_{u'} \subseteq \preccurlyeq^{\mathcal{B}}_u$. First, we observe that, for all $Y \subseteq X \subseteq Q$ and all $v, v' \in \Sigma^*$, we have that $Ctx^{\mathcal{B}}(X, v) \subseteq Ctx^{\mathcal{B}}(X, v') \implies Ctx^{\mathcal{B}}(Y, v) \subseteq Ctx^{\mathcal{B}}(Y, v')$. Consider $u, u' \in \Sigma^*$ such that $u \preccurlyeq^{\mathcal{B}} u'$ and $v, v' \in \Sigma^*$ such that $v \preccurlyeq^{\mathcal{B}}_{u'} v'$. Let $X = post^{\mathcal{B}}(u)$ and $X' = post^{\mathcal{B}}(u')$, we have that $X \subseteq X'$ following $u \preccurlyeq^{\mathcal{B}} u'$. Next, we conclude from $v \preccurlyeq^{\mathcal{B}}_{u'} v'$ that $Ctx^{\mathcal{B}}(X', v) \subseteq Ctx^{\mathcal{B}}(X', v')$, hence that $Ctx^{\mathcal{B}}(X, v) \subseteq Ctx^{\mathcal{B}}(X, v')$ by the above reasoning using $X \subseteq X'$, and finally that $v \preccurlyeq^{\mathcal{B}}_u v'$. □

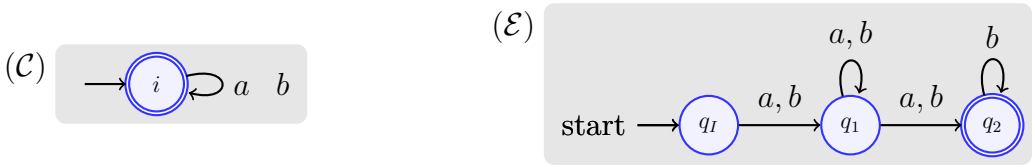


Figure 6.1: Büchi automata $\mathcal{C}$ and $\mathcal{E}$ over the alphabet $\Sigma = \{a, b\}$.

Example 15 (FORQ of a BA)**.** *Consider the BA $\mathcal{E}$ of Figure 6.1. We have $post^{\mathcal{E}}(\epsilon) = \{q_I\}$; $post^{\mathcal{E}}(a) = post^{\mathcal{E}}(b) = \{q_1\}$ and $post^{\mathcal{E}}(u) = \{q_1, q_2\}$ for all $u \in \Sigma^*$ such that $|u| \geq 2$. In particular we conclude from $u_1 \preccurlyeq^{\mathcal{E}} u_2 \overset{\triangle}{\iff} post^{\mathcal{E}}(u_1) \subseteq post^{\mathcal{E}}(u_2)$ that $a \preccurlyeq^{\mathcal{E}} aa$, $a \preccurlyeq^{\mathcal{E}} b$ and $b \preccurlyeq^{\mathcal{E}} a$; ϵ and a are incomparable; and so are ϵ and aa. Since $u \in \Sigma^* \mapsto Ctx^{\mathcal{E}}(post^{\mathcal{E}}(u), \cdot)$ has only three distinct outputs, the set $\{\preccurlyeq^{\mathcal{E}}_u\}_{u \in \Sigma^*}$ contains three distinct quasiorders.*

1. $v_1 \preccurlyeq^{\mathcal{E}}_\epsilon v_2 \overset{\triangle}{\iff} Ctx^{\mathcal{E}}(\{q_I\}, v_1) \subseteq Ctx^{\mathcal{E}}(\{q_I\}, v_2)$ *where*
   - $Ctx^{\mathcal{E}}(\{q_I\}, \epsilon) = \{(q_I, q_I, \bot)\}$
   - $Ctx^{\mathcal{E}}(\{q_I\}, a) = Ctx^{\mathcal{E}}(\{q_I\}, b) = \{(q_I, q_1, \bot)\}$
   - $Ctx^{\mathcal{E}}(\{q_I\}, v) = \{(q_I, q_1, \bot), (q_I, q_2, \bot), (q_I, q_2, \top)\}$ *for all $v \in \Sigma^*$ such that $|v| \geq 2$.*

2. $v_1 \preccurlyeq_a^{\mathcal{E}} v_2 \iff v_1 \preccurlyeq_b^{\mathcal{E}} v_2 \overset{\triangle}{\iff} Ctx^{\mathcal{E}}(\{q_1\}, v_1) \subseteq Ctx^{\mathcal{E}}(\{q_1\}, v_2)$ *where*
   - $Ctx^{\mathcal{E}}(\{q_1\}, \epsilon) = \{(q_1, q_1, \bot)\}$
   - $Ctx^{\mathcal{E}}(\{q_1\}, v) = \{(q_1, q_1, \bot), (q_1, q_2, \bot), (q_1, q_2, \top)\}$ *for all* $v \in \Sigma^+$
3. $v_1 \preccurlyeq_u^{\mathcal{E}} v_2 \overset{\triangle}{\iff} Ctx^{\mathcal{E}}(\{q_1, q_2\}, v_1) \subseteq Ctx^{\mathcal{E}}(\{q_1, q_2\}, v_2)$ *for all* $u \in \Sigma^*$ *such that* $|u| \geq 2$ *where*
   - $Ctx^{\mathcal{E}}(\{q_1, q_2\}, \epsilon) = \{(q_1, q_1, \bot), (q_2, q_2, \bot), (q_2, q_2, \top)\}$
   - $Ctx^{\mathcal{E}}(\{q_1, q_2\}, v) = \{(q_1, q_1, \bot), (q_1, q_2, \bot), (q_1, q_2, \top)\}$ *for all* $v \in \Sigma^+ \backslash \{b\}^+$
   - $Ctx^{\mathcal{E}}(\{q_1, q_2\}, v) = \{(q_1, q_1, \bot), (q_1, q_2, \bot), (q_1, q_2, \top), (q_2, q_2, \bot), (q_2, q_2, \top)\}$ *for all* $v \in \{b\}^+$.

With the above definitions the reader is invited to check that the sets $\{\epsilon, a\}$ *and* $\{\epsilon, b\}$ *are basis for* Σ^* *w.r.t.* $\leqslant^{\mathcal{E}}$*; the set* $\{\epsilon, aa\}$ *is a basis for* Σ^* *w.r.t.* $\leqslant^{\mathcal{E}^{-1}}$*; the set* $\{b\}$ *is a basis for* Σ^+ *w.r.t.* $\preccurlyeq_\epsilon^{\mathcal{E}}$ *as well as w.r.t.* $\preccurlyeq_a^{\mathcal{E}}$*; and the set* $\{a\}$ *is a basis for* Σ^+ *w.r.t.* $\preccurlyeq_{aa}^{\mathcal{E}}$*. Also observe that none of the above finite bases contains comparable words for the ordering thereof.*

As prescribed in Section 6.1, we show that for every BA $\mathcal{B}$ its FORQ is $L(\mathcal{B})$-suitable.

Proposition 13. *Given a BA* $\mathcal{B}$ *its FORQ is* $L^\omega(\mathcal{B})$*-suitable.*

Proof. Let $\mathcal{B} = (\hat{Q}, \hat{q}_I, \hat{\delta}, \hat{F})$ be a BA. Since Q is a finite set the quasiorder $\leqslant^{\mathcal{B}}$, its converse $(\leqslant^{\mathcal{B}})^{-1}$, and $\preccurlyeq_u^{\mathcal{B}}$ for every $u \in \Sigma^*$ are all well-quasiorder. The proof of decidability is trivial by Definition 6.1.2. For the preservation, given $u_0 {v_0}^\omega \in L^\omega(\mathcal{B})$, we show that for all $u \in \Sigma^*$ and all $v \in \Sigma^+$ such that $uv \leqslant^{\mathcal{B}} u$ and $u_0 \leqslant^{\mathcal{B}} u$ and $v_0 \preccurlyeq_u^{\mathcal{B}} v$ then $uv^\omega \in L^\omega(\mathcal{B})$ holds. Let a run $\pi_0 = q_I \xrightarrow{u_0}{}^* q_0 \xrightarrow{v_0}{}^* q_1 \xrightarrow{v_0}{}^* q_2 \ldots$ of $\mathcal{B}$ over $u_0 {v_0}^\omega$ which is accepting. Stated equivalently, we have $q_0 \in post^{\mathcal{B}}(u_0)$ and $(q_i, q_{i+1}, x_i) \in Ctx^{\mathcal{B}}(post^{\mathcal{B}}(u_0 v_0^i), v_0)$ for every $i \in \mathbb{N}$ with the additional constraint that $x_i = \top$ holds infinitely often.

We will show that $\mathcal{B}$ has an accepting run over uv^ω by showing that $q_0 \in post^{\mathcal{B}}(u)$ holds; $(q_i, q_{i+1}, x_i) \in Ctx^{\mathcal{B}}(post^{\mathcal{B}}(uv^i), v)$ holds for every $i \in \mathbb{N}$; and $x_i = \top$ holds infinitely often. Since $u_0 \leqslant^{\mathcal{B}} u$ and $q_0 \in post^{\mathcal{B}}(u_0)$ we find that $q_0 \in post^{\mathcal{B}}(u)$ by definition of $\leqslant^{\mathcal{B}}$. Next we show the remaining constraints by induction. The induction hypothesis states that for all $0 \leq n$ we have $(q_n, q_{n+1}, x_n) \in Ctx^{\mathcal{B}}(post^{\mathcal{B}}(uv^n), v)$. For the base case ($n = 0$) we have to show that $(q_0, q_1, x_0) \in Ctx^{\mathcal{B}}(post^{\mathcal{B}}(u), v)$. We conclude from $(q_0, q_1, x_0) \in Ctx^{\mathcal{B}}(post^{\mathcal{B}}(u), v_0)$, $v_0 \preccurlyeq_u^{\mathcal{B}} v$ and the definition of $\preccurlyeq_u^{\mathcal{B}}$ that $Ctx^{\mathcal{B}}(post^{\mathcal{B}}(u), v_0) \subseteq Ctx^{\mathcal{B}}(post^{\mathcal{B}}(u), v)$ and finally that $(q_0, q_1, x_0) \in Ctx^{\mathcal{B}}(post^{\mathcal{B}}(u), v)$. For the inductive case, assume $(q_n, q_{n+1}, x_n) \in Ctx^{\mathcal{B}}(post^{\mathcal{B}}(uv^n), v)$. The definition of context shows that $q_{n+1} \in post^{\mathcal{B}}(uv^{n+1})$. It takes an easy an induction to show that $uv^n \leqslant^{\mathcal{B}} u$ for all n using $uv \leqslant^{\mathcal{B}} u$ and right-monotonicity of $\leqslant^{\mathcal{B}}$. We conclude from $uv^{n+1} \leqslant^{\mathcal{B}} u$, the definition of $\leqslant^{\mathcal{B}}$ and $q_{n+1} \in post^{\mathcal{B}}(uv^{n+1})$ that $q_{n+1} \in post^{\mathcal{B}}(u)$ also holds, hence that $(q_{n+1}, q_{n+2}, x_{n+1}) \in Ctx^{\mathcal{B}}(post^{\mathcal{B}}(u), v_0)$ following the definition of contexts and that of π_0. Next, we find that $(q_{n+1}, q_{n+2}, x_{n+1}) \in Ctx^{\mathcal{B}}(post^{\mathcal{B}}(u), v)$ following a reasoning analogous to the base case, this time starting with $(q_{n+1}, q_{n+2}, x_{n+1}) \in Ctx^{\mathcal{B}}(post^{\mathcal{B}}(u), v_0)$. Finally, $q_{n+1} \in post^{\mathcal{B}}(uv^{n+1})$ implies that $(q_{n+1}, q_{n+2}, x_{n+1}) \in Ctx^{\mathcal{B}}(post^{\mathcal{B}}(uv^{n+1}), v)$. We have thus shown that $q_0 \in post^{\mathcal{B}}(u)$ and $(q_i, q_{i+1}, x_i) \in Ctx^{\mathcal{B}}(post^{\mathcal{B}}(uv^i), v)$ for every $i \in \mathbb{N}$ with the additional constraint that

$x_i = \top$ holds infinitely often and we are done. □

6.3 FORQ-based Algorithm

In this section we give an effective computation for the bases defining T_{finite}, hence our FORQ-based algorithm, Algorithm 5, deciding whether $L(\mathcal{A}) \subseteq M$ holds.

Given a M-suitable FORQ $\langle \leqslant, \{\preccurlyeq_u\}_{u\in\Sigma^*}\rangle$ Algorithm 5 iteratively computes the finite bases defining T_{finite} by using the fixpoint characterization $S = \bigcup_{p\in F}(\operatorname{lfp} f_{\mathcal{A}})_p \times (\operatorname{lfp} r_{\mathcal{A},p})_p$ defined in Section 5.2.1. The algorithm computes the Kleene iterates of $f_{\mathcal{A}}$ in lines 1 and 2 until they reach a finite basis for $\operatorname{lfp} f_{\mathcal{A}}$ w.r.t. $\leqslant^{|Q|}$ and $(\leqslant^{-1})^{|Q|}$ resp.[1] For every word w in the finite basis $W_p = ({f_{\mathcal{A}}}^k(\vec{\emptyset}))_p$ for $L_{q_I,p}$ w.r.t. $\leqslant^{-1}$ it similarly computes in line 5 a finite basis for $\operatorname{lfp} r_{\mathcal{A},p}$ w.r.t. $\preccurlyeq_w^{|Q|}$. The convergence of the Kleene iterates to a finite basis is checked by comparing two consecutive iterates for the quasiorder $\sqsubseteq_{\ltimes}$ where $\ltimes \in \{\leqslant, \leqslant^{-1}, \preccurlyeq_w\}$ as explained in Section 5.2.2. Finally, the algorithm checks in lines 6–7 whether every ultimately periodic word from T_{finite} belongs to M.

Algorithm 5: Algorithm for deciding $L^\omega(\mathcal{A}) \subseteq M$

Data: BA $\mathcal{A} = (Q, q_I, \delta, F)$.
Data: M-suitable FORQ $\langle \leqslant, \{\preccurlyeq_u\}_{u\in\Sigma^*}\rangle$.
Data: Procedure deciding $uv^\omega \in M$ given (u, v).

1 Compute ${f_{\mathcal{A}}}^m(\vec{\emptyset})$ with least m s.t. ${f_{\mathcal{A}}}^{m+1}(\vec{\emptyset}) \sqsubseteq_{\leqslant}^{|Q|} {f_{\mathcal{A}}}^m(\vec{\emptyset})$;
2 Compute ${f_{\mathcal{A}}}^k(\vec{\emptyset})$ with least k s.t. ${f_{\mathcal{A}}}^{k+1}(\vec{\emptyset}) \sqsubseteq_{\leqslant^{-1}}^{|Q|} {f_{\mathcal{A}}}^k(\vec{\emptyset})$;
3 **foreach** $p \in F$ **do**
4 **foreach** $w \in ({f_{\mathcal{A}}}^k(\vec{\emptyset}))_p$ **do**
5 Compute ${r_{\mathcal{A},p}}^{m'}(\vec{\emptyset})$ with least m' s.t. ${r_{\mathcal{A},p}}^{m'+1}(\vec{\emptyset}) \sqsubseteq_{\preccurlyeq_w}^{|Q|} {r_{\mathcal{A},p}}^{m'}(\vec{\emptyset})$;
6 **foreach** $u \in ({f_{\mathcal{A}}}^m(\vec{\emptyset}))_p$, $v \in ({r_{\mathcal{A},p}}^{m'}(\vec{\emptyset}))_p$ **such that** $u \leqslant w$ **do**
7 **if** $uv^\omega \notin M$ **then return** false;
8 **return** true;

Theorem 10. *Given the required inputs, Algorithm 5 decides the inclusion problem $L^\omega(\mathcal{A}) \subseteq M$.*

Proof. The algorithm computes a finite representation T_{finite} of $L^\omega(\mathcal{A})$ as in (‡). Hence, by Equation (††) we have

$$L^\omega(\mathcal{A}) \subseteq M \iff \forall (u, v) \in T_{\text{finite}},\ uv^\omega \in M\ .$$

□

The following two examples show a run of Algorithm 5 instantiated with the FORQ of Definition 6.2.1. The second one applies the antichain optimization.

[1] We lift the notion of basis to Q-dimensional vectors componentwise.

Example 16 (Run of Algorithm 5)**.** *We consider the BAs of Figure 6.1 for which we want to check whether $L(\mathcal{C}) \subseteq L(\mathcal{E})$ holds using the FORQ of $\mathcal{E}$. We have $f_{\mathcal{C}}(\emptyset) = \{\epsilon\}$, ${f_{\mathcal{C}}}^2(\emptyset) = \{\epsilon, a, b\}$ and ${f_{\mathcal{C}}}^3(\emptyset) = \{\epsilon, a, b, aa, ab, ba, bb\}$ (see Example 12). Thus, from Example 15 we deduce that ${f_{\mathcal{C}}}^3(\emptyset) \sqsubseteq_{\leqslant^{\mathcal{E}}} {f_{\mathcal{C}}}^2(\emptyset)$ so that a finite basis of $\operatorname{lfp} f_{\mathcal{C}}$ w.r.t. $\leqslant^{\mathcal{E}}$ is achieved by ${f_{\mathcal{C}}}^2(\emptyset)$. Similarly, it is routine to check that ${f_{\mathcal{C}}}^3(\emptyset)$ is a finite basis of $\operatorname{lfp} f_{\mathcal{C}}$ w.r.t. $(\leqslant^{\mathcal{E}})^{-1}$. Hence, the subset of words ${f_{\mathcal{C}}}^3(\emptyset)$ induces the three distinct quasiorders $\preccurlyeq^{\mathcal{E}}_{\epsilon}$, $\preccurlyeq^{\mathcal{E}}_{a}$ and $\preccurlyeq^{\mathcal{E}}_{aa}$ defined in Example 15. Then, $r_{\mathcal{C}}(\emptyset)$ is a finite basis of $\operatorname{lfp} r_{\mathcal{C}}$ w.r.t. any of these three quasiorders. Finally, we find $T_{\mathit{finite}} = {f_{\mathcal{C}}}^2(\emptyset) \times r_{\mathcal{C}}(\emptyset)$, $(a, a) \in T_{\mathit{finite}}$ and $a^\omega \notin L(\mathcal{E})$. By checking membership of the ultimately periodic word a^ω from T into $L(\mathcal{E})$ we thus have shown that $L(\mathcal{C}) \subseteq L(\mathcal{E})$ does not hold.*

Example 17 (Antichains Everywhere)**.** *We continue from Example 16. Since a and b are equivalent w.r.t. $\leqslant^{\mathcal{E}}$ we can keep the antichains $\{\epsilon, a\}$ and $\{\epsilon, aa\}$ w.r.t. $\leqslant^{\mathcal{E}}$ and $(\leqslant^{\mathcal{E}})^{-1}$ instead of ${f_{\mathcal{C}}}^2(\emptyset)$ and ${f_{\mathcal{C}}}^3(\emptyset)$. Similarly, we can keep the antichain $\{a\}$ w.r.t. $\preccurlyeq^{\mathcal{E}}_{u}$ for $u \in \{\epsilon, aa\}$ instead of $r_{\mathcal{C}}(\emptyset)$. We then perform the membership querries on the set $T_{\mathit{finite}} = \{\epsilon, a\} \times \{a\}$.*

6.3.1 Why a basis w.r.t. $\leqslant^{-1}$ is computed?

Taking periods in a basis w.r.t. $\preccurlyeq_w$ where $w \in \Sigma^*$ is picked to be a maximal w.r.t. $\leqslant$ word of $L_{q_I,p}$ may seem unnatural. In fact, the language preservation property of FORQ even suggests that an algorithm without computing a basis w.r.t. $\leqslant^{-1}$ may exist. Here, we show that taking periods in a basis w.r.t. $\preccurlyeq_u$ where $u \in \Sigma^*$ is picked in a basis w.r.t. $\leqslant$ is not correct. More precisely, redefining T_{finite} as

$$\tilde{T} = \bigcup_{p \in F} \left\{ (u, v) \in \Sigma^* \times \Sigma^+ \mid u \in U_p, v \in V_p^u \right\} ,$$

where for all $p \in P$, U_p is a basis for $L_{q_I,p}$ w.r.t. $\leqslant$ and V_p^u is a basis for $L_{p,p}\backslash\{\epsilon\}$ w.r.t. $\preccurlyeq_u$, leads to an *incorrect* algorithm because the equivalence (††) given by $L^\omega(\mathcal{A}) \subseteq M \iff \forall (u, v) \in \tilde{T},\ uv^\omega \in M$ no longer holds as shown in the below example.

Example 18. *Consider the BA given by Figure 6.1. We have that $L(\mathcal{C}) \nsubseteq L(\mathcal{E})$ and, in Example 17, we have argued that $T_{\mathit{finite}} = \{\epsilon, a\} \times \{a\}$ contains the ultimately periodic word a^ω which is a counterexample to inclusion. From Example 15 we can set $U_i = \{\epsilon, a\}$ since $\{\epsilon, a\}$ is a basis for $L_{i,i} = \{a, b\}^*$, and $V_i^a = V_i^\epsilon = \{b\}$ since $\{b\}$ is a basis for $L_{i,i}\backslash\{\epsilon\} = \{a, b\}^+$ w.r.t. $\preccurlyeq_a$ and $\preccurlyeq_\epsilon$. We conclude from the above definition that $\tilde{T} = \{(\epsilon, b), (a, b)\}$, hence that $\forall (u, v) \in \tilde{T}, uv^\omega \in L(\mathcal{B})$ which contradicts (††) since $L(\mathcal{C}) \nsubseteq L(\mathcal{E})$.*

We conclude this section with the algorithmic complexity of Algorithm 5.

6.3.2 Complexity

Next, we establish an upper bound on the runtime of Algorithm 5 when the input FORQ is $\langle \leqslant^{\mathcal{B}}, \{\preccurlyeq^{\mathcal{B}}_u\}_{u \in \Sigma^*} \rangle$ induced by a BA $\mathcal{B}$. Let n and $\hat{n}$ be respectively the number of states of $\mathcal{A}$ and $\mathcal{B}$. By analogue arguments as in Proposition 5 we find that $|\Sigma| \cdot n^2 \cdot 2^{O(\hat{n}^2)}$ is an upper bound on the runtime of lines $1, 2$ and 5. Next, we derive an upper bound on the number of membership queries performed at line 7. The number of iterations of the loops of lines $3, 4, 5$ and 6 is n, $2^{\hat{n}}$, $n \cdot 2^{(2\hat{n}^2)}$ and $2^{(2\hat{n}^2)} \cdot 2^{\hat{n}}$, respectively. Since all loops are nested, we multiply these bounds to end up with $n^2 \cdot 2^{\mathcal{O}(\hat{n}^2)}$ as an upper bound on the number of membership

queries. The runtime for each ultimately periodic word membership query (with a prefix, a period and $\mathcal{B}$ as input) is upper bounded by an expression polynomial in the size $\hat{n}$ of $\mathcal{B}$, $2^{\hat{n}}$ for the length of the prefix and $2^{(2\hat{n}^2)}$ for the length of the period.

6.4 Discussions

This section provides informations that we consider of interest although not essential for the correctness of the FORQ algorithm or its evaluation.

6.4.1 Origin of FORQ

Our definition of FORQ and their suitability property (in particular the language preservation) are directly inspired from the definitions related to families of right congruences introduced by Maler and Staiger in 1993 [59] (revised in 2008 [61]). We now explain how our definition of FORQ generalizes and relaxes previous definitions [61, Definitions 5 and 6].

First we explain why the FORQ constraint does not appear in the setting of families of right congruences. In the context of congruences, relations are symmetric and thus, the FORQ constraint reduces to $u \leqslant u' \Rightarrow \preccurlyeq_{u'} = \preccurlyeq_u$. Therefore the FORQ constraint trivially holds if the set $\{\preccurlyeq_u\}_{u\in\Sigma^*}$ is quotiented by the congruence relation $\leqslant$, which is the case in the definition [60, Definition 5].

Second, we point that the condition $v \preccurlyeq_u v' \Rightarrow uv \leqslant uv'$ which appears in the definition for right families of congruences [61, Definition 5] is not needed in our setting. Nevertheless, this condition enables an improvement of the FORQ-based algorithm that we describe next.

6.4.2 Less membership queries

We put forward a property of the FORQ of a BA allowing us to reduce the number of membership queries performed by Algorithm 5. Hereafter, we refer to the *picky constraint* as the property of a FORQ stating $v \preccurlyeq_u v' \Rightarrow uv \leqslant uv'$ where $u, v, v' \in \Sigma^*$. We first show how thanks to the picky constraint we can reduce the number of candidate counterexamples in the FORQ-based algorithm and then, we show that the FORQ of a BA of Definition 6.2.1 satisfies the picky constraint.

In Algorithm 5, periods are taken in a basis w.r.t. $\preccurlyeq_w$ where $w \in \Sigma^*$ belongs to a finite basis w.r.t. $\leqslant^{-1}$. The only restriction on w is that of being comparable to the stem u, as ensured by the test at line 6. The following lemma formalizes the fact that we could consider a stronger restriction.

Lemma 6. *Let $\leqslant$ be a quasiorder over Σ^* such that $\leqslant^{-1}$ is a right-monotonic well-quasiorder. Let $S \subseteq \Sigma^*$ and S' be a basis for S w.r.t. $\leqslant^{-1}$ such that S' is an antichain. For all $u \in \Sigma^*$ and $v \in \Sigma^+$ such that $u \in S$ and $\{wv \mid w \in S\} \subseteq S$, there exists $\mathring{w} \in S'$ such that $uv^i \leqslant \mathring{w}$ and $\mathring{w}v^j \leqslant \mathring{w}$ for some $i, j \in \mathbb{N} \setminus \{0\}$.*

Proof. Since $\leqslant^{-1}$ is a well-quasiorder, S admits a finite basis $S' = \{\mathring{w}_1, \ldots, \mathring{w}_k\}$ that is an antichain w.r.t. $\leqslant^{-1}$. Let $u \in \Sigma^*$ and $v \in \Sigma^+$ such that $u \in S$ and $\{wv \mid w \in S\} \subseteq S$. For

all $i \in \{1, \dots, k\}$, we have $\mathring{w}_i v \in S$ by hypothesis, and thus there exists $j \in \{1, \dots, k\}$ such that $\mathring{w}_j \leqslant^{-1} \mathring{w}_i v$. Hence, the function $f \colon \{1, \dots, k\} \mapsto \{1, \dots, k\}$ defined by $f \colon i \mapsto \min\{j \mid \mathring{w}_j \leqslant^{-1} \mathring{w}_i v\}$ is well defined and we have $\mathring{w}_{f(i)} \leqslant^{-1} \mathring{w}_i v$ for all $i \in \{1, \dots, k\}$.

Knowing $u \in S$, we exhibit some $l \in \{1, \dots, k\}$ satisfying $\mathring{w}_l \leqslant^{-1} u$. Consider the infinite sequence $\{f^n(l)\}_{n \in \mathbb{N}}$ where $f^0(l) = l$ and $f^{n+1}(l) = f(f^n(l))$. We prove by induction that $\mathring{w}_{f^{i+j}(l)} \leqslant^{-1} \mathring{w}_{f^i(l)} v^j$ for all $i, j \in \mathbb{N}$. If $j = 0$, then $\mathring{w}_{f^i(l)} \leqslant^{-1} \mathring{w}_{f^i(l)}$ holds for all $i \in \mathbb{N}$ by reflexivity. Assume that $j > 0$ and $\mathring{w}_{f^{i+j}(l)} \leqslant^{-1} \mathring{w}_{f^i(l)} v^j$. The right-monotonicity of $\leqslant^{-1}$ implies $\mathring{w}_{f^{i+j}(l)} v \leqslant^{-1} \mathring{w}_{f^i(l)} v^{j+1}$. By construction $\mathring{w}_{f^{i+j+1}(l)} \leqslant^{-1} \mathring{w}_{f^{i+j}(l)} v$. Hence $\mathring{w}_{f^{i+j+1}(l)} \leqslant^{-1} \mathring{w}_{f^i(l)} v^{j+1}$ holds.

As a consequence of the finiteness of the set $\{f^n(l) \mid n \in \mathbb{N}_{\neq 0}\}$, there exists $x, y \in \mathbb{N}_{\neq 0}$ such that $0 < x < y$ and $f^x(l) = f^y(l)$. Thanks to the property previously proved, the following holds.

- $\mathring{w}_{f^y(l)} \leqslant^{-1} \mathring{w}_{f^x(l)} v^{y-x}$ by taking $i = x$ and $j = y - x$.
- $\mathring{w}_{f^x(l)} \leqslant^{-1} \mathring{w}_l v^x$ by taking $i = 0$ and $j = x$.

Let $\mathring{w} = \mathring{w}_{f^x(l)} = \mathring{w}_{f^y(l)}$. We have that $\mathring{w} \leqslant^{-1} \mathring{w} v^{y-x}$ and $\mathring{w} \leqslant^{-1} \mathring{w}_l v^x$. In addition $\mathring{w}_l \leqslant^{-1} u$ implies $\mathring{w} \leqslant^{-1} u v^x$ by right-monotonicity and transitivity of $\leqslant^{-1}$. □

Next, we show that the equivalence (††) holds but this time for an alternative definition of T_{finite} we provide next. Given a M-suitable FORQ $\langle \leqslant, \{\preccurlyeq_u\}_{u \in \Sigma^*} \rangle$ that satisfies the picky constraint let

$$\hat{T} = \{(u, v) \in T_{\text{finite}} \mid \exists s \in F \colon u \in U_s, v \in V_s^w \text{ for some } w \in W_s \text{ with } u \leqslant w, wv \leqslant w\}$$

where for all $s \in F$ we fixed a basis U_p for $L_{q_I,p}$ w.r.t. $\leqslant$, a basis W_p for $L_{q_I,p}$ w.r.t. $\leqslant^{-1}$, and a basis V_p^w for $L_{p,p} \backslash \{\epsilon\}$ w.r.t. $\preccurlyeq_w$ for every $w \in W_p$.

Proposition 14. *Given a M-suitable FORQ that satisfies the picky constraint we have* $L^\omega(\mathcal{A}) \subseteq M \iff \forall (u, v) \in \hat{T},\ uv^\omega \in M$.

Proof. The direction $\Rightarrow$ of the equivalence holds because $\hat{T}$ is a subset of decompositions of ultimately periodic words of $L^\omega(\mathcal{A})$. For the converse direction assume that $\forall (u, v) \in \hat{T},\ uv^\omega \in M$. Let $uv^\omega \in L^\omega(\mathcal{A})$ and T_{finite} be such that the basis W_s is an antichain. Since T_{finite} is a representation of $L^\omega(\mathcal{A})$ we can assume that $(u, v) \in T_{\text{finite}}$. By definition of T_{finite} there is $s \in F$ such that $u \in L_{q_I,s}$ and $v \in L_{s,s}$. Thus, in the context of Lemma 6, taking $S = L_{q_I,s}$ fulfills the requirements $u \in S$ and $\{wv \mid w \in S\} \subseteq S$. We can thus apply the lemma with $S = L_{q_I,s}$ and $S' = W_s$ to ensure the existence of some $w_0 \in W_s$ satisfying $uv^i \leqslant w_0$ and $w_0 v^j \leqslant w_0$ for some $i, j \in \mathbb{N} \setminus \{0\}$. Since $uv^i \in L_{q_I,s}$ and $v^j \in L_{s,s}$ we find that there exist $u_0 \in U_s$ and $v_0 \in V_s^{w_0}$ such that $u_0 \leqslant uv^i$ and $v_0 \preccurlyeq_{w_0} v^j$ by definition of basis. Using the picky constraint we conclude from $v_0 \preccurlyeq_{w_0} v^j$ that $w_0 v_0 \leqslant w_0 v^j$, and finally from $w_0 v^j \leqslant w_0$ that $w_0 v_0 \leqslant w_0$ by transitivity. By definition $(u_0, v_0) \in \hat{T}$, thus by assumption $u_0 v_0^\omega \in M$ From $uv^i \leqslant w_0$, $v_0 \preccurlyeq_{w_0} v^j$ and the FORQ constraint we deduce that $v_0 \preccurlyeq_{uv^i} v^j$. Finally, from $u_0 \leqslant uv^i$, $v_0 \preccurlyeq_{uv^i} v^j$ and $u_0 v_0^\omega \in M$ we deduce by preservation that $uv^\omega \in M$. □

To summarize, if the considered FORQ fulfills the picky constraint then Algorithm 5 remains

correct when discarding the periods v at line 6 such that $wv \not\leqslant w$. Observe that discarding one period v possibly means skipping several membership queries $(u_1v^\omega, u_2v^\omega, \ldots)$. As proved below, the picky constraint holds for the FORQ of all BA.

Lemma 7. *Given a BA $\mathcal{B}$, its FORQ $\langle \leqslant^{\mathcal{B}}, \{\preccurlyeq_u^{\mathcal{B}}\}_{u\in\Sigma^*}\rangle$ satisfies the picky constraint.*

Proof. We show that for every $u \in \Sigma^*$ and every $v, v' \in \Sigma^+$ if $v \preccurlyeq_u^{\mathcal{B}} v'$ then $uv \leqslant^{\mathcal{B}} uv'$. For every $q' \in post^{\mathcal{B}}(uv)$, there exists $q \in post^{\mathcal{B}}(u)$ such that $(q, q') \in ctx^{\mathcal{B}}(v)$. Hence $(q, q', \bot) \in Ctx^{\mathcal{B}}(post^{\mathcal{B}}(u), v)$. In fact $(q, q', \bot) \in Ctx^{\mathcal{B}}(post^{\mathcal{B}}(u), v')$ holds as well since $v \preccurlyeq_u^{\mathcal{B}} v'$. We deduce from the definition of $Ctx^{\mathcal{B}}$ that $q' \in post^{\mathcal{B}}(uv')$. Thus $post^{\mathcal{B}}(uv) \subseteq post^{\mathcal{B}}(uv')$ i.e., $uv \leqslant^{\mathcal{B}} uv'$. □

In the following section, we will introduce our tool, called FORKLIFT, which implements Algorithm 5 instantiated by the FORQ of Definition 6.2.1. We emphasize that the optimization we discussed earlier, which reduces the number of membership queries, was not included in our experimental evaluation. This omission is motivated by two considerations: firstly, the simplification it brings to the proof of correctness, and secondly, FORKLIFT already scales up well without this specific optimization. We leave for future work the precise effect of such optimization.

6.5 Implementation and experiments

In this section, we carry out an evaluation of the FORQ-based framework introduced in this chapter, and we also compare it to the framework presented in Chapter 5 (specifically, Algorithm 4), where only two quasiorders (one for prefixes and another for periods) are employed to eliminate potential counterexamples.

We have implemented Algorithm 4 in a tool named BAIT [29]. It's worth noting that Algorithm 4 exclusively operates on automata states and doesn't handle or store words directly.

For the FORQ-based approach, we have similarly implemented a "state-based" variant of Algorithm 5 using the FORQ of the BA $\mathcal{B}$ as defined in Definition 6.2.1. More precisely, our implementation does not recompute the associated set of $post^{\mathcal{B}}$ nor $Ctx^{\mathcal{B}}$ for the newly computed words from scratch. For every prefix $u \in \Sigma^*$ and every letter $a \in \Sigma$, the set of states $post^{\mathcal{B}}(ua)$ is computed from $post^{\mathcal{B}}(u)$ thanks to the equality 4.1 essentially stating that $post^{\mathcal{B}}$ can be computed inductively. Analogously, for every period $v \in \Sigma^+$, every $X \subseteq \hat{Q}$ and every $a \in \Sigma$, the set of contexts $Ctx^{\mathcal{B}}(X, va)$ is computed from $Ctx^{\mathcal{B}}(X, v)$ thanks to the following equality:

$$Ctx^{\mathcal{B}}(X, va) = \left\{ (q_0, q, k) \in Q^2 \times \{\bot, \top\} \;\middle|\; \begin{array}{l} (q_0, q', k') \in Ctx^{\mathcal{B}}(X, v),\ : (q', q) \in ctx^{\mathcal{B}}(a) \\ (k = \bot \vee k' = \top \vee (q', q) \in ctx^{\mathcal{B}}_{\circledast}(a)) \end{array} \right\}.$$

Intuitively $Ctx^{\mathcal{B}}$ can be computed inductively as we did for $post^{\mathcal{B}}$. The first part of the condition defines how new context are obtained by appending a transition to the right of an existing context while the second part defines the bit of information keeping record of whether an accepting state was visited.

Both FORKLIFT and BAIT incorporate the antichain optimization, which prunes subsumed words, ensuring that each computed set contains the fewest possible words.

Furthermore, our implementations are naïve prototypes, consisting of less than 1,000 lines of Java code, with our primary objective in designing these tools being the preservation of code simplicity.

6.5.1 Experimental Evaluation

Benchmarks. Our evaluation uses benchmarks stemming from various application domains including benchmarks from theorem proving and software verification. In this section, a *benchmark* means an ordered pair of BA such that the "left"/"right" BA refer, resp., to the automata on the left/right of the inclusion sign. The BA of the Pecan [68] benchmarks encode sets of solutions of predicates, hence a logical implication between predicates reduces to a language inclusion problem between BA. The benchmarks correspond to theorems of type $\forall x, \exists y,\ P(x) \implies Q(y)$ about Sturmian words [46]. We collected 60 benchmarks from Pecan for which inclusion holds, where the BA have alphabets of up to 256 symbols and have up to 21 395 states.

The second collection of benchmarks stems from software verification. The Ultimate Automizer (UA) [44, 43] benchmarks encode termination problems for programs where the left BA models a program and the right BA its termination proof. Overall, we collected 600 benchmarks from UA for which inclusion holds for all but one benchmark. The BA have alphabets of up to 13 173 symbols and are as large as 6 972 states.

The RABIT benchmarks are BA modeling mutual exclusion algorithms [3], where in each benchmark one BA is the result of translating a set of guarded commands defining the protocol while the other BA translates a modified set of guarded commands, typically obtained by randomly weakening or strengthening one guard. The resulting BA are on a binary alphabet and are as large as 7 963 states. Inclusion holds for 9 out of the 14 benchmarks.

All the benchmarks are publicly available on GitHub [25]. We used all the benchmarks we collected, that is, we discarded no benchmarks.

Tools. We compared FORKLIFT [28] and BAIT [29] with the following tools: SPOT 2.10.3, GOAL (20200822), RABIT 2.5.0, and ROLL 1.0.

SPOT [33, 32] decides inclusion problems by complementing the "right" BA via determinization to parity automata with some additional optimizations including simulation-based optimizations. It is invoked through the command line tool `autfilt` with the option `-included-in`. It is worth pointing out that SPOT works with symbolic alphabets where symbols are encoded using Boolean propositions, and sets of symbols are represented and processed using OBDDs. SPOT is written in C++ and its code is publicly available [74].

GOAL [77] contains several language inclusion checkers available with multiple options. We used the Piterman algorithm using the options `containment -m piterman` with and without the additional options `-sim -pre`. In our plots GOAL is the invocation

with the additional options `-sim -pre` which compute and use simulation relations to further improve performance while GOAL$^-$ is the one without the additional options. Inclusion is checked by constructing on-the-fly the intersection of the "left" BA and the complement of the "right" BA which is itself built on-the-fly by the Piterman construction [66]. The Piterman check was deemed the "best effort" (cf. [18, Section 9.1] and [78]) among the inclusion checkers provided in GOAL. GOAL is written in Java and the source code of the release we used is not publicly available [41].

RABIT [18] performs the following operations to check inclusion: (1) Removing dead states and minimizing the automata with simulation-based techniques, thus yielding a smaller instance; (2) Witnessing inclusion by simulation already during the minimization phase; (3) Using a Ramsey-based method with antichain heuristics to witness inclusion or non-inclusion. The antichain heuristics of Step (3) uses a unique quasiorder leveraging simulation relations to discard candidate counterexamples. In our experiments we ran RABIT with options `-fast -jf` which RABIT states as providing the "best performance". RABIT is written in Java and is publicly available [71].

ROLL [56, 55] contains an inclusion checker that does a preprocessing analogous to that of RABIT and then relies on automata learning and word sampling techniques to decide inclusion. ROLL is written in Java and is publicly available [72].

As far as we can tell all the above implementations are sequential except for RABIT which, using the `-jf` option, performs some computations in a separate thread.

Experimental Setup. We ran our experiments on a server with 24 GB of RAM, 2 Xeon E5640 2.6 GHz CPUs and Debian Stretch 64-bit. We used openJDK 11.0.12 2021-07-20 when compiling Java code and ran the JVM with default options. For RABIT, BAIT and FORKLIFT the execution time is computed using timers internal to their implementations. For ROLL, GOAL and SPOT the execution time is given by the "real" value of the `time(1)` command. We preprocessed the benchmarks passed to FORKLIFT and BAIT with a reduction of the set of final states of the "left" BA that does not alter the language it recognizes. This preprocessing aims to minimize the number of iterations of the loop at line 3 of Algorithm 5 over the set of final states. It is carried out by GOAL using the `acc -min` command. Internally, GOAL uses a polynomial time algorithm that relies on computing strongly connected components. The time taken by this preprocessing is negligible.

Plots. We use survival plots for displaying our experimental results in Figure 7.1. Let us recall how to obtain them for a family of benchmarks $\{p_i\}_{i=1}^n$: (1) run the tool on each benchmark p_i and store its runtime t_i; (2) sort the t_i's in increasing order and discard pairs corresponding to abnormal program termination like time out or memory out; (3) plot the points $(t_1, 1), (t_1 + t_2, 2), \ldots$, and in general $(\sum_{i=1}^k t_i, k)$; (4) repeat for each tool under evaluation.

Survival plots are effective at comparing how tools scale up on benchmarks: the further right and the flatter a plot goes, the better the tool thereof scales up. Also the closer to the x-axis a plot is, the less time the tool needs to solve the benchmarks.

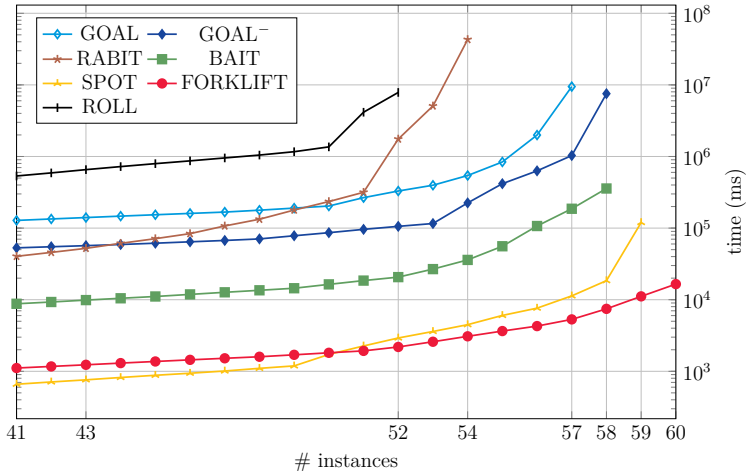
Analysis. It is clear from Figure 6.2a and 6.2b that FORKLIFT scales up best on both the Pecan and UA benchmarks. FORKLIFT's scalability is particularly evident on the PECAN benchmarks of Figure 6.2a where its curve is the flattest and no other tool finishes on all benchmarks. Note that, in Figure 6.2b, the plot for SPOT is missing because we did not succeed into translating the UA benchmarks in the input format of SPOT. On the UA benchmarks, FORKLIFT, BAIT and GOAL scale up well and we expect SPOT to scale up at least equally well. On the other hand, RABIT and ROLL scaled up poorly on these benchmarks.

On the RABIT benchmarks at Figure 6.2c both FORKLIFT and SPOT terminate 13 out of 14 times; BAIT terminates 9 out of 14 times; and GOAL, ROLL and RABIT terminate all the times. We claim that the RABIT benchmarks can all be solved efficiently by leveraging simulation relations which FORKLIFT does not use let alone compute. Next, we justify this claim. First observe at Figure 6.2c how GOAL is doing noticeably better than GOAL$^-$ while we have the opposite situation for the Pecan benchmarks Figure 6.2a and no noticeable difference for the UA benchmarks Figure 6.2b. Furthermore observe how ROLL and RABIT, which both leverage simulation relations in one way or another, scale up well on the RABIT benchmarks but scale up poorly on the PECAN and UA benchmarks.

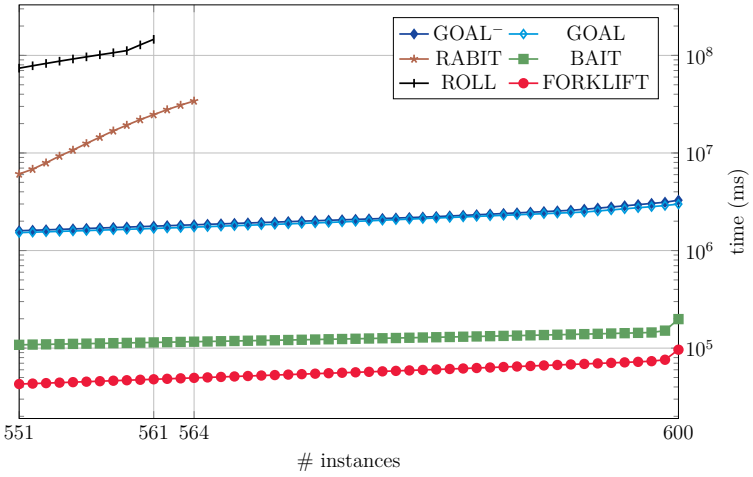
The reduced RABIT benchmarks at Figure 6.2d are obtained by pre-processing every BA of every RABIT benchmark with the simulation-based reduction operation of SPOT given by `autfilt -high -ba`. This preprocessing reduces the state space of the BA by more than 90% in some cases. The reduction significantly improves how FORKLIFT scales up (it now terminates on all benchmarks) while it has less impact on RABIT, ROLL and SPOT which, as we said above, already leverage simulation relation internally. It is also worth noting that GOAL has a regression (from 14/14 before the reduction to 13/14).

Overall FORKLIFT, even though it is a prototype implementation, is the tool that returns most often (673/674). Its unique failure disappears after a preprocessing using simulation relations of the two BA. The FORKLIFT curve for the Pecan benchmarks shows FORKLIFT scales up best.

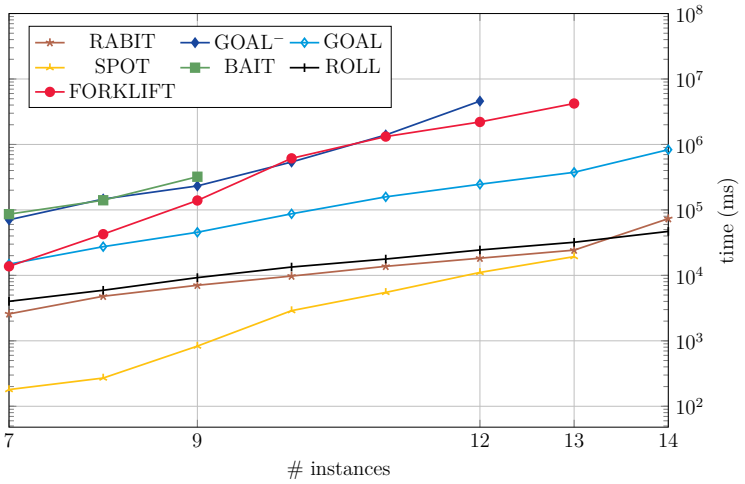
Our conclusion from the empirical evaluation is that, in practice FORKLIFT is competitive compared to the state-of-the-art in terms of scalability. Moreover the behavior of the FORQ-based algorithm in practice is far from its worst case exponential runtime.



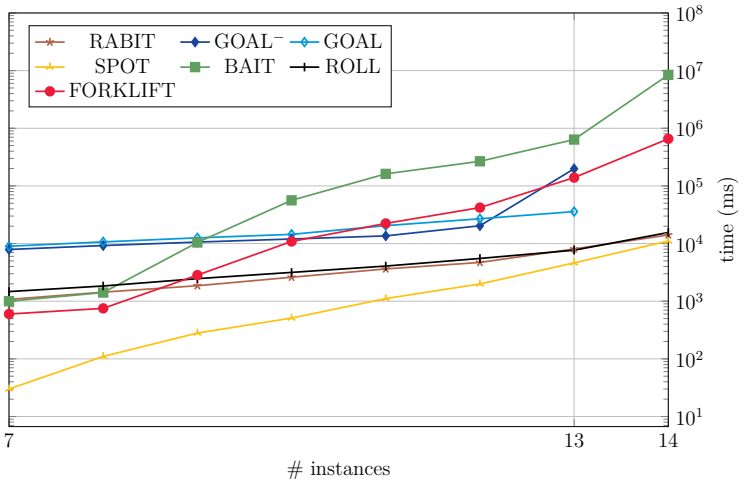
(a) Benchmarks from Pecan



(b) Benchmarks from Ultimate Automizer



(c) Benchmarks from RABIT



(d) Benchmarks from RABIT (reduced)

Figure 6.2: Survival plot with a logarithmic y axis and linear x axis. Each benchmark has a timeout value of 12h. Parts of the plots left out for clarity. A point is plotted for abscissa value x and tool r iff r returns with an answer for x benchmarks. All the failures of BAIT and the one of FORKLIFT are memory out.

Chapter 7

INCLUSION FOR VISIBLY PUSHDOWN LANGUAGES

In this chapter we define algorithms for the inclusion problem between two visibly pushdown languages of infinite words.

7.1 Overview

We solve the inclusion problem $L^\omega(\mathcal{A}) \subseteq M$ where, $\mathcal{A} = (Q, q_I, \Gamma, \delta, F)$ is a VPA and M a ω-VPL, by identifying a finite subset S_{finite} of ultimately periodic words of $L^\omega(\mathcal{A})$ such that

$$L^\omega(\mathcal{A}) \subseteq M \iff \forall (u, v) \in S_{\text{finite}},\ uv^\omega \in M \ . \tag{$\dagger$}$$

We select S_{finite} as a basis w.r.t. a pair of quasiorders. A difference with the framework of Chapter 5 is that now, the subset S_{finite} only contains a specific type of decompositions of ultimately periodic words called *legitimate decompositions*. The set of legitimate decompositions is given by

$$\texttt{Ld} \triangleq \texttt{C} \times \texttt{C} \ \cup \ \texttt{U}_\texttt{c} \times \texttt{R} \ ,$$

where, $\texttt{C}$ is the set of finite words where all call positions are matched, $\texttt{R}$ is the set of finite words where all return positions are matched and, $\texttt{U}_\texttt{c}$ is the set of finite words with at least one unmatched call position (see Definition 3.6.1).

The Equivalence ($\dagger$) yields a correct algorithm because of the following.

1. Ultimately periodic words are sufficient to solve the inclusion problem between ω-VPL, as shown by the following theorem[1]

   Theorem 11. *Let $L, M \subseteq \Sigma^\omega$ be ω-VPL. Then, $L \subseteq M$ iff $\forall uv^\omega \in L, uv^\omega \in M$.*

2. We can identify a sufficient subset S_{finite} of legitimate decompositions of $L(\mathcal{A})$ that admits a least fixed point characterization.

[1] Theorem 11 can be easily obtained by adapting the proof of Fact 1 in [17].

3. The language M admits an M-suitable pair of quasiorders i.e., a pair of decidable, M-preserving, well-quasiorders that satisfy monotonicity conditions aligned with the fixpoint functions. In the VPL setting, we restrict the definition of M-preservation to the legitimate decompositions as follows. A pair $\leqslant, \preccurlyeq$ of quasiorders on Σ^* is said to be M-preserving if for all $(u,v),(u',v') \in \mathtt{Ld}$ such that $(u,v),(u',v') \in \mathtt{C} \times \mathtt{C}$ or $(u,v),(u',v') \in \mathtt{U_c} \times \mathtt{R}$,if $uv^\omega \in M, u \leqslant u'$ and $v \preccurlyeq v'$ then $u'v'^\omega \in M$.

7.2 Reduction to a Finite Basis

In this section we show how to reduce the inclusion problem $L^\omega(\mathcal{A}) \subseteq M$ to a finite basis w.r.t. a pair of quasiorders.

7.2.1 A Sufficient Subset of Legitimate Decompositions

Our first step is to reduce the inclusion check to a subset of ultimately periodic words of $L^\omega(\mathcal{A})$ given by legitimate decompositions. As shown next, every ultimately periodic word admits a legitimate decomposition.

Proposition 15. *Let $st^\omega \in \Sigma^\omega$. There exists $(u,v) \in \mathtt{Ld}$ such that $st^\omega = uv^\omega$.*

Proof. Let $st^\omega \in \Sigma^\omega$ be an ultimately periodic word. We distinguish two cases: all call positions of st^ω are matched, or at least one call position is unmatched.

- Assume that all call positions of st^ω are matched. If $s \in \mathtt{U_c}$, then there must be some finite prefix t' of t^ω such that $st' \in \mathtt{C}$ (or st^ω would have an unmatched call position). Hence, there are $t', t'' \in \Sigma^*$ such that $st' \in \mathtt{C}$, $t't'' = t^n$ for some $n \in \mathbb{N}$ and $st^\omega = st't''t^\omega$. Let $t = t_0 \dots t_k \in \Sigma^*$. Then $t' = t^{n'}t_0 \cdots t_m$ and $t'' = t_{m+1} \cdots t_k t^{n''}$ for some $m \in [0,k]$ and $n', n'' \in \mathbb{N}$ such that $n = n' + n'' + 1$. Hence, for $x = t_{m+1} \cdots t_k t_0 \cdots t_m$ we have that $st^\omega = st'x^\omega$. If x^ω admits a decomposition uv^ω with $u, v \in \mathtt{C}$, so does st^ω, since $st^\omega = st'uv^\omega$ and $st'u \in \mathtt{C}$. Next we show that x^ω admits such a decomposition. If $x \in \mathtt{C}$ then we are done. Otherwise let $x^\omega \triangleq x_0x_1 \cdots$ and let $n \in [0,k]$ be the first unmatched call position in x. Since x^ω is a suffix of st^ω all its call positions are matched. Thus, there is $j \in [1, n-1]$ such that $n \curvearrowright_{x^\omega} k+j$. Since n is the first unmatched call position in x we have $s' \triangleq x_0 \cdots x_{n-1} \in \mathtt{C}$ and by the definition of $\curvearrowright_{x^\omega}$ we deduce that $w \triangleq x_n \cdots x_{k+j} \in \mathtt{W}$. Since $x_{j+1} \cdots x_{n-1}$ is a suffix of $s' \in \mathtt{C}$ we also have $x_{j+1} \cdots x_{n-1} \in \mathtt{C}$. Thus, $wx_{j+1} \cdots x_{n-1} \in \mathtt{C}$ and $x^\omega = s'(wx_{j+1} \cdots x_{n-1})^\omega$ is the desired decomposition.
- Assume that st^ω at least one call position j which is unmatched. We can assume that $j < |s|$, otherwise we take $s' \triangleq st^n$ for some n such that $j < |st^n|$ and consider the decomposition $s't^\omega$ instead. The call position j is also unmatched in s, hence, $s \in \mathtt{U_c}$. We can assume that j is the last unmatched call position in s. Assume that $t \notin \mathtt{R}$ and let $k \in [0, |t|-1]$ be the first unmatched return position in t. We have that $j \curvearrowright_{st^\omega} |s| + k$. This is a contradiction since the call position j is unmatched in st^ω. Hence, $t \in \mathtt{R}$.

□

As a consequence of Theorem 11 and Proposition 15, we obtain the following theorem, stating that the legitimate decompositions are sufficient for the inclusion problem between ω-VPL.

Theorem 12. *Let $L, M \subseteq \Sigma^\omega$ be ω-VPL. Then, $L \subseteq M$ iff $\forall (u,v) \in \mathtt{Ld}$, $uv^\omega \in L \implies uv^\omega \in M$.*

We now leverage the relations $\vdash^*$ and $\vdash^\circledast$ of $\mathcal{A}$ to characterize the legitimate decompositions of the ultimately periodic words in $L^\omega(\mathcal{A})$. For each pair $p, q \in Q$ of states of $\mathcal{A}$: $L_{p,q} \triangleq \{u \in \Sigma^* \mid \exists w \in \Gamma^*,\ (p,\bot) \vdash^{*u} (q,w)\}$ and $L^\circledast_{p,q} \triangleq \{u \in \Sigma^+ \mid \exists w \in \Gamma^*,\ (p,\bot) \vdash^{\circledast u} (q,w)\}$. In turn, define the following subset of $\mathtt{Ld}$:

$$S \triangleq \bigcup_{p \in Q} L_{q_I,p|\mathtt{C}} \times L^\circledast_{p,p|\mathtt{C}} \ \cup\ L_{q_I,p|\mathtt{U_c}} \times L^\circledast_{p,p|\mathtt{R}}\ .$$

Here, $L_{|K}$ is defined as $L \cap K$ to emphasize that L is restricted to K.

Example 19. *Consider the VPA $\mathcal{A}$ and $\mathcal{B}$ depicted in Fig. 3.2. We have $L^\omega(\mathcal{A}) = \mathtt{R}^\omega$, $S = (\mathtt{W} \times \mathtt{W}\backslash\{\epsilon\}) \cup (\mathtt{R}\backslash\mathtt{C} \times \mathtt{R}\backslash\{\epsilon\})$ and $L^\omega(\mathcal{B}) = ((\mathtt{W}\backslash\{\epsilon\})r)^\omega$.*

The subset S is sufficient for the inclusion problem $L^\omega(\mathcal{A}) \subseteq M$ as shown next.

Proposition 16. *$L^\omega(\mathcal{A}) \subseteq M \iff \forall (u,v) \in S,\ uv^\omega \in M$.*

Proof. By Theorem 12 it suffices to show that $uv^\omega \in L^\omega(\mathcal{A}) \iff \exists (u',v') \in S,\ uv^\omega = u'v'^\omega$.

$\Rightarrow$: Let $uv^\omega \in L^\omega(\mathcal{A})$. By Proposition 15 we can assume that $(u,v) \in \mathtt{Ld}$.

- If $(u,v) \in \mathtt{C} \times \mathtt{C}$ then an accepting run of $\mathcal{A}$ on uv^ω can be factored as

  $$(q_I,\bot) \vdash^{*u} (q_0,\bot) \vdash^{*v} (q_1,\bot) \vdash^{*v} \cdots$$

  and there is $p \in Q$ such that $q_m = p$ for infinitely many $m \in \mathbb{N}$. Since this run passes from final states infinitely often, there are $m, k \in \mathbb{N}$ such that $(q_I,\bot) \vdash^{*uv^m} (p,\bot)$ and $(p,\bot) \vdash^{\circledast v^k} (p,\bot)$. For $u' \triangleq uv^m$ and $v' \triangleq v^k$, we have that $(u',v') \in L_{q_I,p|\mathtt{C}} \times L^\circledast_{p,p|\mathtt{C}}$.

- If $(u,v) \in \mathtt{U_c} \times \mathtt{R}$ then an accepting run of $\mathcal{A}$ on uv^ω can be factored as

  $$(q_I,\bot) \vdash^{*u} (q_0,w_0) \vdash^{*v} (q_1,w_0w_1) \vdash^{*v} \cdots\ ,$$

  where $w_j \in \Gamma^*$ for all $j \in \mathbb{N}$ and there is $p \in Q$ such that $q_m = p$ for infinitely many $m \in \mathbb{N}$. Since $v \in \mathtt{R}$ the VPA never pops any symbols from $w_0 \cdots w_j$ while reading v in $(q_j, w_0 \cdots w_j) \vdash^{*v} (q_{j+1}, w_0 \cdots w_{j+1})$ and so we have $(q_j,\bot) \vdash^{*v} (q_{j+1}, w_{j+1})$. Since an accepting run passes from final states infinitely often, we deduce that there are $m, k \in \mathbb{N}$ such that $(q_I,\bot) \vdash^{*uv^m} (p, w_0 \cdots w_m)$ and $(p,\bot) \vdash^{\circledast v^k} (p, w_{m+1} \cdots w_{m+k})$. For $u' \triangleq uv^m$ and $v' \triangleq v^k$, we have that $(u',v') \in L_{q_I,p|\mathtt{U_c}} \times L^\circledast_{p,p|\mathtt{R}}$.

$\Leftarrow$: Let $(u,v) \in S$. There is $p \in Q$ such that $(u,v) \in L_{q_I,p|\mathtt{C}} \times L^\circledast_{p,p|\mathtt{C}}$ or $(u,v) \in L_{q_I,p|\mathtt{U_c}} \times L^\circledast_{p,p|\mathtt{R}}$. Thus, there are two sequences of transitions $(q_I,\bot) \vdash^{*u} (p,z)$ and $(p,\bot) \vdash^{\circledast v} (p,w)$ for some $z, w \in \Gamma^*$. If $(u,v) \in \mathtt{C} \times \mathtt{C}$ then $z = w = \bot$. Thus, $(q_I,\bot) \vdash^{*u} (p,\bot) \vdash^{\circledast v} (p,\bot) \vdash^{\circledast v} \cdots$ is an accepting run of $\mathcal{A}$ on uv^ω. If $(u,v) \in \mathtt{U_c} \times \mathtt{R}$ then $(p,w') \vdash^{*v} (p,w'w)$ for every $w' \in \Gamma^*$ and in particular we have $(p,z) \vdash^{*v} (p,zw)$. Therefore, $(q_I,\bot) \vdash^{*u} (p,z) \vdash^{\circledast v} (p,zw) \vdash^{\circledast v} (p,zww) \cdots$ is an accepting run of $\mathcal{A}$ on uv^ω. $\square$

7.2.2 Reduction to a Finite Basis

Next, we fix a pair of M-preserving well-quasiorders $\leqslant, \preccurlyeq$ and show the existence of a subset S_{finite} such that Equation (†) holds. Since $\leqslant \times \preccurlyeq$ is a well-quasiorder, there exist two finite bases S_1 and S_2 for $S_{|\mathtt{C}\times\mathtt{C}}$ and $S_{|\mathtt{U_c}\times\mathtt{R}}$ respectively, w.r.t. $\leqslant \times \preccurlyeq$. We define S_{finite} to be the union of such sets S_1, S_2, viz., $S_{\text{finite}} \triangleq S_1 \cup S_2 \subseteq S$. We have that: $\forall (u,v) \in S,\ uv^\omega \in M \implies \forall (u,v) \in S_{\text{finite}},\ uv^\omega \in M$. We now turn to the converse implication. Assume that $\forall (u,v) \in S_{\text{finite}},\ uv^\omega \in M$. Let $(u,v) \in S$. If $(u,v) \in S_{|\mathtt{C}\times\mathtt{C}}$ then there is $(u_0,v_0) \in S_1$ such that $(u_0,v_0) \leqslant \times \preccurlyeq (u,v)$. Since $S_1 \subseteq S_{|\mathtt{C}\times\mathtt{C}} \subseteq \mathtt{C}\times\mathtt{C}$ we have that $(u_0,v_0),(u,v) \in \mathtt{C}\times\mathtt{C}$. Since $u_0 v_0^\omega \in M$ and the pair $\leqslant, \preccurlyeq$ is M-preserving, we conclude that $uv^\omega \in M$. The case $(u,v) \in S_{|\mathtt{U_c}\times\mathtt{R}}$ proceeds analogously. It follows that $\forall (u,v) \in S,\ uv^\omega \in M \impliedby \forall (u,v) \in S_{\text{finite}},\ uv^\omega \in M$. Hence, we derive Equation (†) using Proposition (16).

7.3 Fixpoint Characterization

In this section, we present a least fixpoint characterization of S for the VPA $\mathcal{A} = (Q, q_I, \Gamma, \delta, F)$.

To this end, we work with the complete lattice $(\wp(\Sigma^*)^{n\cdot|Q|^2}, \subseteq \times \cdots \times \subseteq)$, where $n \in \{4,6\}$, and each Cartesian product consists of $n \cdot |Q|^2$ factors. Given a $n\cdot|Q|^2$-dimensional vector X and a $|Q|^2$-dimensional vector Y on $\wp(\Sigma^*)$ we write $X_{i,p,q}$, for the (i,p,q)-component of X and $Y_{p,q}$ for the (p,q)-component of Y. We define the following equations where $X, X' \in \wp(\mathtt{W})^{|Q|^2}$, $Y, Y' \in \wp(\mathtt{C})^{|Q|^2}$, $Z, Z' \in \wp(\mathtt{R})^{|Q|^2}$, and $T \in \wp(\mathtt{U_c})^{|Q|^2}$:

$$W(X) = \langle L_{p,q_{|(\Sigma_i \cup \{\epsilon\})}} \cup \bigcup_{\substack{(p,c,p',\gamma)\in\delta_c,\\ (q',r,\gamma,q)\in\delta_r}} cX_{p',q'}r \cup \bigcup_{q'\in Q} X_{p,q'}\ X_{q',q} \rangle_{p,q\in Q}$$

$$C(X,Y) = \langle L_{p,q_{|\Sigma_r}} \cup X_{p,q} \cup \bigcup_{q'\in Q} Y_{p,q'}Y_{q',q} \rangle_{p,q\in Q}$$

$$R(X,Z) = \langle L_{p,q_{|\Sigma_c}} \cup X_{p,q} \cup \bigcup_{q'\in Q} Z_{p,q'}Z_{q',q} \rangle_{p,q\in Q}$$

$$U(Y,Z,T) = \langle L_{p,q_{|\Sigma_c}} \cup \bigcup_{p',q'\in Q,} Y_{p,p'}T_{p',q'}Z_{q',q} \rangle_{p,q\in Q}$$

$$W_{\circledast}(X,X') = \langle L^{\circledast}_{p,q_{|\Sigma_i}} \cup \bigcup_{\substack{(p,c,p',\gamma)\in\delta_c,\\ (q',r,\gamma,q)\in\delta_r,\\ \{p,q\}\cap F\neq\emptyset}} cX_{p',q'}r \cup \bigcup_{\substack{(p,c,p',\gamma)\in\delta_c,\\ (q',r,\gamma,q)\in\delta_r\\ \{p,q\}\cap F=\emptyset}} cX'_{p',q'}r \cup \bigcup_{q'\in Q} (X'_{p,q'}X_{q',q} \cup X_{p,q'}X'_{q',q}) \rangle_{p,q\in Q}$$

$$C_{\circledast}(X',Y,Y') = \langle L^{\circledast}_{p,q_{|\Sigma_r}} \cup X'_{p,q} \cup \bigcup_{q'\in Q} (Y'_{p,q'}Y_{q',q} \cup Y_{p,q'}Y'_{q',q}) \rangle_{p,q\in Q}$$

$$R_{\circledast}(X',Z,Z') = \langle L^{\circledast}_{p,q_{|\Sigma_c}} \cup X'_{p,q} \cup \bigcup_{q'\in Q} (Z'_{p,q'}Z_{q',q} \cup Z_{p,q'}Z'_{q',q}) \rangle_{p,q\in Q}\ .$$

The equations W, C, R and U are used to obtain the set of words in $\mathtt{W}$, $\mathtt{C}$, $\mathtt{R}$ and $\mathtt{U_c}$, respectively, that connects two configurations of $\mathcal{A}$. The equations $W_{\circledast}$, $C_{\circledast}$ and $R_{\circledast}$ refine W, C and R by filtering out words not visiting final states. In turn we define the functions $f_{\mathcal{A}}$ and $r_{\mathcal{A}}$ used to

obtain the prefixes u and the periods v for the decompositions $(u, v) \in S$. Define

$$\begin{aligned} f_{\mathcal{A}} \colon \wp(\Sigma^*)^{4\cdot|Q|^2} &\longrightarrow \wp(\Sigma^*)^{4\cdot|Q|^2} \\ (X, Y, Z, T) &\longmapsto (W(X), C(X, Y), R(X, Z), U(Y, Z, T)) \end{aligned}$$

for the prefixes, and for the periods define

$$\begin{aligned} r_{\mathcal{A}} \colon \wp(\Sigma^*)^{6\cdot|Q|^2} &\longrightarrow \wp(\Sigma^*)^{6\cdot|Q|^2} \\ (X, Y, Z, X', Y', Z') &\longmapsto \left(W(X), C(X,Y), R(X,Z), W_{\circledast}(X,X'), C_{\circledast}(X',Y,Y'), R_{\circledast}(X',Z,Z')\right) . \end{aligned}$$

Note that the function $f_{\mathcal{A}}$ (resp. $r_{\mathcal{A}}$) is increasing and the supremum of the ascending sequence of its Kleene iterates starting at the bottom value $\vec{\emptyset} \triangleq (\emptyset, \ldots, \emptyset)$ of dimension $4 \cdot |Q|^2$ (resp. $6 \cdot |Q|^2$) is the vector $(\Lambda_{|\mathtt{W}}, \Lambda_{|\mathtt{C}}, \Lambda_{|\mathtt{R}}, \Lambda_{|\mathtt{U_c}})$ (resp. $(\Lambda_{|\mathtt{W}}, \Lambda_{|\mathtt{C}}, \Lambda_{|\mathtt{R}}, \Lambda^{\circledast}_{|\mathtt{W}}, \Lambda^{\circledast}_{|\mathtt{C}}, \Lambda^{\circledast}_{|\mathtt{R}})$), where $\Lambda_{|\mathtt{J}} \triangleq \langle L_{p,q|\mathtt{J}} \rangle_{p,q\in Q}$ and $\Lambda^{\circledast}_{|\mathtt{J}} \triangleq \langle L^{\circledast}_{p,q|\mathtt{J}} \rangle_{p,q\in Q}$ for $\mathtt{J} \in \{\mathtt{W}, \mathtt{C}, \mathtt{R}, \mathtt{U_c}\}$. Therefore, by the Knaster–Tarski Theorem we obtain the following proposition.

Proposition 17. $\operatorname{lfp} f_{\mathcal{A}} = (\Lambda_{|W}, \Lambda_{|C}, \Lambda_{|R}, \Lambda_{|U_c})$ *and* $\operatorname{lfp} r_{\mathcal{A}} = (\Lambda_{|W}, \Lambda_{|C}, \Lambda_{|R}, \Lambda^{\circledast}_{|W}, \Lambda^{\circledast}_{|C}, \Lambda^{\circledast}_{|R})$.

Proof. We prove that the supremum of the ascending sequence of Kleene iterates of $f_{\mathcal{A}}$ (resp. $r_{\mathcal{A}}$) starting at the bottom value $\vec{\emptyset}$ of dimension $4 \cdot |Q|^2$ (resp. $6 \cdot |Q|^2$) is the vector $(\Lambda_{|\mathtt{W}}, \Lambda_{|\mathtt{C}}, \Lambda_{|\mathtt{R}}, \Lambda_{|\mathtt{U_c}})$ (resp. $(\Lambda_{|\mathtt{W}}, \Lambda_{|\mathtt{C}}, \Lambda_{|\mathtt{R}}, \Lambda^{\circledast}_{|\mathtt{W}}, \Lambda^{\circledast}_{|\mathtt{C}}, \Lambda^{\circledast}_{|\mathtt{R}})$). By observing that for all $X \subseteq^{|Q|^2} \Lambda_{|\mathtt{W}}$, $X' \subseteq^{|Q|^2} \Lambda^{\circledast}{}_{|\mathtt{W}}$, $Y \subseteq^{|Q|^2} \Lambda_{|\mathtt{C}}$, $Y' \subseteq^{|Q|^2} \Lambda^{\circledast}{}_{|\mathtt{C}}$, $Z \subseteq^{|Q|^2} \Lambda_{|\mathtt{R}}$, $Z' \subseteq^{|Q|^2} \Lambda^{\circledast}{}_{|\mathtt{R}}$ and $T \subseteq^{|Q|^2} \Lambda_{|\mathtt{U_c}}$ we have

- $W(X) \subseteq^{|Q|^2} \Lambda_{|\mathtt{W}}$, $C(X, Y) \subseteq^{|Q|^2} \Lambda_{|\mathtt{C}}$, $R(X, Z) \subseteq^{|Q|^2} \Lambda_{|\mathtt{R}}$, $U(Y, T, Z) \subseteq^{|Q|^2} \Lambda_{|\mathtt{U_c}}$,
- $W_{\circledast}(X, X') \subseteq^{|Q|^2} \Lambda^{\circledast}{}_{|\mathtt{W}}$, $C_{\circledast}(X', Y, Y') \subseteq^{|Q|^2} \Lambda^{\circledast}{}_{|\mathtt{C}}$, $R_{\circledast}(X', Z, Z') \subseteq^{|Q|^2} \Lambda^{\circledast}{}_{|\mathtt{R}}$,

we deduce that for every $n \in \mathbb{N}$ we have ${f_{\mathcal{A}}}^n(\vec{\emptyset}) \subseteq^{4\cdot|Q|^2} (\Lambda_{|\mathtt{W}}, \Lambda_{|\mathtt{C}}, \Lambda_{|\mathtt{R}}, \Lambda_{|\mathtt{U_c}})$ and ${r_{\mathcal{A}}}^n(\vec{\emptyset}) \subseteq^{6\cdot|Q|^2} (\Lambda_{|\mathtt{W}}, \Lambda_{|\mathtt{C}}, \Lambda_{|\mathtt{R}}, \Lambda^{\circledast}_{|\mathtt{W}}, \Lambda^{\circledast}_{|\mathtt{C}}, \Lambda^{\circledast}_{|\mathtt{R}})$. Next we prove the converse inclusions.

- For every $n \in \mathbb{N}$ we define $(X_n, Y_n, Z_n, U_n) \triangleq {f_{\mathcal{A}}}^n(\vec{\emptyset})$. First we show by induction on the length of the words that for all $p, q \in Q$ and $x \in L_{p,q|\mathtt{W}}$ there is $n \in \mathbb{N}$ such that $x \in (X_n)_{p,q}$. Let $p, q \in Q$ and $x \in L_{p,q|\mathtt{W}}$. By Definition 3.6.1 we have that $x \in \Sigma_i \cup \{\epsilon\}$, or $x = cx'r$ with $c \in \Sigma_c$, $x' \in \mathtt{W}$ and $r \in \Sigma_r$, or $x = x'x''$ with $x', x'' \in \mathtt{W}\backslash\{\epsilon\}$. If $x \in \Sigma_i \cup \{\epsilon\}$ then $x \in (X_1)_{p,q}$. Let $2 \leq |x|$. Assume that for all $p', q' \in Q$ and $x' \in L_{p',q'|\mathtt{W}}$ such that $|x'| < |x|$ we have $x' \in (X_n)_{p',q'}$ for some $n \in \mathbb{N}$. If $x = cx'r$ with $c \in \Sigma_c$, $x' \in \mathtt{W}$ and $r \in \Sigma_r$, then there are $(p, c, p', \gamma) \in \delta_c$ and $(q', r, \gamma, q) \in \delta_r$ such that $(p', \gamma) \vdash^{*x} (q', \gamma)$. Since $x' \in \mathtt{W}$ we deduce that $x' \in L_{p',q'|\mathtt{W}}$. Hence, by induction assumption we have that $x' \in (X_n)_{p',q'}$ for some $n \in \mathbb{N}$. Thus, $x \in c(X_n)_{p',q'}r \subseteq (X_{n+1})_{p,q}$. If $x = x'x''$ with $x', x'' \in \mathtt{W}\backslash\{\epsilon\}$, then $x' \in L_{p,q'|\mathtt{W}}$ and $x'' \in L_{q'q|\mathtt{W}}$ for some $q' \in Q$. Using the induction assumption we find that $x \in (X_n)_{p,q'}(X_m)_{q',p}$ for some $n, m \in \mathbb{N}$. We can assume that $n \leq m$. Since, $X_n \subseteq^{|Q|^2} X_m$ we find that $x \in (X_m)_{p,q'}(X_m)_{q',q} \subseteq (X_{m+1})_{p,q}$. Thus, $\Lambda_{|\mathtt{W}} \subseteq^{4\cdot|Q|^2} \bigcup_{n\in\mathbb{N}} X_n$. The proofs that $\Lambda_{|\mathtt{C}} \subseteq^{4\cdot|Q|^2} \bigcup_{n\in\mathbb{N}} Y_n$, $\Lambda_{|\mathtt{R}} \subseteq^{4\cdot|Q|^2} \bigcup_{n\in\mathbb{N}} Z_n$ and $\Lambda_{|\mathtt{U_c}} \subseteq^{4\cdot|Q|^2} \bigcup_{n\in\mathbb{N}} U_n$ proceed similarly.
- For every $n \in \mathbb{N}$ we define $(X_n, Y_n, Z_n, X^{\circledast}_n, Y^{\circledast}_n, Z^{\circledast}_n) \triangleq {r_{\mathcal{A}}}^n(\vec{\emptyset})$. The cases of $\Lambda_{|\mathtt{W}} \subseteq^{|Q|^2} \bigcup_{n\in\mathbb{N}} X_n$, $\Lambda_{|\mathtt{C}} \subseteq^{|Q|^2} \bigcup_{n\in\mathbb{N}} Y_n$ and $\Lambda_{|\mathtt{R}} \subseteq^{|Q|^2} \bigcup_{n\in\mathbb{N}} Z_n$ proceed as done previously. We show

by induction on the length of the words that for all $p, q \in Q$ and $x \in L^{\circledast}_{p,q|\mathtt{W}}$ there is $n \in \mathbb{N}$ such that $x \in (X^{\circledast}_n)_{p,q}$. Let $p, q \in Q$ and $x \in L^{\circledast}_{p,q|\mathtt{W}}$. We have $x \neq \epsilon$. If $x \in \Sigma_i$ then $x \in (X^{\circledast}_1)_{p,q}$. Let $2 \leq |x|$. Assume that for all $p', q' \in Q$ and $x' \in L^{\circledast}_{p',q'|\mathtt{W}}$ such that $|x'| < |x|$ we have $x' \in (X^{\circledast}_n)_{p',q'}$ for some $n \in \mathbb{N}$. If $x = cx'r$ with $c \in \Sigma_c$, $x' \in \mathtt{W}$ and $r \in \Sigma_r$, then there are $(p, c, p', \gamma) \in \delta_c$ and $(q', r, \gamma, q) \in \delta_r$ such that $x' \in L_{p',q'|\mathtt{W}}$ and $p \in F$, or $q \in F$ or $x' \in L^{\circledast}_{p',q'|\mathtt{W}}$. Since $\Lambda_{|\mathtt{W}} \subseteq^{|Q|^2} \bigcup_{n\in\mathbb{N}} X_n$ there is $n \in \mathbb{N}$ such that $x' \in (X_n)_{p',q'}$. If $p \in F$ or $q \in F$ then $x \in c(X_n)_{p',q'}r \subseteq (X^{\circledast}_{n+1})_{p,q}$. If $x' \in L^{\circledast}_{p',q'|\mathtt{W}}$ we conclude similarly using the induction hypothesis. If $x = x'x''$ with $x', x'' \in \mathtt{W}\backslash\{\epsilon\}$ and $|x'|, |x''| < |x|$, then $x' \in L_{p,q'|\mathtt{W}}$ and $x'' \in L_{q'q|\mathtt{W}}$ for some $q' \in Q$ and we have that $x' \in L^{\circledast}_{p,q'|\mathtt{W}}$ or $x'' \in L^{\circledast}_{q'q|\mathtt{W}}$. Using the induction assumption we find that $x \in \bigcup_{q'\in Q}\Big((X_n)_{p,q'}(X^{\circledast}_n)_{q',q} \cup (X^{\circledast}_n)_{p,q'}(X_n)_{q',q}\Big) \subseteq (X^{\circledast}_{n+1})_{p,q}$ for some $n \in \mathbb{N}$. Thus, $\Lambda^{\circledast}_{|\mathtt{W}} \subseteq^{|Q|^2} \bigcup_{n\in\mathbb{N}}, X^{\circledast}_n$. The proofs that $\Lambda^{\circledast}_{|\mathtt{C}} \subseteq^{|Q|^2} \bigcup_{n\in\mathbb{N}} Y^{\circledast}_n$ and $\Lambda^{\circledast}_{|\mathtt{R}} \subseteq^{|Q|^2} \bigcup_{n\in\mathbb{N}} Z^{\circledast}_n$ are similar.

□

Finally, by Proposition 17, we obtain the desired fixpoint characterization of S:

$$S = \bigcup_{p\in Q}\Big(\Big((\mathrm{lfp}\, f_{\mathcal{A}})_{2,q_I,p} \times (\mathrm{lfp}\, r_{\mathcal{A}})_{5,p,p}\Big) \cup \Big((\mathrm{lfp}\, f_{\mathcal{A}})_{4,q_I,p} \times (\mathrm{lfp}\, r_{\mathcal{A}})_{6,p,p}\Big)\Big) \ . \tag{7.1}$$

Example 20. *We derive from the VPA $\mathcal{A}$ depicted in Fig. 3.2 the following functions*

$$\begin{aligned} W(X) &\triangleq \{\epsilon\} \cup cXr \cup XX, & C(X,Y) &\triangleq X \cup YY, \\ R(X,Z) &\triangleq \{c\} \cup X \cup ZZ, & U(Y,Z,T) &\triangleq \{c\} \cup YTZ \ . \end{aligned}$$

Hence, we obtain the function

$$\begin{aligned} &f_{\mathcal{A}} \colon \wp(\Sigma^*)^4 \longrightarrow \wp(\Sigma^*)^4 \\ &(X,Y,Z,T) \longmapsto (W(X), C(X,Y), R(X,Z), U(Y,Z,T)) \ . \end{aligned}$$

The first three iterates of the least fixpoint computation of $\mathrm{lfp}\, f_{\mathcal{A}}$ *are given by*

$$\begin{aligned} f_{\mathcal{A}}(\vec{\emptyset}) &= (\{\epsilon\}, \emptyset, \{c\}, \{c\}), \\ {f_{\mathcal{A}}}^2(\vec{\emptyset}) &= (\{\epsilon, cr\}, \{\epsilon\}, \{\epsilon, c, c^2\}, \{c\}), \\ {f_{\mathcal{A}}}^3(\vec{\emptyset}) &= (\{\epsilon, cr, c^2r^2, (cr)^2\}, \{\epsilon, cr\}, \{\epsilon, cr, c, c^2, c^3, c^4\}, \{c, c^2, c^3\}) \\ &\vdots \\ \mathrm{lfp}\, f_{\mathcal{A}} &= (\mathtt{W}, \mathtt{W}, \mathtt{R}, \mathtt{R}\backslash\mathtt{C}) \end{aligned}$$

Since the unique state of $\mathcal{A}$ is a final state we have that $L_{q_I,q_I} = L^{\circledast}_{q_I,q_I}$. Consequently, the function $f_{\mathcal{A}}$ suffices to describe both the set of prefixes and the set of periods of S given by $\Big((\mathrm{lfp}\, f_{\mathcal{A}})_2 \times (\mathrm{lfp}\, f_{\mathcal{A}})_2\backslash\{\epsilon\}\Big) \cup \Big((\mathrm{lfp}\, f_{\mathcal{A}})_4 \times (\mathrm{lfp}\, f_{\mathcal{A}})_3\backslash\{\epsilon\}\Big)$.

7.4 Monotonicity Requirements

In order to detect finite bases among the Kleene iterates of the functions defined in the previous section we replace the set inclusion on $\wp(\Sigma^*)$, used so far, with the qo $\sqsubseteq_\ltimes \subseteq \wp(\Sigma^*) \times \wp(\Sigma^*)$ defined by $X \sqsubseteq_\ltimes Y \stackrel{\triangle}{\iff} \forall x \in X, \exists y \in Y, y \ltimes x$. The qo $\sqsubseteq_\ltimes$ leverage the notion of basis: given $X \in \wp(\Sigma^*)$ a subset $Y \subseteq X$ is a basis for X with respect to $\ltimes$ whenever $X \sqsubseteq_\ltimes Y$.

In the following we lift the notion of basis to n-dimensional vectors component-wise and work with the quasiordered sets $(\wp(\Sigma^*)^{n\cdot|Q|^2}, \sqsubseteq_\ltimes^{n\cdot|Q|^2})$, where $n \in \{4, 6\}$ and the ordering $\sqsubseteq_\ltimes^{n\cdot|Q|^2}$ is given by the product $\sqsubseteq_\ltimes \times \cdots \times \sqsubseteq_\ltimes$ of $n \cdot |Q|^2$ factors. Given a pair $\leqslant, \preccurlyeq$ of well-quasiorders, the orderings $\sqsubseteq_\leqslant^{4\cdot|Q|^2}$ and $\sqsubseteq_\preccurlyeq^{6\cdot|Q|^2}$ are used to compare the Kleene iterates of the functions $f_\mathcal{A}$ and $r_\mathcal{A}$, respectively. For them to be apt to detect finite bases for the least fixpoints of these functions the quasiorders $\leqslant$ and $\preccurlyeq$ need to verify some monotonicity conditions.

We introduce the monotonicity conditions $\mathbf{W}, \mathbf{C}, \mathbf{R}, \boldsymbol{C}_\circledast, \mathbf{R}_\circledast$ and $\mathbf{U}$ on a qo $\ltimes \subseteq \Sigma^* \times \Sigma^*$ as follows: for all $u, u' \in \Sigma^*$ such that $u \ltimes u'$

$$
\begin{array}{rl}
(\mathbf{W}) & \text{if } u, u' \in \mathtt{W} \text{ and } c \in \Sigma_c, r \in \Sigma_r \text{ then } cur \ltimes cu'r, \\
(\mathbf{C}) & \text{if } u, u' \in \mathtt{C} \text{ and } s \in \mathtt{C}\ , t \in \Sigma^* \text{ then } sut \ltimes su't, \\
(\mathbf{R}) & \text{if } u, u' \in \mathtt{R} \text{ and } s \in \Sigma^*, t \in \mathtt{R} \text{ then } sut \ltimes su't, \\
(\mathbf{U}) & \text{if } u, u' \in \mathtt{U_c} \text{ and } s \in \mathtt{C}\ , t \in \mathtt{R} \text{ then } sut \ltimes su't, \\
(\mathbf{C}_\circledast) & \text{if } u, u' \in \mathtt{C} \text{ and } s \in \mathtt{C}\ , t \in \mathtt{C} \text{ then } sut \ltimes su't, \\
(\mathbf{R}_\circledast) & \text{if } u, u' \in \mathtt{R} \text{ and } s \in \mathtt{R}\ , t \in \mathtt{R} \text{ then } sut \ltimes su't.
\end{array}
$$

A pair of quasiorders $\leqslant, \preccurlyeq$ is *monotonic* if $\leqslant$ verifies $\mathbf{W}, \mathbf{C}, \mathbf{R}, \mathbf{U}$ and $\preccurlyeq$ verifies $\mathbf{W}, \mathbf{C}_\circledast, \mathbf{R}_\circledast$.

Proposition 18. *Let $\leqslant, \preccurlyeq$ be a pair of well-quasiorders. There is a positive integer n such that ${f_\mathcal{A}}^{n+1}(\vec{\emptyset}) \sqsubseteq_\leqslant^{4\cdot|Q|^2} {f_\mathcal{A}}^n(\vec{\emptyset})$ (resp. ${r_\mathcal{A}}^{n+1}(\vec{\emptyset}) \sqsubseteq_\preccurlyeq^{6\cdot|Q|^2} {r_\mathcal{A}}^n(\vec{\emptyset})$); and, if the pair of well-quasiorders is monotonic then $\operatorname{lfp} f_\mathcal{A} \sqsubseteq_\leqslant^{4\cdot|Q|^2} {f_\mathcal{A}}^n(\vec{\emptyset})$ (resp. $\operatorname{lfp} r_\mathcal{A} \sqsubseteq_\preccurlyeq^{6\cdot|Q|^2} {r_\mathcal{A}}^n(\vec{\emptyset})$).*

Proof. The sequence of Kleene iterates of $f_\mathcal{A}$ is ascending w.r.t. $\sqsubseteq_\leqslant^{4\cdot|Q|^2}$ and since $\leq$ is a well-quasiorder the quasiordered set $(\wp(\Sigma^*)^{4\cdot|Q|^2}, \sqsubseteq_\leqslant^{4\cdot|Q|^2})$ satisfies the ascending chain condition [23]. Therefore, there is a positive integer n such that for every $n' \geq n$ we have ${f_\mathcal{A}}^{n'}(\vec{\emptyset}) \sqsubseteq_\leqslant^{4\cdot|Q|^2} {f_\mathcal{A}}^n(\vec{\emptyset})$ and ${f_\mathcal{A}}^n(\vec{\emptyset}) \sqsubseteq_\leqslant^{4\cdot|Q|^2} {f_\mathcal{A}}^{n'}(\vec{\emptyset})$. Next we show that we can detect such a n using $\sqsubseteq_\leqslant^{4\cdot|Q|^2}$.

Let m be a positive integer such that ${f_\mathcal{A}}^{m+1}(\vec{\emptyset}) \sqsubseteq_\leqslant^{4\cdot|Q|^2} {f_\mathcal{A}}^m(\vec{\emptyset})$. An easy induction that uses Proposition 22 shows that for every $k \geq m$ we have ${f_\mathcal{A}}^{k+1}(\vec{\emptyset}) \sqsubseteq_\leqslant^{4\cdot|Q|^2} {f_\mathcal{A}}^k(\vec{\emptyset})$. Hence, by transitivity of $\sqsubseteq_\leqslant^{4\cdot|Q|^2}$ we deduce that for every $k \geq m$ we have ${f_\mathcal{A}}^k(\vec{\emptyset}) \sqsubseteq_\leqslant^{4\cdot|Q|^2} {f_\mathcal{A}}^m(\vec{\emptyset})$. Since the sequence of Kleene iterates of $f_\mathcal{A}$ is ascending we also have ${f_\mathcal{A}}^m(\vec{\emptyset}) \sqsubseteq_\leqslant^{4\cdot|Q|^2} {f_\mathcal{A}}^k(\vec{\emptyset})$ for every $k \geq m$. We have ${f_\mathcal{A}}^m(\vec{\emptyset}) \subseteq^{4\cdot|Q|^2} \operatorname{lfp} f_\mathcal{A}$ and since $\operatorname{lfp} f_\mathcal{A}$ is the supremum of the sequence of Kleene iterates of $f_\mathcal{A}$ we deduce that $\operatorname{lfp} f_\mathcal{A} \sqsubseteq_\leqslant^{4\cdot|Q|^2} {f_\mathcal{A}}^m(\vec{\emptyset})$. Thus, ${f_\mathcal{A}}^n(\vec{\emptyset})$ is a basis for $\operatorname{lfp} f_\mathcal{A}$. The analogue reasoning apply for $r_\mathcal{A}$.

□

Each Kleene iterate of $f_\mathcal{A}$ and $r_\mathcal{A}$ is computable and given a decidable qo $\ltimes$ on Σ^* and two finite sets $X, Y \subseteq \Sigma^*$ it is decidable whether $X \sqsubseteq_\ltimes Y$ holds. Thus, given a monotonic pair

$\leqslant, \preccurlyeq$ of decidable well-quasiorders, by Proposition 18, we can compute a finite basis for $\operatorname{lfp} f_{\mathcal{A}}$ w.r.t. $\leqslant$ and a finite basis for $\operatorname{lfp} r_{\mathcal{A}}$ w.r.t. $\preccurlyeq$. Hence, by Equation (7.1) we can compute a finite basis for S w.r.t. $\leqslant \times \preccurlyeq$.

7.5 Quasiorders for ω-VPL

In the following we present two M-suitable pairs of quasiorders to solve the inclusion problem $L^\omega(\mathcal{A}) \subseteq M$.

7.5.1 A State-based Pair

Given a VPA $\mathcal{B} = (\hat{Q}, \hat{q_I}, \hat{\Gamma}, \hat{\delta}, \hat{F})$ we associate to each word $u \in \Sigma^*$ its *context* $ctx^{\mathcal{B}}(u)$ and *final context* $ctx^{\mathcal{B}}_{\circledast}(u)$ *in* $\mathcal{B}$ as follows:

$$ctx^{\mathcal{B}}(u) \triangleq \{(p,q) \in \hat{Q}^2 \mid \exists w \in \hat{\Gamma}^*, (p,\bot) \vdash^{*u} (q,w)\},$$
$$ctx^{\mathcal{B}}_{\circledast}(u) \triangleq \{(p,q) \in \hat{Q}^2 \mid \exists w \in \hat{\Gamma}^*, (p,\bot) \vdash^{\circledast u} (q,w)\} \ .$$

Hence, we define the following quasiorders on words in Σ^*:

$$u \leqslant^{\mathcal{B}} u' \overset{\triangle}{\iff} ctx^{\mathcal{B}}(u) \subseteq ctx^{\mathcal{B}}(u'), \quad u \preccurlyeq^{\mathcal{B}} u' \overset{\triangle}{\iff} u \leqslant^{\mathcal{B}} u' \wedge ctx^{\mathcal{B}}_{\circledast}(u) \subseteq ctx^{\mathcal{B}}_{\circledast}(u') \ .$$

Proposition 19. *Let $\mathcal{B}$ be a VPA. The pair $\leqslant^{\mathcal{B}}, \preccurlyeq^{\mathcal{B}}$ is $L^\omega(\mathcal{B})$-suitable.*

Proof. The quasiorders $\leqslant^{\mathcal{B}}$ and $\preccurlyeq^{\mathcal{B}}$ are decidable well-quasiorders since $\hat{Q}$ is a finite set. We show that the pair $\leqslant^{\mathcal{B}}, \preccurlyeq^{\mathcal{B}}$ is monotonic.

(W) Let $u, u' \in \mathtt{W}$ such that $u \leqslant^{\mathcal{B}} u'$, $c \in \Sigma_c$ and $r \in \Sigma_r$. We show that $cur \leqslant^{\mathcal{B}} cu'r$. Let $(p,q) \in ctx^{\mathcal{B}}(cur)$. Since $u, cur \in \mathtt{W}$, there is $\gamma \in \Gamma$ and $(p',q') \in ctx^{\mathcal{B}}(u)$ such that $(p,\bot) \vdash^c (p',\gamma)$ and $(q',\gamma) \vdash^r (q,\bot)$. Since $u \leqslant^{\mathcal{B}} u'$ and $u' \in \mathtt{W}$ we have $(p',\bot) \vdash^{*u'} (q',\bot)$. Hence, $(p,\bot) \vdash^{*cu'r} (q,\bot)$, that is $(p,q) \in ctx^{\mathcal{B}}(cu'r)$. Thus, $cur \leqslant^{\mathcal{B}} cu'r$.

We assume that $u \preccurlyeq^{\mathcal{B}} u'$ and show that $cur \preccurlyeq^{\mathcal{B}} cu'r$. We have $ctx^{\mathcal{B}}(cur) \subseteq ctx^{\mathcal{B}}(cu'r)$ and if $(p,q) \in ctx^{\mathcal{B}}_{\circledast}(cur)$ then $(p,\bot) \vdash^{\circledast c} (p',\gamma)$, or $(p',\bot) \vdash^{\circledast u} (q',\bot)$ or $(q',\gamma) \vdash^{\circledast r} (q,\bot)$. Thus, $(p,q) \in ctx^{\mathcal{B}}_{\circledast}(cu'r)$.

(C) Let $u, u' \in \mathtt{C}$ such that $u \leqslant^{\mathcal{B}} u'$, $s \in \mathtt{C}$ and $t \in \Sigma^*$. We show that $sut \leqslant^{\mathcal{B}} su't$. Let $(p,q) \in ctx^{\mathcal{B}}(sut)$. We have $(p,\bot) \vdash^{*sut} (q,w)$ for some $w \in \Gamma^*$. Since $s \in \mathtt{C}$ and $u \in \mathtt{C}$, there are two states $p', q' \in Q$ such that $(p,\bot) \vdash^{*s} (p',\bot)$, $(p',\bot) \vdash^{*u} (q',\bot)$ and $(q',\bot) \vdash^{*t} (q,w)$. Since $u \leqslant^{\mathcal{B}} u'$, we also have $(p',\bot) \vdash^{*u'} (q',\bot)$. Hence, $(p,\bot) \vdash^{*su't} (q,w)$ and $(p,q) \in ctx^{\mathcal{B}}(su't)$.

We assume that $u \preccurlyeq^{\mathcal{B}} u'$ and show that $sut \preccurlyeq^{\mathcal{B}} su't$. We have $ctx^{\mathcal{B}}(sut) \subseteq ctx^{\mathcal{B}}(su't)$ and if $(p,q) \in ctx^{\mathcal{B}}_{\circledast}(sut)$ then $(p,\bot) \vdash^{\circledast s} (p',\bot)$ or $(p',\bot) \vdash^{\circledast u} (q',\bot)$ or $(q',\bot) \vdash^{\circledast t} (q,w)$. If $(p',\bot) \vdash^{\circledast u} (q',\bot)$ then since $u \preccurlyeq^{\mathcal{B}} u'$ we also have $(p',\bot) \vdash^{\circledast u'} (q',\bot)$. In any case, we deduce that $(p,\bot) \vdash^{\circledast su't} (q,w)$, that is $(p,q) \in ctx^{\mathcal{B}}_{\circledast}(su't)$. Hence, $sut \preccurlyeq^{\mathcal{B}} su't$.

(R) Let $u, u' \in \mathtt{R}$ $s \in \Sigma^*$ and $t \in \mathtt{R}$. We assume that $u \leqslant^{\mathcal{A}} u'$ and show that $sut \leqslant^{\mathcal{B}} su't$. Let $(p,q) \in ctx^{\mathcal{B}}(sut)$. We have $(p,\bot) \vdash^{*sut} (q,w)$ for some $w \in \Gamma^*$. Since $u, t \in \mathtt{R}$, we deduce that there are two states $p', q' \in Q$ such that $(p,\bot) \vdash^{*s} (p',w')$, $(p',\bot) \vdash^{*u} (q',w'')$, $(q',\bot) \vdash^{*t} (q,w''')$ and $w = w'''w''w'$. Since $u \leqslant^{\mathcal{A}} u'$, we also have $(p',\bot) \vdash^{*u'} (q',z)$ for some $z \in \Gamma^*$. Hence we have $(p,\bot) \vdash^{*su't} (q, w'zw''')$ and $(p,q) \in ctx^{\mathcal{B}}(su't)$. Hence,$sut \leqslant^{\mathcal{B}} su't$.

We assume that $u \not\preccurlyeq^{\mathcal{B}} u'$ and show that $sut \not\preccurlyeq^{\mathcal{B}} su't$. We have $ctx^{\mathcal{B}}(sut) \subseteq ctx^{\mathcal{B}}(su't)$ and by a similar reasoning as in 2. we find that $ctx^{\mathcal{B}}_{\circledast}(sut) \subseteq ctx^{\mathcal{B}}_{\circledast}(su't)$. Hence, $sut \not\preccurlyeq^{\mathcal{B}} su't$.

(U) Let $u, u' \in \Sigma^*$, $s \in \mathtt{C}$ and $t \in \mathtt{R}$. We assume that $u \leqslant^{\mathcal{B}} u'$ and show that $sut \leqslant^{\mathcal{B}} su't$. Let $(p,q) \in ctx^{\mathcal{B}}(sut)$. We have $(p,\bot) \vdash^{*sut} (q,w)$ for some $w \in \Gamma^*$. Since $s \in \mathtt{C}$ and $t \in \mathtt{R}$, there are two states $p', q' \in Q$ and two words $w', w'' \in \Gamma^*$ such that $(p,\bot) \vdash^{*s} (p',\bot)$, $(p',\bot) \vdash^{*u} (q',w')$ and $(q',\bot) \vdash^{*t} (q,w'')$, with $w = w'w''$. Since $u \leqslant^{\mathcal{B}} u'$, we also have $(p',\bot) \vdash^{*u'} (q',z)$ for some $z \in \Gamma^*$. Hence, we have $(p,\bot) \vdash^{*su't} (q,zw'')$ and $(p,q) \in ctx^{\mathcal{B}}(su't)$. Hence, $sut \leqslant^{\mathcal{B}} su't$.

We assume that $u \not\preccurlyeq^{\mathcal{B}} u'$ and show that $sut \not\preccurlyeq^{\mathcal{B}} su't$. We have $ctx^{\mathcal{B}}(sut) \subseteq ctx^{\mathcal{B}}(su't)$ and by a similar reasoning as in 2. we find that $ctx^{\mathcal{B}}_{\circledast}(sut) \subseteq ctx^{\mathcal{B}}_{\circledast}(su't)$. Hence, $sut \not\preccurlyeq^{\mathcal{B}} su't$.

We now show that the pair $\leqslant^{\mathcal{B}}, \not\preccurlyeq^{\mathcal{B}}$ is $L^\omega(\mathcal{B})$-preserving. Let $(u,v),(u',v') \in \mathtt{Ld}$ such that $(u,v),(u',v') \in \mathtt{C}\times\mathtt{C}$ or $(u,v),(u',v') \in \mathtt{U_c}\times\mathtt{R}$ and such that $u \leqslant^{\mathcal{B}} u'$, $v \not\preccurlyeq^{\mathcal{B}} v'$ and $uv^\omega \in L^\omega(\mathcal{B})$. Since $(u,v) \in \mathtt{Ld}$ an accepting run for uv^ω on $\mathcal{B}$ can be factored by $(\hat{q_I},\bot) \vdash^{*u} (q,w_1) \vdash^{*v^n} (p,w_2) \vdash^{\circledast v^m} (p,w_3)$ for some $n, m \in \mathbb{N}$, with $m \neq 0$, $q, p \in \hat{Q}$, and words $w_1, w_2, w_3 \in \Gamma^*$. If $(u,v),(u',v') \in \mathtt{C}\times\mathtt{C}$ then $v^k, (v')^k \in \mathtt{C}$ for all $k \in \mathbb{N}$ and $w_1 = w_2 = w_3 = \bot$. Since $\not\preccurlyeq^{\mathcal{B}}$ satisfies the monotonicity condition $\mathbf{C}_\circledast$ and $v \not\preccurlyeq^{\mathcal{B}} v'$ we deduce that $v^k \not\preccurlyeq^{\mathcal{B}} (v')^k$, for all $k \in \mathbb{N}$. Therefore, we have $(q,\bot) \vdash^{*(v')^n} (p,\bot)$ and $(p,\bot) \vdash^{\circledast (v')^m} (p,\bot)$. Since $u \leqslant^{\mathcal{B}} u'$ we have $(\hat{q_I},\bot) \vdash^{*u'} (q,\bot)$. Hence, there is an accepting run of $\mathcal{B}$ on $u'v'^\omega$. Likewise, if $(u,v),(u',v') \in \mathtt{U_c}\times\mathtt{R}$ then $v^k, (v')^k \in \mathtt{R}$ for all $k \in \mathbb{N}$ and since $\not\preccurlyeq^{\mathcal{B}}$ satisfies the monotonicity condition $\mathbf{R}_\circledast$ and $v \not\preccurlyeq^{\mathcal{B}} v'$, we deduce that $v^k \not\preccurlyeq^{\mathcal{B}} (v')^k$, for all $k \in \mathbb{N}$. Hence, we deduce an accepting run of $\mathcal{B}$ on $u'v'^\omega$.

□

Example 21. *Consider the pair of quasiorders $\leqslant^{\mathcal{B}}, \not\preccurlyeq^{\mathcal{B}}$ derived as explained above from $\mathcal{B}$ (Fig. 3.2) and the set $S = (\mathtt{W}\times\mathtt{W}\backslash\{\epsilon\}) \cup (\mathtt{R}\backslash\mathtt{C}\times\mathtt{R}\backslash\{\epsilon\})$ from Example 19. We have that $ctx^{\mathcal{B}}(\epsilon) = \{(p,p),(q,q)\}$, $ctx^{\mathcal{B}}_{\circledast}(\epsilon) = \{(p,p)\}$, $ctx^{\mathcal{B}}(u) = \{(p,q),(q,q)\}$ and $ctx^{\mathcal{B}}_{\circledast}(u) = \{(p,q)\}$ for every $u \in \mathtt{R}\backslash\{\epsilon\}$. We have that $\{c\}$ is a basis for $\mathtt{R}\backslash\{\epsilon\}$ w.r.t. $\not\preccurlyeq^{\mathcal{B}}$ since $c \not\preccurlyeq^{\mathcal{B}} u$ for every $u \in \mathtt{R}\backslash\{\epsilon\}$. Since $\mathtt{R}\backslash\mathtt{C} \subseteq \mathtt{R}\backslash\{\epsilon\}$ and $\{c\} \subseteq \mathtt{R}\backslash\mathtt{C}$ we deduce that $\{c\}$ is also a basis for $\mathtt{R}\backslash\mathtt{C}$ w.r.t $\leqslant^{\mathcal{B}}$. Similarly we deduce that $\{\epsilon, cr\}$ is basis for $\mathtt{W}$ w.r.t $\leqslant^{\mathcal{B}}$ and that $\{cr\}$ is a basis for $\mathtt{W}\backslash\{\epsilon\}$ w.r.t. $\not\preccurlyeq^{\mathcal{B}}$. Hence, $(\{\epsilon, cr\}\times\{cr\}) \cup (\{c\}\times\{c\})$ is a basis for S w.r.t. $\leqslant^{\mathcal{B}} \times \not\preccurlyeq^{\mathcal{B}}$.*

7.5.2 A Syntactic Pair

Given a ω-VPL M we associate to each word $u \in \Sigma^*$ its *context* $ctx^M(u)$ and *final context* $ctx^M_\circledast(u)$ *in* M as follows:

$$ctx^M(u) \triangleq \{(s,\xi) \in \Sigma^* \times \Sigma^\omega \mid su\xi \in M\},$$
$$ctx^M_\circledast(u) \triangleq \{(s,t) \in \Sigma^* \times \Sigma^* \mid s(ut)^\omega \in M\} \ .$$

At first glance, we are tempted to define the syntactic quasiorders from ctx^M and $ctx^M_\circledast$ in the analogue way we defined the state-based quasiorders from the contexts and final contexts relatively to a VPA. Although, this definition provides a pair of M-preserving quasiorders, it does not guarantee that the pair is M-suitable. To overcome this, we impose the respect of the partition $\mathcal{P} \triangleq \{\mathtt{W}, \mathtt{C}\backslash\mathtt{W}, \mathtt{R}\backslash\mathtt{W}, \mathtt{U_c}\backslash\mathtt{R}\}$ of Σ^*, meaning that two words compare only if they belong to the same subset of $\mathcal{P}$. Additionally, given $\mathtt{J} \in \mathcal{P}$ we compare two words of $\mathtt{J}$ by considering a restriction of their context and final context in M which depends on $\mathtt{J}$. More precisely, we define the qo $\leqslant^M$ on Σ^* as the union $\bigcup_{\mathtt{J}\in\mathcal{P}} \leqslant^M_\mathtt{J}$ where for every $\mathtt{J} \in \mathcal{P}$, the qo $\leqslant^M_\mathtt{J} \subseteq \mathtt{J} \times \mathtt{J}$ is defined by

$$\begin{aligned}
u \leqslant^M_\mathtt{W} u' &\overset{\triangle}{\iff} ctx^M(u) \subseteq ctx^M(u'),\\
u \leqslant^M_{\mathtt{C}\backslash\mathtt{W}} u' &\overset{\triangle}{\iff} ctx^M(u)_{|\mathtt{C}\times\Sigma^\omega} \subseteq ctx^M(u')_{|\mathtt{C}\times\Sigma^\omega},\\
u \leqslant^M_{\mathtt{R}\backslash\mathtt{W}} u' &\overset{\triangle}{\iff} ctx^M(u)_{|\Sigma^*\times\mathtt{R}^\omega} \subseteq ctx^M(u')_{|\Sigma^*\times\mathtt{R}^\omega},\\
u \leqslant^M_{\mathtt{U_c}\backslash\mathtt{R}} u' &\overset{\triangle}{\iff} ctx^M(u)_{|\mathtt{C}\times\mathtt{R}^\omega} \subseteq ctx^M(u')_{|\mathtt{C}\times\mathtt{R}^\omega} \ .
\end{aligned}$$

Similarly, we define the qo $\preccurlyeq^M \triangleq \bigcup_{\mathtt{J}\in\mathcal{P}} \preccurlyeq^M_\mathtt{J}$ on Σ^* where for every $\mathtt{J} \in \mathcal{P}$, $\preccurlyeq^M_\mathtt{J} \subseteq \mathtt{J} \times \mathtt{J}$ is the qo defined by

$$\begin{aligned}
u \preccurlyeq^M_\mathtt{W} u' &\overset{\triangle}{\iff} u \leqslant^M_\mathtt{W} u' \ \wedge \ ctx^M_\circledast(u) \subseteq ctx^M_\circledast(u'),\\
u \preccurlyeq^M_{\mathtt{C}\backslash\mathtt{W}} u' &\overset{\triangle}{\iff} u \leqslant^M_{\mathtt{C}\backslash\mathtt{W}} u' \ \wedge \ \Big(ctx^M_\circledast(u)_{|\mathtt{C}\times\mathtt{C}} \subseteq ctx^M_\circledast(u')_{|\mathtt{C}\times\mathtt{C}}\Big),\\
u \preccurlyeq^M_{\mathtt{R}\backslash\mathtt{W}} u' &\overset{\triangle}{\iff} u \leqslant^M_{\mathtt{R}\backslash\mathtt{W}} u' \ \wedge \ \Big(ctx^M_\circledast(u)_{|\Sigma^*\times\mathtt{R}} \subseteq ctx^M_\circledast(u')_{|\Sigma^*\times\mathtt{R}}\Big),\\
u \preccurlyeq^M_{\mathtt{U_c}\backslash\mathtt{R}} u' &\overset{\triangle}{\iff} u, u' \in \mathtt{U_c}\backslash\mathtt{R} \ .
\end{aligned}$$

Lemma 8. *Given a VPA $\mathcal{B}$, $J \in \mathcal{P}$ and two words $u, u' \in J$ we can decide whether $u \leqslant^{L^\omega(\mathcal{B})}_J u'$.*

Proof. Let $\mathcal{B} = (Q, q_I, \Gamma, \delta, F)$ be a VPA, $\mathtt{J} \in \mathcal{P}$ and $u \in \mathtt{J}$. We define the VPA $\mathcal{B}^\mathtt{J}_u = (Q \cup \overline{Q} \cup \{*\}, q_I, \delta \cup \overline{\delta} \cup \delta_J, \Gamma \cup \{\$\}, \overline{F})$ over the alphabet $\Sigma_\$ \triangleq \Sigma \cup \{\$, \$_c, \$_r\}$ where $\$, \$_c, \$_r \notin \Sigma \cup \Gamma$ are new symbols for internal, call and return positions resp., and where $\overline{Q} \triangleq \{\overline{q} \mid q \in Q\}$ is a copy of Q, $\overline{F} \triangleq \{\overline{q} \in \overline{Q} \mid q \in F\}$, $\overline{\delta}$ simulates δ on $\overline{Q}$ and δ_J is defined according to $\mathtt{J}$ as follows:

- for $\mathtt{J} = \mathtt{W}$, $\delta_\mathtt{W} \triangleq \{(p, \$, \overline{q}) \mid (p,q) \in ctx^\mathcal{B}(u)\}$,
- for $\mathtt{J} = \mathtt{C}\backslash\mathtt{W}$, $\delta_{\mathtt{C}\backslash\mathtt{W}} \triangleq \{(p, \$_r, \bot, \overline{q}) \mid (p,q) \in ctx^\mathcal{B}(u)\}$,
- for $\mathtt{J} = \mathtt{R}\backslash\mathtt{W}$, $\delta_{\mathtt{C}\backslash\mathtt{W}} \triangleq \{(p, \$_c, \overline{q}, \$) \mid (p,q) \in ctx^\mathcal{B}(u)\}$ and,
- for $\mathtt{J} = \mathtt{U_c}\backslash\mathtt{R}$, $\delta_{\mathtt{U_c}\backslash\mathtt{R}} \triangleq \{(p, \$_r, \bot, *), (*, \$_c, \overline{q}, \$) \mid (p,q) \in ctx^\mathcal{B}(u)\}$.

We have $L^\omega(\mathcal{B}_u^{\mathtt{W}}) = \{s\$\xi \in \Sigma_\$^\omega \mid su\xi \in M\}$, $L^\omega(\mathcal{B}_u^{\mathtt{C}\backslash\mathtt{W}}) = \{s\$_r\xi \in \Sigma_\$^\omega \mid s \in \mathtt{C}, su\xi \in M\}$, $L^\omega(\mathcal{B}_u^{\mathtt{R}\backslash\mathtt{W}}) = \{s\$_c\xi \in \Sigma_\$^\omega \mid \xi \in \mathtt{R}^\omega, su\xi \in M\}$ and $L^\omega(\mathcal{B}_u^{\mathtt{U_c}\backslash\mathtt{R}}) = \{s\$_r\xi \in \Sigma_\$^\omega \mid s \in \mathtt{C}, \xi \in \mathtt{R}^\omega, su\xi \in M\}$. Thus, for $u, u' \in \mathtt{J}$ we have $u \leqslant_{\mathtt{J}}^{L^\omega(\mathcal{B})} u' \iff L^\omega(\mathcal{B}_u^{\mathtt{J}}) \subseteq L^\omega(\mathcal{B}_{u'}^{\mathtt{J}})$. □

Lemma 9. *Given a VPA $\mathcal{B}$, $\mathtt{J} \in \mathcal{P}$ and two words $u, u' \in \mathtt{J}$ we can decide whether $u \preccurlyeq_J^{L^\omega(\mathcal{B})} u'$.*

Proof. Let $\mathcal{B} = (Q, q_I, \Gamma, \delta, F)$ be a VPA and $\Sigma_\$ \triangleq \Sigma \cup \{\$, \$_c, \$_r\}$ where $\$, \$_c, \$_r \notin \Sigma \cup \Gamma$ are new symbols for internal, call and return positions resp. For every $\mathtt{J} \in \{\mathtt{W}, \mathtt{C}\backslash\mathtt{W}, \mathtt{R}\backslash\mathtt{W}\}$ and $u \in \mathtt{J}$ we will define two VPA $\mathcal{C}_u^{1,\mathtt{J}}$ and $\mathcal{C}_u^{2,\mathtt{J}}$ over $\Sigma_\$$ to simulate the accepting runs of $\mathcal{B}$ of the form $(q_I, \bot) \vdash^{*s} (q_0, w_0) \vdash^{*u} (p_0, w_0) \vdash^{*t_1} (q_1, w_1) \vdash^{*u} (p_1, w_1) \cdots$ such that $\mathcal{C}_u^{1,\mathtt{J}}$ simulates the runs with $(q_n, w_n) \vdash^{\circledast u} (p_n, w_n)$ for infinitely many $n \in \mathbb{N}$ and $\mathcal{C}_u^{2,\mathtt{J}}$ simulates the runs with $(p_n, w_n) \vdash^{\circledast t_{n+1}} (q_{n+1}, w_{n+1})$ for infinitely many $n \in \mathbb{N}$. Define $M_u^{\mathtt{J}} = L^\omega(\mathcal{B}_u^{\mathtt{J}})$, where $\mathcal{B}_u^{\mathtt{J}}$ is the VPA defined in the proof of Lemma 8 and, $M_u^{\circledast\mathtt{J}} = L^\omega(\mathcal{C}_u^{1,\mathtt{J}}) \cup L^\omega(\mathcal{C}_u^{2,\mathtt{J}})$. For every word w we will define $\mathcal{C}_w^{1,\mathtt{J}}$ and $\mathcal{C}_w^{2,\mathtt{J}}$ such that $u \preccurlyeq_{\mathtt{J}}^M u' \iff M_u^{\mathtt{J}} \subseteq M_{u'}^{\mathtt{J}} \wedge M_u^{\circledast\mathtt{J}} \subseteq M_{u'}^{\circledast\mathtt{J}}$.

We define $\mathcal{C}_u^{1,\mathtt{J}} = (Q \cup_{q\in Q} \{p_q \mid p \in Q\}, q_I, \Gamma, \delta^{1,\mathtt{J}}, \{p_q \mid (q,p) \in ctx_\circledast^{\mathcal{B}}(u)\})$ where $\delta^{1,\mathtt{J}} \triangleq \delta \cup \delta_{\mathtt{J}}^{\$,1} \cup_{q\in Q} \delta^q$ with $\delta^q \triangleq \{(p_q, a, p') \mid (p, a, p') \in \delta_i\} \cup \{(p_q, a, p', \gamma) \mid (p, a, p', \gamma) \in \delta_c\} \cup \{(p_q, a, \gamma, p') \mid (p, a, \gamma, p') \in \delta_r\}$ and $\delta_{\mathtt{J}}^{\$,1}$ is defined according to $\mathtt{J}$ by $\delta_{\mathtt{W}}^{\$,1} \triangleq \{(q, \$, p_q) \mid (q,p) \in ctx^{\mathcal{B}}(u)\}$, $\delta_{\mathtt{C}\backslash\mathtt{W}}^{\$,1} \triangleq \{(q, \$_r, \bot, p_q) \mid (q,p) \in ctx^{\mathcal{B}}(u)\}$ and $\delta_{\mathtt{R}\backslash\mathtt{W}}^{\$,1} \triangleq \{(q, \$_c, p_q, \$) \mid (q,p) \in ctx^{\mathcal{B}}(u)\}$.

We define $\mathcal{C}_u^{2,\mathtt{J}} = (Q \cup \overline{Q} \cup F_u^2, q_I, \Gamma, \delta^{2,\mathtt{J}}, F_u^2)$ over $\Sigma_\$$ where $\overline{Q}$ is a copy of Q, $F_u^2 \triangleq \{p_\$ \mid \exists q \in Q, (q,p) \in ctx^{\mathcal{B}}(u)\}$, and $\delta^{2,\mathtt{J}} \triangleq \delta \cup \overline{\delta} \cup \delta_{\mathtt{J}}^{\$,2} \cup \delta^2$, $\overline{\delta}$ simulates δ on $\overline{Q}$, $\delta^2 \triangleq \{(p_\$, a, \overline{q}) \mid (p, a, q) \in \delta_i\} \cup \{(p_\$, a, \overline{q}, \gamma) \mid (p, a, q, \gamma) \in \delta_c\} \cup \{(p_\$, a, \gamma, \overline{q}) \mid (p, a, \gamma, q) \in \delta_r\} \cup \bigcup_{p\in F}\{(\overline{p}, a, q) \mid (p, a, q) \in \delta_i\} \cup \{(\overline{p}, a, q, \gamma) \mid (p, a, q, \gamma) \in \delta_c\} \cup \{(\overline{p}, a, \gamma, q) \mid (p, a, \gamma, q) \in \delta_r\}$ and $\delta_{\mathtt{J}}^{\$,2}$ is defined according to $\mathtt{J}$ by $\delta_{\mathtt{W}}^{\$,2} \triangleq \{(q, \$, p_\$) \mid (q,p) \in ctx^{\mathcal{B}}(u)\}$, $\delta_{\mathtt{C}\backslash\mathtt{W}}^{\$,2} \triangleq \{(q, \$_r, \bot, p_\$) \mid (q,p) \in ctx^{\mathcal{B}}(u)\}$ and $\delta_{\mathtt{R}\backslash\mathtt{W}}^{\$,2} \triangleq \{(q, \$_c, p_\$, \$) \mid (q,p) \in ctx^{\mathcal{B}}(u)\}$.

Using Lemma 8 and the definitions of the VPA $\mathcal{C}_u^{1,\mathtt{J}}$ and $\mathcal{C}_u^{2,\mathtt{J}}$ it is an easy exercise to show that $u \preccurlyeq_{\mathtt{J}}^M u' \Leftarrow M_u^{\mathtt{J}} \subseteq M_{u'}^{\mathtt{J}} \wedge M_u^{\circledast\mathtt{J}} \subseteq M_{u'}^{\circledast\mathtt{J}}$. Since by Lemma 8 $u \leqslant_{\mathtt{J}}^{L^\omega(\mathcal{B})} u' \iff M_u^{\mathtt{J}} \subseteq M_{u'}^{\mathtt{J}}$ it remains to show that $u \preccurlyeq_{\mathtt{J}}^M u' \implies M_u^{\circledast\mathtt{J}} \subseteq M_{u'}^{\circledast\mathtt{J}}$. Next we show this for $\mathtt{J} = \mathtt{W}$ (the other cases are similar). Assume $u \preccurlyeq_{\mathtt{W}}^M u'$. By Theorem 11 it suffices to show that all ultimately periodic words of $M_u^{\circledast\mathtt{W}}$ are in $M_{u'}^{\circledast\mathtt{W}}$. Let $s\$s_1\$s_2 \cdots \$s_k(\$t_1\$t_2 \cdots \$t_n)^\omega \in M_u^\circledast$. For $\overline{s} \triangleq sus_1us_2 \cdots us_k$ and $\overline{t} \triangleq t_1ut_2 \cdots ut_n$ we have $(\overline{s}, \overline{t}) \in ctx_\circledast^M(u)$. Since $ctx_\circledast^M(u) \subseteq ctx_\circledast^M(u')$ we deduce that $\overline{s}(u'\overline{t})^\omega = \overline{s}u't_1(ut_2 \cdots ut_nu't_1)^\omega \in M$. By iterating the previous argument we deduce that $\overline{s}(u't_1u't_2 \cdots u't_n)^\omega \in M$. For $\xi \triangleq s_1us_2 \cdots us_k(u't_1u't_2 \cdots u't_n)^\omega$ we have $(s, \xi) \in ctx^M(u)$. Since $ctx^M(u) \subseteq ctx^M(u')$ we deduce that $su'\xi = (su's_1)us_2 \cdots us_k(u't_1u't_2 \cdots u't_n)^\omega \in M$. By iterating the previous argument we finally deduce that $su's_1u's_2 \cdots u's_k(u't_1u't_2 \cdots u't_n)^\omega \in M$. Hence, $s\$s_1\$s_2 \cdots \$s_k(\$t_1\$t_2 \cdots \$t_n)^\omega \in M_{u'}^{\circledast\mathtt{W}}$.

□

Proposition 20. *Let $\mathcal{B}$ be a VPA. The pair $\leqslant^{L^\omega(\mathcal{B})}, \preccurlyeq^{L^\omega(\mathcal{B})}$ is $L^\omega(\mathcal{B})$-suitable.*

Proof. First we show that the pair $\leqslant^M, \preccurlyeq^M$ is M-preserving, where $M \triangleq L^\omega(\mathcal{B})$. Let $(u,v), (u',v') \in \mathtt{C} \times \mathtt{C}$ (resp. $\mathtt{U_c} \times \mathtt{R}$) such that $u \leqslant^M u'$, $v \preccurlyeq^M v'$ and $uv^\omega \in M$. From $u \leqslant^M u'$ and $uv^\omega \in M$ we deduce that $(\epsilon, v^\omega) \in ctx^M(u)_{|\mathtt{C}\times\Sigma^\omega} \subseteq ctx^M(u')_{|\mathtt{C}\times\Sigma^\omega}$ (resp. $(\epsilon, v^\omega) \in$

$ctx^M(u)_{|\mathtt{C}\times\mathtt{R}^\omega} \subseteq ctx^M(u')_{|\mathtt{C}\times\mathtt{R}^\omega})$. Thus, $u'v^\omega \in M$. From $v \preccurlyeq^M v'$ and $u'v^\omega \in M$ we deduce that $(u', \epsilon) \in ctx^M_\circledast(v)_{|\mathtt{C}\times\mathtt{C}} \subseteq ctx^M_\circledast(v')_{|\mathtt{C}\times\mathtt{C}}$ (resp. $(u', \epsilon) \in ctx^M_\circledast(v)_{|\Sigma^*\times\mathtt{R}} \subseteq ctx^M_\circledast(v')_{|\Sigma^*\times\mathtt{R}}$). Thus, $u'v'^\omega \in M$.

We now show that the qo $\leqslant^M$ satisfies the monotonicity conditions **C** and **R**. Let $u \leqslant^M u'$ such that $u, u' \in \mathtt{C}$ (resp. $u, u' \in \mathtt{R}$). Let $s \in \mathtt{C}$ and $t \in \Sigma^*$ (resp. $s \in \Sigma^*$ and $t \in \mathtt{R}$). If $u, u' \in \mathtt{W}$ then it is easy to check that $sut \leqslant^M su't$. Otherwise $u, u' \in \mathtt{C}\backslash\mathtt{W}$ (resp. $u, u' \in \mathtt{R}\backslash\mathtt{W}$) and we distinguish two cases: if $t \in \mathtt{C}$ (resp. $s \in \mathtt{R}$) then $sut, su't \in \mathtt{C}\backslash\mathtt{W}$ (resp. $sut, su't \in \mathtt{R}\backslash\mathtt{W}$). We show that $sut \leqslant^M_{\mathtt{C}\backslash\mathtt{W}} su't$ (resp. $sut \leqslant^M_{\mathtt{R}\backslash\mathtt{W}} su't$). Let $(s', \xi) \in ctx^M(sut)_{|\mathtt{C}\times\Sigma^\omega}$ (resp. $(s', \xi) \in ctx^M(sut)_{|\Sigma^*\times\mathtt{R}^\omega}$). Since $s's \in \mathtt{C}$ (resp. $t\xi \in \mathtt{R}^\omega$), we deduce from $u \leqslant^M_{\mathtt{C}\backslash\mathtt{W}} u'$ (resp. $u \leqslant^M_{\mathtt{R}\backslash\mathtt{W}} u'$) that $(s', \xi) \in ctx^M(su't)_{|\mathtt{C}\times\Sigma^\omega}$ (resp. $(s', \xi) \in ctx^M(su't)_{|\Sigma^*\times\mathtt{R}^\omega}$). If $t \in \mathtt{U_c}$ (resp. $s \in \Sigma^*\backslash\mathtt{R}$) then $sut, su't \in \mathtt{U_c}\backslash\mathtt{R}$ and similarly we can show that $sut \leqslant^M_{\mathtt{U_c}\backslash\mathtt{R}} su't$.

Next we show that the qo $\preccurlyeq^M$ satisfies the monotonicity conditions $\mathbf{C}_\circledast$ and $\mathbf{R}_\circledast$. Let $u \preccurlyeq^M u'$ such that $u, u' \in \mathtt{C}$ (resp. $u, u' \in \mathtt{R}$). Let $s, t \in \mathtt{C}$ (resp. $s, t \in \mathtt{R}$). If $u, u' \in \mathtt{C}\backslash\mathtt{W}$ (resp. $u, u' \in \mathtt{R}\backslash\mathtt{W}$) then $sut, su't \in \mathtt{C}\backslash\mathtt{W}$ (resp. $sut, su't \in \mathtt{R}\backslash\mathtt{W}$). We show that $sut \preccurlyeq^M_{\mathtt{C}\backslash\mathtt{W}} su't$ (resp. $sut \preccurlyeq^M_{\mathtt{R}\backslash\mathtt{W}} su't$). Previously we established that $sut \leqslant^M_{\mathtt{C}\backslash\mathtt{W}} su't$ (resp. $sut \leqslant^M_{\mathtt{R}\backslash\mathtt{W}} su't$). Let $(s', t') \in ctx^M_\circledast(sut)_{|\mathtt{C}\times\mathtt{C}}$ (resp. $(s', t') \in ctx^M_\circledast(sut)_{|\Sigma^*\times\mathtt{R}}$). Since $s'(sutt')^\omega = s's(utt's) \in M$ we deduce that $(s's, tt's) \in ctx^M_\circledast(u)_{|\mathtt{C}\times\mathtt{C}} \subseteq ctx^M_\circledast(u')_{|\mathtt{C}\times\mathtt{C}}$ (resp. $(s's, tt's) \in ctx^M_\circledast(u)_{|\Sigma^*\times\mathtt{R}} \subseteq ctx^M_\circledast(u')_{|\Sigma^*\times\mathtt{R}}$). Thus, $(s', t') \in ctx^M_\circledast(su't)_{|\mathtt{C}\times\mathtt{C}}$ (resp. $(s', t') \in ctx^M_\circledast(su't)_{|\Sigma^*\times\mathtt{R}}$). If $u, u' \in \mathtt{W}$ then following similar arguments as previously we can show that $sut \preccurlyeq^M su't$. The monotonicity conditions **W** and **U** for $\leqslant^M$ and the monotonicity condition **W** for $\preccurlyeq^M$ are left as an exercise to the reader.

The quasiorders are decidable by Lemmas 8 and 9. Finally, the proof that $\leqslant^M$ and $\preccurlyeq^M$ are well-quasiorders follows from [23, Prop 1.2] by observing that for every $\mathtt{J}$ in the partition $\mathcal{P}$ of Σ^* we have $\leqslant^{\mathcal{B}}_{|\mathtt{J}\times\mathtt{J}} \subseteq \leqslant^M_{\mathtt{J}}$ and $\preccurlyeq^{\mathcal{B}}_{|\mathtt{J}\times\mathtt{J}} \subseteq \preccurlyeq^M_{\mathtt{J}}$, where $\leqslant^{\mathcal{B}}$ and $\preccurlyeq^{\mathcal{B}}$ are the state-based quasiorders defined in Section 7.5. Next we show that $\leqslant^{\mathcal{B}}_{|\mathtt{J}\times\mathtt{J}} \subseteq \leqslant^M_{\mathtt{J}}$ and $\preccurlyeq^{\mathcal{B}}_{|\mathtt{J}\times\mathtt{J}} \subseteq \preccurlyeq^M_{\mathtt{J}}$.

- Let $\mathtt{J} \in \mathcal{P}$ and $u \leqslant^{\mathcal{B}}_{|\mathtt{J}\times\mathtt{J}} u'$. Let $(s, \xi) \in ctx^M(u)$ and

$$e\colon (q_I, \bot) \vdash^{*s} (q, w) \vdash^{*u} (p, w_0) \vdash^{\xi_1} (p_1, w_1) \vdash^{\xi_2} (p_2, w_2) \cdots$$

  be an accepting run of $\mathcal{B}$ for $su\xi$ where $\xi = \xi_1\xi_2\cdots$. If $\mathtt{J} = \mathtt{C}\backslash\mathtt{W}$ then we restrict $s \in \mathtt{C}$. Therefore, $w = w_0 = \bot$. An accepting run of $\mathcal{B}$ for $su'\xi$ is obtained from e by replacing the sequence of transitions $(q, \bot) \vdash^{*u} (p, \bot)$ by a sequence $(q, \bot) \vdash^{*u'} (p, \bot)$ which exists since $(q, p) \in ctx^{\mathcal{B}}(u) \subseteq ctx^{\mathcal{B}}(u')$. If $\mathtt{J} = \mathtt{R}\backslash\mathtt{W}$ then we restrict $\xi \in \mathtt{R}^\omega$. Since $u \in \mathtt{R}$ no symbol from the stack w is popped while reading u in e. Therefore, $(q, p) \in ctx^{\mathcal{B}}(u)$. Thus, we can derive an accepting run of $\mathcal{B}$ for $su'\xi$ from e by replacing the sequence of transitions $(q, w) \vdash^{*u} (p, w_0)$ by a sequence $(q, w) \vdash^{*u'} (p, wz)$ (which exists since $ctx^{\mathcal{B}}(u) \subseteq ctx^{\mathcal{B}}(u')$ and $u' \in \mathtt{R}$) and by observing that, since $\xi \in \mathtt{R}^\omega$, the same infinite sequence of transitions $(p, w_0) \vdash^{\xi_1} (p_1, w_1) \vdash^{\xi_2} (p_2, w_2) \cdots$ can also be achieved starting from (p, wz). The cases $\mathtt{J} \in \{\mathtt{W}, \mathtt{U_c}\backslash\mathtt{R}\}$ are handled similarly. In every case $\mathtt{J}$, we have $u \leqslant^M_{\mathtt{J}} u'$.
- Let $\mathtt{J} \in \mathcal{P}$ and $u \preccurlyeq^{\mathcal{B}}_{|\mathtt{J}\times\mathtt{J}} u'$. Since $u \leqslant^{\mathcal{B}}_{|\mathtt{J}\times\mathtt{J}} u'$ and by the previous proof $\leqslant^{\mathcal{B}}_{|\mathtt{J}\times\mathtt{J}} \subseteq \leqslant^M_{\mathtt{J}}$ we

have $u \leqslant^M_{\mathtt{J}} u'$. Let $(s,t) \in ctx^M_{\circledast}(u)$ and

$$d : \ (q_I, \bot) \vdash^{*s} (q_0, z_0) \vdash^{*u} (p_0, w_0) \vdash^{*t} (q_1, z_1) \vdash^{*u} (p_1, w_1) \vdash^{*t} (q_2, z_2) \cdots$$

be an accepting run of $\mathcal{B}$ for $s(ut)^\omega$. If $\mathtt{J} = \mathtt{C}\backslash\mathtt{W}$ then we restrict $s, t \in \mathtt{C}$. Thus, $w_n = z_n = \bot$ for all $n \in \mathbb{N}$. An accepting run of $\mathcal{B}$ for $s(u't)^\omega$ is obtained from d by replacing every sequence of transitions $(q_n, \bot) \vdash^{*u} (p_n, \bot)$ by a sequence $(q_n, \bot) \vdash^{*u'} (p_n, \bot)$ which exists since $ctx^{\mathcal{B}}(u) \subseteq ctx^{\mathcal{B}}(u')$ and $ctx^{\mathcal{B}}_{\circledast}(u) \subseteq ctx^{\mathcal{B}}_{\circledast}(u')$. The cases $\mathtt{J} \in \{\mathtt{W}, \mathtt{R}\backslash\mathtt{W}\}$ are analogues and use similar arguments as those in the proofs of $\leqslant^{\mathcal{B}}_{|\mathtt{J}\times\mathtt{J}} \subseteq \leqslant^M_{\mathtt{J}}$. The case $\mathtt{J} = \mathtt{U_c}\backslash\mathtt{R}$ is trivial.

□

Deciding the quasiorders $\leqslant^M$ and $\preccurlyeq^M$ when M is ω-VPL, is as hard as the inclusion problem between ω-VPL generated by VPA, as shown by the proofs of Lemmas 8 and 9. Nevertheless, these quasiorders act as a gold standard for quasiorders in the sense formalized in the next proposition.

Proposition 21. *Let $M \subseteq \Sigma^\omega$ be an ω-VPL and $\leqslant, \preccurlyeq$ be a M-suitable pair of quasiorders such that $\preccurlyeq \subseteq \leqslant$. For every $\mathtt{J} \in \mathcal{P}$ we have $\leqslant_{|J\times J} \subseteq \leqslant^M$ and $\preccurlyeq_{|J\times J} \subseteq \preccurlyeq^M$.*

Proof. First we show that for every $\mathtt{J} \in \mathcal{P}$ we have $\leqslant_{|\mathtt{J}\times\mathtt{J}} \subseteq \leqslant^M$. For that we observe that when M is ω-VPL we have $\leqslant^M = \leqslant^M_{UP}$ where $\leqslant^M_{UP}$ is the quasiorder on Σ^* defined the analogue way to $\leqslant^M$ but using $ctx^M_{UP}(u) \triangleq \{(x, st^\omega) \in \Sigma^* \times \Sigma^\omega \mid xust^\omega \in M\}$ instead of $ctx^M(u)$. Hence, it suffices to show $\leqslant_{|\mathtt{J}\times\mathtt{J}} \subseteq \leqslant^M_{UP}$. Let $\mathtt{J} \in \mathcal{P}$ and $u, u' \in \mathtt{J}$ such that $u \leqslant u'$.

- If $\mathtt{J} = \mathtt{W}$ let $s \in \Sigma^*$ and $wv^\omega \in \Sigma^\omega$ such that $suwv^\omega \in M$. By Proposition 15 there is a decomposition $suwv^\omega = s_0 t_0^\omega$ with $(s_0, t_0) \in \mathtt{Ld}$. We can assume that su is a prefix of s_0, i.e. $s_0 = sus'$ for some $s' \in \Sigma^*$ (otherwise replace s by $s_0 t_0^n$ for $n \in \mathbb{N}$ large enough). Hence, we have $suwv^\omega = sus't_0^\omega$ and $(sus', t_0) \in \mathtt{Ld}$. It is an easy exercise to deduce from $u, u' \in \mathtt{W} = \mathtt{C} \cap \mathtt{R}$ and the monotonicity conditions $\mathbf{W}, \mathbf{C}, \mathbf{R}$ that $sus' \leqslant su's'$. Since $u, u' \in \mathtt{W}$ we have $sus' \in \mathtt{J}' \iff su's' \in \mathtt{J}'$ for $\mathtt{J}' \in \mathcal{P}$, and since $sus' \leqslant su's'$, $t_0 \preccurlyeq t_0$, the pair $\leqslant, \preccurlyeq$ is M-preserving and $sus't_0^\omega \in M$ we deduce that $su's't_0^\omega = su'wv^\omega \in M$.
- If $\mathtt{J} = \mathtt{C}\backslash\mathtt{W}$ let $s \in \mathtt{C}$ and $s_0 t_0^\omega \in \Sigma^\omega$ such that $sus_0 t_0^\omega \in M$. By Proposition 15 we can assume $s_0 t_0^\omega \in \mathtt{Ld}$. By the monotonicity condition $\mathbf{C}$ we have $sus_0 \leqslant su's_0$. Since $sus_0 \in \mathtt{C} \iff su's_0 \in \mathtt{C}$, $t_0 \preccurlyeq t_0$ and $sus_0 t_0^\omega \in M$ and the pair $\leqslant, \preccurlyeq$ is M-preserving we deduce that $su's_0 t_0^\omega \in M$.
- If $\mathtt{J} = \mathtt{R}\backslash\mathtt{W}$ (resp. $\mathtt{J} = \mathtt{U_c}\backslash\mathtt{R}$) let $s \in \Sigma^*$ (resp. $s \in \mathtt{C}$) and $wv^\omega \in \mathtt{R}^\omega$ such that $suwv^\omega \in M$. We can assume $w, v \in \mathtt{R}$. By the monotonicity condition $\mathbf{R}$ (resp. $\mathbf{U}$) we have $suw \leqslant su'w$. Since $su, su' \in \mathtt{U_c}$, $v \in \mathtt{R}$, $v \preccurlyeq v$ and the pair $\leqslant, \preccurlyeq$ is M-preserving we have that $su'wv^\omega \in M$.

Second we show that for every $\mathtt{J} \in \mathcal{P}$ we have $\preccurlyeq_{|\mathtt{J}\times\mathtt{J}} \subseteq \preccurlyeq^M$. Let $\mathtt{J} \in \mathcal{P}$ and $u, u' \in \mathtt{J}$ such that $u \preccurlyeq u'$. Since $\preccurlyeq \subseteq \leqslant \subseteq \leqslant^M$ we have that $u \leqslant^M_{\mathtt{J}} u'$.

- If $\mathtt{J} = \mathtt{W}$ let $s, t \in \Sigma^*$ such that $s(ut)^\omega \in M$. Since $u \preccurlyeq^M_{\mathtt{W}} u'$ by the monotonicity condition $\mathbf{C}$ we have $ut \preccurlyeq^M u't$. Hence, since $\leqslant, \preccurlyeq$ is M-preserving if $(s,t) \in \mathtt{Ld}$ then

$s(u't)^\omega \in M$. Assume $s \in \mathtt{U_c}$ and $t \notin \mathtt{R}$. If $t \in \mathtt{C}$ then there is $n \in \mathbb{N}$ such that $s(ut)^n \in \mathtt{C}$. Hence, by preservation we find that $\overline{s}(u't)^\omega \in M$ where $\overline{s} \triangleq s(ut)^n$. We then use that $u \leqslant u'$ (since $\preccurlyeq \subseteq \leqslant$) to deduce that $s(u't)^\omega \in M$. If $t \in \mathtt{U_c}\backslash\mathtt{R}$ then $t = t_1t_2$ with $t_1 \in \mathtt{C}\backslash\mathtt{R}$ and $t_2 \in \mathtt{R}\backslash\mathtt{C}$ and we have $s(ut)^\omega = sut_1(t_2ut_1)^\omega$. Since $u \preccurlyeq_{\mathtt{W}}^M u'$ by the monotonicity conditions $\mathbf{W}, \mathbf{C}_\circledast, \mathbf{R}_\circledast$ we deduce that $t_2ut_1 \preccurlyeq^M t_2u't_1$. We then reason by splitting cases and by using similar arguments as previously. The case $s \in \mathtt{C}$ and $t \in \mathtt{U_c}$ is reduced to the case $s \in \mathtt{U_c}$ by replacing s with sut and by using similar arguments.

- If $\mathtt{J} = \mathtt{C}\backslash\mathtt{W}$ let $s, t \in \mathtt{C}$ such that $s(ut)^\omega \in M$. We have $ut \preccurlyeq u't$ (monotonicity condition $\mathbf{C}_\circledast$), $s \leqslant s$, $s(ut)^\omega \in M$ and $s, ut, u't \in \mathtt{C}$ thus by preservation $s(u't)^\omega \in M$.
- If $\mathtt{J} = \mathtt{R}\backslash\mathtt{W}$ let $s, t \in \mathtt{R}$ such that $s(ut)^\omega \in M$. By the monotonicity condition $\mathbf{R}_\circledast$ we have $sut \preccurlyeq su't$ and $ut \preccurlyeq u't$, thus also $sut \leqslant su't$. Since $sut, su't \in \mathtt{U_c}$ and $ut, u't \in \mathtt{R}$ we deduce by preservation that $s(u't)^\omega \in M$.
- If $\mathtt{J} = \mathtt{U_c}\backslash\mathtt{R}$ then $u, u' \in \mathtt{U_c}\backslash\mathtt{R}$, thus $u \preccurlyeq^M u'$.

□

By Propositions 20 and 21 the pair $\leqslant^{L^\omega(\mathcal{B})}, \preccurlyeq^{L^\omega(\mathcal{B})}$ is the greatest (w.r.t $\subseteq \times \subseteq$) among the $L^\omega(\mathcal{B})$-suitable pairs $\leqslant, \preccurlyeq$ of quasiorders that respect the partition $\mathcal{P}$ and that verify $\preccurlyeq \subseteq \leqslant$.

7.6 Algorithm

We are now in position to present our algorithm which, given two VPA $\mathcal{A} = (Q, q_I, \Gamma, \delta, F)$ and $\mathcal{B} = (\hat{Q}, \hat{q}_I, \hat{\Gamma}, \hat{\delta}, \hat{F})$ and a pair of $L^\omega(\mathcal{B})$-suitable quasiorders, decides the inclusion problem $L^\omega(\mathcal{A}) \subseteq L^\omega(\mathcal{B})$.

Algorithm 6 computes a finite basis for S w.r.t. $\leqslant \times \preccurlyeq$ (lines 1–2) and afterwards checks membership in $L^\omega(\mathcal{B})$ on every ultimately periodic word uv^ω stemming from this finite basis (lines 3–7).

Algorithm 6: Algorithm for deciding $L^\omega(\mathcal{A}) \subseteq L^\omega(\mathcal{B})$

Data: VPA $\mathcal{A} = (Q, q_I, \Gamma, \delta, F)$ and $\mathcal{B} = (\hat{Q}, \hat{q}_I, \hat{\Gamma}, \hat{\delta}, \hat{F})$.
Data: $L^\omega(\mathcal{B})$-suitable pair $\leqslant, \preccurlyeq$.
Data: Procedure deciding $uv^\omega \in L^\omega(\mathcal{B})$ given (u, v).

1 Compute ${f_\mathcal{A}}^m(\vec{\emptyset})$ with least m s.t. ${f_\mathcal{A}}^{m+1}(\vec{\emptyset}) \sqsubseteq_\leqslant^{4\cdot|Q|^2} {f_\mathcal{A}}^m(\vec{\emptyset})$;
2 Compute ${r_\mathcal{A}}^{m'}(\vec{\emptyset})$ with least m' s.t. ${r_\mathcal{A}}^{m'+1}(\vec{\emptyset}) \sqsubseteq_\preccurlyeq^{6\cdot|Q|^2} {r_\mathcal{A}}^{m'}(\vec{\emptyset})$;
3 **foreach** $p \in Q$ **do**
4 **foreach** $u \in ({f_\mathcal{A}}^m(\vec{\emptyset}))_{2,q_I,p}$, $v \in ({r_\mathcal{A}}^{m'}(\vec{\emptyset}))_{5,p,p}$ **do**
5 **if** $uv^\omega \notin L^\omega(\mathcal{B})$ **then return** false;
6 **foreach** $u \in ({f_\mathcal{A}}^m(\vec{\emptyset}))_{4,q_I,p}$, $v \in ({r_\mathcal{A}}^{m'}(\vec{\emptyset}))_{6,p,p}$ **do**
7 **if** $uv^\omega \notin L^\omega(\mathcal{B})$ **then return** false;
8 **return** true;

Theorem 13. *Given the required inputs, Algorithm 6 decides the inclusion problem* $L^\omega(\mathcal{A}) \subseteq L^\omega(\mathcal{B})$.

Proof. As established by Proposition 18, given a monotonic pair $\leqslant, \preccurlyeq$ of decidable well-quasiorders, Algorithm 6 computes in line 1 (resp. line 2) a finite basis ${f_\mathcal{A}}^m(\vec{\emptyset})$ (resp. ${r_\mathcal{A}}^{m'}(\vec{\emptyset})$) for $\operatorname{lfp} f_\mathcal{A}$ (resp. $\operatorname{lfp} r_\mathcal{A}$) w.r.t. $\leqslant$ (resp. $\preccurlyeq$). Next define:

$$S_\mathcal{A}^{m,m'} \triangleq \bigcup_{p\in Q}\Big(\big(({f_\mathcal{A}}^m(\vec{\emptyset}))_{2,q_I,p} \times ({r_\mathcal{A}}^{m'}(\vec{\emptyset}))_{5,p,p}\big) \cup \big(({f_\mathcal{A}}^m(\vec{\emptyset}))_{4,q_I,p} \times ({r_\mathcal{A}}^{m'}(\vec{\emptyset}))_{6,p,p}\big)\Big) \ .$$

Using Equation (7.1) we deduce that $S_\mathcal{A}^{m,m'}$ is a finite basis for S w.r.t. $\leqslant \times \preccurlyeq$. Since the pair $\leqslant, \preccurlyeq$ is $L^\omega(\mathcal{B})$-preserving, by Section 7.1, we deduce that

$$L^\omega(\mathcal{A}) \subseteq L^\omega(\mathcal{B}) \iff \forall (u,v) \in S_\mathcal{A}^{m,m'},\ uv^\omega \in L^\omega(\mathcal{B}) \ .$$

□

We remark that Algorithm 6 can be easily adapted to decide the inclusion problem between visibly pushdown languages of finite words. The adaptation to the finite words case omits the fixpoint computation of line 2 and iterates over the components (i, q_I, p) where $i \in \{2,3,4\}$ and where $p \in F$ is a final state.

Example 22. *Consider the iterates of the function $f_\mathcal{A}$ from Example 20. One can check that ${f_\mathcal{A}}^4(\vec{\emptyset}) \sqsubseteq^4_{\preccurlyeq^\mathcal{B}} {f_\mathcal{A}}^3(\vec{\emptyset})$ (thus also ${f_\mathcal{A}}^4(\vec{\emptyset}) \sqsubseteq^4_{\leqslant^\mathcal{B}} {f_\mathcal{A}}^3(\vec{\emptyset})$ since $\preccurlyeq^\mathcal{B} \subseteq \leqslant^\mathcal{B}$). Thus, we check whether the inclusion $L^\omega(\mathcal{A}) \subseteq L^\omega(\mathcal{B})$ holds on the finite set $(\{\epsilon, cr\} \times \{cr\}) \cup (\{c, c^2, c^3\} \times \{cr, c, c^2, c^3, c^4\})$ and find the counterexample $c(cr)^\omega \in L^\omega(\mathcal{A}) \backslash L^\omega(\mathcal{B})$.*

7.6.1 Antichains Everywhere

We show next that Algorithm 6 remains correct if, in the sequence of Kleene iterates of $f_\mathcal{A}$ or $r_\mathcal{A}$, for each application of $f_\mathcal{A}$ or $r_\mathcal{A}$ we first select a finite basis for their arguments instead (using $\leqslant^{4\cdot|Q|^2}$ for $f_\mathcal{A}$ and $\preccurlyeq^{6\cdot|Q|^2}$ for $r_\mathcal{A}$).

Proposition 22. *Let $\ltimes$ be a qo that verifies the monotonicity conditions $\mathbf{W}, \mathbf{C}, \mathbf{R}, \mathbf{U}$. If B is a basis for $(X, Y, Z, T) \in \wp(\mathtt{W})^{|Q|^2} \times \wp(\mathtt{C})^{|Q|^2} \times \wp(\mathtt{R})^{|Q|^2} \times \wp(\mathtt{U_c})^{|Q|^2}$ w.r.t. $\ltimes^{4\cdot|Q|^2}$, then $f_\mathcal{A}(B)$ is a basis for $f_\mathcal{A}(X, Y, Z, T)$ w.r.t. $\ltimes^{4\cdot|Q|^2}$. The analogue result holds for $r_\mathcal{A}$ when $\ltimes$ satisfies the monotonicity conditions $\mathbf{W}, \mathbf{C}_\circledast, \mathbf{R}_\circledast$.*

Proof. Let $\ltimes$ be a qo that verifies the monotonicity conditions $\mathbf{W}, \mathbf{C}, \mathbf{R}, \mathbf{U}$. Let $(Y, Z, T) \in \wp(\mathtt{C})^{|Q|^2} \times \wp(\mathtt{R})^{|Q|^2} \times \wp(\mathtt{U_c})^{|Q|^2}$ such that $(\overline{Y}, \overline{Z}, \overline{T})$ is a basis for (Y, Z, T). Let $u \in U(Y, Z, T)_{p,q}$ such that $u = ytz \in Y_{p,p'} T_{p',q'} Z_{q',q}$, where $y \in Y_{p,p'}$, $z \in Z_{q',q}$ and $t \in T_{p',q'}$ for some $p', q' \in Q$. There are $\overline{y} \in \overline{Y}_{p,p'}$, $\overline{t} \in \overline{T}_{p',q'}$ and $\overline{z} \in \overline{Z}_{q',q}$ such that $\overline{y} \ltimes y$, $\overline{t} \ltimes t$ and $\overline{z} \ltimes z$. Since $\overline{y} \ltimes y$ by the monotonicity condition $\mathbf{C}$ we have $\overline{y}tz \ltimes ytz$. Since $\overline{t} \ltimes t$ by the monotonicity condition $\mathbf{U}$ we have $\overline{y}\overline{t}z \ltimes \overline{y}tz$. Since $\overline{z} \ltimes z$ by the monotonicity condition $\mathbf{R}$ we have that $\overline{y}\overline{t}\overline{z} \ltimes \overline{y}\overline{t}z$. By transitivity of $\ltimes$ we thus obtain $\overline{y}\overline{t}\overline{z} \ltimes ytz$. Therefore, $U(Y, Z, T) \sqsubseteq^{3\cdot|Q|^2}_\ltimes U(\overline{Y}, \overline{Z}, \overline{T})$. Intuitively, if a vector (Y, Z, T) is subsumed by a vector $(\overline{Y}, \overline{Z}, \overline{T})$ then its image by U is

subsumed by the image of $(\overline{Y}, \overline{Z}, \overline{T})$ by U. Similar results hold for the other components of $f_{\mathcal{A}}$ paired with the previous monotonicity conditions.

□

Since every Kleene iterate of $f_{\mathcal{A}}$ belongs to $\wp(\mathtt{W})^{|Q|^2} \times \wp(\mathtt{C})^{|Q|^2} \times \wp(\mathtt{R})^{|Q|^2} \times \wp(\mathtt{U_c})^{|Q|^2}$ given a basis B for ${f_{\mathcal{A}}}^n(\vec{\emptyset})$ w.r.t. $\leqslant^{4\cdot|Q|^2}$, by Proposition 22, $f_{\mathcal{A}}(B)$ is a basis for ${f_{\mathcal{A}}}^{n+1}(\vec{\emptyset})$ w.r.t. $\leqslant^{4\cdot|Q|^2}$. Hence, at each iteration we can select, for each (i,p,q)-component, a basis w.r.t. $\leqslant$ and then apply $f_{\mathcal{A}}$. In particular, we can keep antichains for each (i,p,q)-component, that is, finite bases of incomparable words. The analogue result holds for the Kleene iterates of $r_{\mathcal{A}}$.

7.7 State-based Algorithm

Next we consider Algorithm 6 instantiated with the pair of state-based quasiorders of Section 7.5.1.

7.7.1 Fixpoint Computation

Given an input vector the functions $f_{\mathcal{A}}$ and $r_{\mathcal{A}}$ add new words of type uu', and cur to its components, where c and r are fixed letters, and u, u' are words already present in some components of the vector. The following equalities show that we can inductively compute the contexts and final contexts in $\mathcal{B}$ of newly added words in these functions: for every $u, u' \in \mathtt{C} \cup \mathtt{R}$, $c \in \Sigma_c$, $r \in \Sigma_r$, we have

$$ctx^{\mathcal{B}}(uu') = \{(p,q) \in \hat{Q}^2 \mid \exists p_i \in \hat{Q}, (p,p_i) \in ctx^{\mathcal{B}}(u), (p_i,q) \in ctx^{\mathcal{B}}(u')\},$$
$$ctx^{\mathcal{B}}(cur) = \{(p,q) \in \hat{Q}^2 \mid \exists (p',q') \in ctx^{\mathcal{B}}(u), \exists \gamma \in \hat{\Gamma}, (p,c,p',\gamma) \in \hat{\delta}_c, (q',r,\gamma,q) \in \hat{\delta}_r\} \ .$$

The definitions for $ctx^{\mathcal{B}}_{\circledast}(uu')$ and $ctx^{\mathcal{B}}_{\circledast}(cur)$ are left as exercise to the reader.

Example 23. *Using the above definition it is routine to check that* $ctx^{\mathcal{B}}(c\,r) = \{(p,q),(q,q)\}$ *because* $cr = c\epsilon r$, $ctx^{\mathcal{B}}(\epsilon) = \{(p,p),(q,q)\}$ *(Example 21) and* $(p,c,q,A),(q,c,q,A) \in \hat{\delta}_c$, $(q,r,A,q) \in \hat{\delta}_r$.

Using the context information cached along words we check convergence of the fixpoint computations (lines 1–2) using the following quasiorders directly on contexts $\sqsubseteq_{\subseteq}$ on $\wp(\wp(\hat{Q}^2))^4$ for prefixes and $\sqsubseteq_{\subseteq\times\subseteq}$ on $\wp(\wp(\hat{Q}^2) \times \wp(\hat{Q}^2))^6$ for periods.

Incidentally, as we show below, we can perform the membership checks of lines 5 and 7 (asking whether $uv^\omega \in L^\omega(\mathcal{B})$ given u and v) using the context information associated to the prefix u and period v and nothing else.

7.7.2 Membership Check

To decide membership in $L^\omega(\mathcal{B})$ we use the membership predicate $\mathrm{Inc}^{\mathcal{B}}$ defined for $x, y_1, y_2 \in \wp(\hat{Q}^2)$ as follows:

$$\mathrm{Inc}^{\mathcal{B}}(x, y_1, y_2) \triangleq \exists q, p \in \hat{Q}, (\hat{q_I}, q) \in x \wedge (q,p) \in y_1^* \wedge (p,p) \in y_1^* \circ y_2 \circ y_1^* \ ,$$

where, given two binary relations $y, y' \in \wp(\hat{Q}^2)$ on states of $\mathcal{B}$, the notation $y \circ y'$ denotes their composition, and y^* denotes the Kleene closure of y.

Proposition 23. *For all* $(u,v) \in \texttt{Ld}$, $Inc^{\mathcal{B}}(ctx^{\mathcal{B}}(u), ctx^{\mathcal{B}}(v), ctx^{\mathcal{B}}_{\circledast}(v)) \iff uv^\omega \in L^\omega(\mathcal{B})$.

Proof. Let $(u,v) \in \texttt{Ld}$. Note that if $v \in \texttt{C}$ (resp. $v \in \texttt{R}$) then for every positive integer n we have $v^n \in \texttt{C}$ (resp. $v^n \in \texttt{R}$) and $(p,q) \in \mathrm{ctx}^{\mathcal{B}}(v)^* \iff \exists n, (p,q) \in ctx^{\mathcal{B}}(v^n)$. Therefore, if $\mathrm{Inc}^{\mathcal{B}}(ctx^{\mathcal{B}}(u), \mathrm{ctx}^{\mathcal{B}}(v), \mathrm{ctx}^{\mathcal{B}}_{\circledast}(v))$ holds then there are $q, p \in \hat{Q}$ and two positive integers n, m such that $(\hat{q_I}, q) \in ctx^{\mathcal{B}}(u)$, $(q,p) \in \mathrm{ctx}^{\mathcal{B}}(v^n)$ and $(p,p) \in \mathrm{ctx}^{\mathcal{B}}_{\circledast}(v^m)$. If $(u,v) \in \texttt{C} \times \texttt{C}$ then we deduce an accepting run of $\mathcal{B}$ on uv^ω of the form $(\hat{q_I}, \bot) \vdash^{*u} (q, \bot) \vdash^{*v^n} (p, \bot) \vdash^{\circledast v^m} (p, \bot)$ for uv^ω. If $(u,v) \in \texttt{U}_\texttt{c} \times \texttt{R}$ then we deduce an accepting run of $\mathcal{B}$ on uv^ω of the form $(\hat{q_I}, \bot) \vdash^{*u} (q, w) \vdash^{*v^n} (p, ww') \vdash^{\circledast v^m} (p, ww'w'')$ for some $w, w', w'' \in \Gamma$.

Conversely if $uv^\omega \in L^\omega(\mathcal{B})$ then there is an accepting run of $\mathcal{B}$ on uv^ω.

- If $(u,v) \in \texttt{C} \times \texttt{C}$ then this run is of the form

  $$(\hat{q_I}, \bot) \vdash^{*u} (q, \bot) \vdash^{*v} (q_1, \bot) \vdash^{*v} (q_2, \bot) \vdash^{*v} \cdots$$

  Since $\hat{Q}$ is finite, there is $p \in \hat{Q}$ and a sequence $\{n_k\}_{k \in \mathbb{N}}$ such that $q_{n_k} = p$ for all $k \in \mathbb{N}$. Since the run is accepting there is $m \in \mathbb{N}$ such that $(p, \bot) \vdash^{\circledast v^m} (p, \bot)$.

- If $(u,v) \in \texttt{U}_\texttt{c} \times \texttt{R}$ then it is of the form

  $$(\hat{q_I}, \bot) \vdash^{*u} (q, w_0) \vdash^{*v} (q_1, w_1) \vdash^{*v} (q_2, w_1 w_2) \vdash^{*v} \cdots$$

  where for each $j \in \mathbb{N}$ no symbol of w_j is popped while reading v in the sequence of transitions $(q_j, w_j) \vdash^{*v} (q_{j+1}, w_j w_{j+1})$. Thus, we can derive sequences $(q_j, \bot) \vdash^{*v} (q_{j+1}, w_{j+1})$ for every $j \in \mathbb{N}$. There is $p \in \hat{Q}$ and a sequence $\{n_k\}_{k \in \mathbb{N}}$ such that $q_{n_k} = p$ for all $k \in \mathbb{N}$ and since the run is accepting there is $m \in \mathbb{N}$ such that $(p, \bot) \vdash^{\circledast v^m} (p, w_{n_j} \cdots w_{n_{j+m}})$.

In both cases we deduce that $(\hat{q_I}, q) \in ctx^{\mathcal{B}}(u)$, $(q,p) \in ctx^{\mathcal{B}}(v^{n_0})$ and $(p,p) \in \mathrm{ctx}^{\mathcal{B}}_{\circledast}(v^m)$. Thus, $\mathrm{Inc}^{\mathcal{B}}(ctx^{\mathcal{B}}(u), \mathrm{ctx}^{\mathcal{B}}(v), \mathrm{ctx}^{\mathcal{B}}_{\circledast}(v))$ holds.

□

By showing how to reason on contexts directly (for comparisons, for applying functions $f_{\mathcal{A}}$ and $r_{\mathcal{A}}$, for convergence check and for membership check) we removed the need to store words altogether since their contexts suffice. To sum up, Algorithm 6 instantiated with the state-based quasiorders can be implemented directly manipulating subsets of $\wp(\hat{Q}^2)$ (for the prefixes) and pairs of subsets of $\wp(\hat{Q}^2)$ (for the periods) thereby removing the need to store and manipulate words. We call this implementation of Algorithm 6 the *state-based algorithm*. We conclude this section with its complexity.

Proposition 24. *Let* $n \triangleq |Q|$, $\hat{n} \triangleq |\hat{Q}|$ *and* $m \triangleq max\{1, |\Sigma|\}$. *The running time of the state-based algorithm is* $2^{O(\hat{n}^2)} m^2 n^4$.

Proof. We assume that computing the compositions $x \circ y$ for $x, y \in \wp(\hat{Q}^2)$ and checking the inclusion $x \subseteq y$ take $O(\hat{n}^4)$. The number of composition operations needed for each component

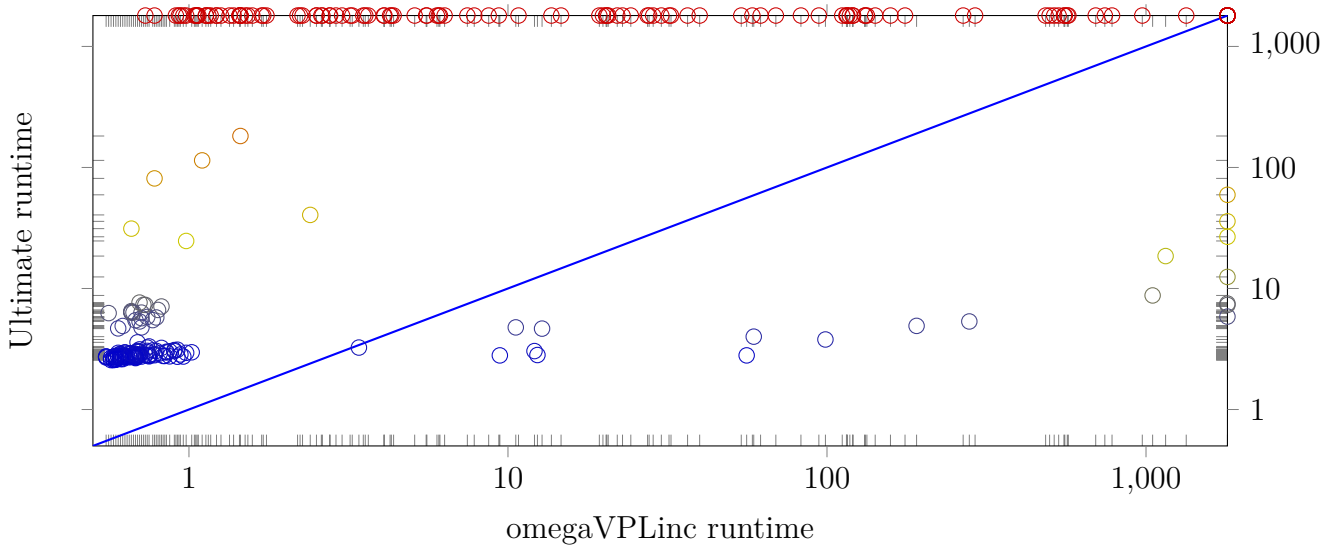


Figure 7.1: Scatter plot comparing the runtime (in seconds) of Ultimate and omegaVPLinc on the Ultimate suite. Both axis feature a logarithmic scale. When a tool does not return an answer within 1800 seconds (it runs out of time or memory) the data point is plotted on the edge thereof (top edge for Ultimate, right edge for omegaVPLinc).

of each function W, C, R, U, $W_\circledast$, $C_\circledast$, $R_\circledast$ is bounded by $2^{O(\hat{n}^2)}m^2n^2$. Since each function W, C, R, U, $W_\circledast$, $C_\circledast$, $R_\circledast$ has n^2 components it takes $2^{O(\hat{n}^2)}m^2n^4$ to compute $\mathrm{ctx}^{\mathcal{B}}(f_{\mathcal{A}}(X,Y,Z,U))$ and $\mathrm{ctx}^{\mathcal{B}}_\circledast(r_{\mathcal{A}}(X,X',Y,Y',Z,Z'))$ given $\mathrm{ctx}^{\mathcal{B}}(X,Y,Z,U)$ and $\mathrm{ctx}^{\mathcal{B}}_\circledast(X,X',Y,Y',Z,Z')$. For all $k \in \mathbb{N}$, we have that $\mathrm{ctx}^{\mathcal{B}}_\circledast({r_{\mathcal{A}}}^k(\vec{\emptyset})) \subseteq \left(\wp(\hat{Q}^2) \times \wp(\hat{Q}^2)\right)^{6\cdot|Q|^2}$. At worst the computation of line 2 adds exactly one element of $\wp(\hat{Q}^2) \times \wp(\hat{Q}^2)$ at each iteration step k to some entry of the $6 \cdot |Q|^2$-dimensional vector $\mathrm{ctx}^{\mathcal{B}}_\circledast({r_{\mathcal{A}}}^k(\vec{\emptyset}))$, so that $6n^2 2^{2\hat{n}^2}$ is an upper bound on the number of iterations needed to compute $\mathrm{ctx}^{\mathcal{B}}_\circledast({r_{\mathcal{A},\mathcal{B}}}^m(\vec{\emptyset}))$. For $X', Y' \subseteq \wp(\hat{Q}^2) \times \wp(\hat{Q}^2)$ the time to check $X' \sqsubseteq^\circledast Y'$ is bounded by $2^{O(\hat{n}^2)}$. The analogue holds for the Kleene iterates for the prefixes and the qo $\sqsubseteq$. It follows that lines 1 to 2 take at most $2^{O(\hat{n}^2)}m^2n^3$ time. For $x \in \wp(\hat{Q}^2)$ the time to compute the transitive closure x^* is bounded by $O(\hat{n}^8)$. Hence, the time to check $\mathrm{Inc}^{\mathcal{B}}(x, y_1, y_2)$ is bounded by $O(\hat{n}^{10})$. Since $|\mathrm{ctx}^{\mathcal{B}}({f_{\mathcal{A}}}^m(\vec{\emptyset}))|, |\mathrm{ctx}^{\mathcal{B}}_\circledast({r_{\mathcal{A}}}^m(\vec{\emptyset}))| \leq 2^{2\hat{n}^2}$ the time to execute lines 3 to 8 is bounded by $2^{O(\hat{n}^2)}n$.

□

7.8 Experiments

We implemented *omegaVPLinc* [27] , a Java prototype of the state-based algorithm and evaluated it against Ultimate from Heizmann et al. [44] which decides inclusion via complementation, intersection and emptiness check.[2]

[2]We excluded FADecider [38] from our evaluation because it returned 22 false positive answers on a randomly chosen subset of 50 from our 286 benchmarks. Counterexamples to inclusion for these benchmarks were validated with Ultimate. The problem has been reported.

Benchmarks. Our experiments use two sets of benchmarks. The first stems from [38] and consists of 5 queries $L^\omega(\mathcal{A}) \subseteq L^\omega(\mathcal{B})$ given $\mathcal{A}$ and $\mathcal{B}$. We first translated those VPA into the AutomataScript language that Ultimate and omegaVPLinc can use and then we minimized them with Ultimate. The second set of benchmarks consists of 281 instances of VPA $\mathcal{A}, \mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n$ for which we run the query $L^\omega(\mathcal{A}) \subseteq \bigcup_{i=1}^{n} L^\omega(\mathcal{B}_i)$. These VPA were computed by Ultimate from randomly selected tasks in SV-COMP (Software Verification Competition) termination category. We used Ultimate to compute the unions of $\mathcal{B}_1, \ldots, \mathcal{B}_n$ and then minimize the result before running each query.

Experimental Setup. We ran our experiment in Debian/GNU Linux 11 (Bullseye) 64bit, running on a server with 20 GB of RAM and 2 Xeon E5640 2.6 GHz CPUs. We used Ultimate version 0.2.1, with openJDK 11.0.13, whereas omegaVPLinc uses openJDK 17.0.1. Maximal heap size for both programs was set to 6 GB and they were given a timeout of 30 minutes (or, equivalently, 1800 seconds).

Results. Of the 5 benchmarks in the FADecider suite, omegaVPLinc is faster on 4 of them. Our prototype times out on the remaining one, while Ultimate runs out of memory. Of the 281 benchmarks in the Ultimate suite, omegaVPLinc correctly returns an answer on 253 (165 $\subseteq$ and 88 $\not\subseteq$), times out on 27 and runs out of memory on 1. Ultimate, however, only terminates on 142 benchmarks, running out of memory on the remaining 139 (the red data points on the top edge in Figure 7.1). There are 7 benchmarks for which Ultimate terminates, but omegaVPLinc doesn't (the data points on the right edge but not the top one), whereas there are 118 benchmarks for which omegaVPLinc terminates, but Ultimate doesn't (the red data points on the top edge but not the right one). Of the 135 benchmarks on which both tools terminate, omegaVPLinc is faster than Ultimate on 123 (data points touching no edges and above the diagonal). Moreover omegaVPLinc and Ultimate coincide on whether inclusion holds (98) or not (37). This empirical evaluation suggests that omegaVPLinc scales up better than Ultimate on both of these benchmark sets.

Chapter 8

A MYHILL-NERODE THEOREM FOR TIMED AUTOMATA WITH INTEGER RESETS

We consider the subclass of timed automata with integer resets, which is known to have good automata-theoretic properties and is also useful for practical modeling. We present a Nerode-style equivalence for this class that depends on a constant K and leads to the construction of a canonical one-clock integer-reset timed automaton with maximum constant K.

8.1 Languages with Integer Resets

We define the class of *One-Clock Integer Reset Timed Automata* (1-IRTA) where transitions reset the clock provided its value is an integer. Formally, we say that a 1-TA $\mathcal{A} = (Q, q_I, T, F)$ is a 1-IRTA when for every resetting transition $(q, q', a, \phi, 0) \in T$ the clock constraint ϕ is of the form $x = m$, or, equivalently, $[\![\phi]\!] \in \mathbb{N}$. A deterministic 1-IRTA is called a 1-IRDTA.

Example 24. *The* 1*-TAs in Figures 1.4, 1.5 and 8.1 are* 1*-IRTAs.*

The definition of a run of a 1-IRTA simply follows from that of 1-TA. However, due to the special syntax, we can identify points in the word where the resets can potentially happen. The next definition makes this idea precise.

Definition 8.1.1. Given $d = t_1 \cdots t_n \in \mathbb{T}$ and a $K \in \mathbb{N}$, we define the longest sequence of indices $s_d = \{0 = i_0 < i_1 < \ldots < i_p \leq n\}$ such that for every $j \in \{0, \ldots, p-1\}$ the value $\sum_{i=(i_j)+1}^{i_{(j+1)}} t_i$ is an integer between 0 and K. We refer to the set of positions of the sequence s_d as the *integral positions* of d. Note that s_d is never empty since it always contains 0. Next, define

$$c^K(d) = \sum_{i=(i_p)+1}^{n} t_i \ .$$

The definitions of integral positions and the function c^K apply equally to timed words by

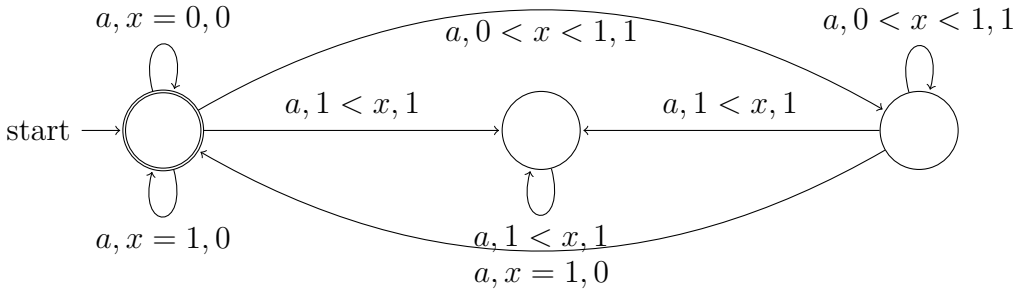


Figure 8.1: A strict 1-IRTA with alphabet $\Sigma = \{a\}$ accepting $M = \{u \in \mathbb{T}\Sigma^* \mid c^1(u) = 0\}$.

taking the timestamp of the timed word.

Example 25. *For $K = 1$ and $u = (0.2 \cdot a)(0.8 \cdot a)(0.2 \cdot a)$ we have $s_u = \{0 < 2\}$ and $c^1(u) = 0.2$. For $K = 1$ and $u' = (1.2 \cdot a)(0.8 \cdot a)(0.2 \cdot a)$ we have $s_{u'} = \{0\}$ and $c^1(u') = 2.2$. For $K = 2$, we have $s_{u'} = \{0 < 2\}$ and $c^2(u') = 0.2$. Notice that the sequence $s_{u'}$ depends on the constant K (we do not explicitly add K to the notation for simplicity, as in our later usage, K will be clear from the context).*

Consider a run of a 1-IRTA on a word $u = (t_1 \cdot a_1) \cdots (t_n \cdot a_n) \in \mathbb{T}\Sigma^*$ and factor it according to $s_u = \{0 = i_0 < i_1 < \ldots < i_p \leq n\}$:

$$(q_{i_0}, \nu_{i_0}) \xrightarrow{t_{(i_0)+1}, \theta_{(i_0)+1} \cdots t_{i_1}, \theta_{i_1}} (q_{i_1}, \nu_{i_1}) \xrightarrow{t_{(i_1)+1}, \theta_{(i_1)+1} \cdots t_{i_2}, \theta_{i_2}} (q_{i_2} \nu_{i_2}) \to \cdots$$
$$\to (q_{i_p}, \nu_{i_p}) \xrightarrow{t_{(i_p)+1}, \theta_{(i_p)+1} \cdots t_n, \theta_n} (q_n, \nu_n)$$

At each position i_j with $j \in \{0, \ldots, p\}$, $\nu_{i_j} \in \mathbb{N}$ and, moreover, $\nu_{i_j} = 0$ when $r_{i_j} = 0$. Note that when $i_p = n$ then $c^K(u) = 0$, otherwise $c^K(u)$ can be any real value except an integer value between 0 and K, i.e. $c^K(u) \in \mathbb{R}_{\geq 0} \setminus \{0, \ldots, K\}$.

In Definition 8.1.1 we identified the integral positions at which a 1-IRTA *could potentially* reset the clock. In the following, we recall a subclass of 1-IRTAs called strict 1-IRTAs [10] where every transition with an equality guard ϕ (ϕ is of the form $x = m$ or, equivalently, $[\![\phi]\!] \in \mathbb{N}$) *must* reset the clock. This feature, along with a special requirement on guards forces a reset on every position given by s_u for a word u.

8.1.1 The subclass of strict 1-IRTA

A 1-IRTA is said to be *strict* if each of its transitions (q, q', a, ϕ, s) satisfies the following conditions:

1. the clock constraint of the guard ϕ is either $x = m$, $m < x \wedge x < m + 1$, or $K < x$,
2. the clock constraint of the guard ϕ is an equality iff $s = 0$.

Example 26. *The 1-TA in Figures 1.5 and 8.1 are strict 1-IRTAs. The 1-TA in Figure 1.4 is not a strict 1-IRTA since the transition $q_I \xrightarrow{a, x=1, 1} q$ does not reset the clock.*

A run of a strict 1-IRTA on a word u can be factored similarly as explained for a general 1-IRTA, however now, every r_{i_j} will be a reset transition: notice that we require each transition to be guarded using constraints of a special form, either $x = m$ or $m < x < m + 1$ or $K < x$; therefore, the transition reading (t_{i_1}, a_{i_1}) will necessarily have an equality guard $x = m$ forcing a reset, similarly at i_2 and so on. Therefore, the sequence s_u identifies the exact reset points in the word, no matter which strict 1-IRTA reads it. The quantity $c^K(u)$ gives the value of the clock on reading u by any strict 1-IRTA. This *input-determinism* is a fundamental property of strict 1-IRTAs that helps in the Myhill-Nerode characterization that we present in the later sections.

The question now is how expressive are strict 1-IRTAs. As shown by the proposition below, every language definable by a 1-IRTA is also definable by a strict 1-IRTA. Therefore, we could simply consider strict 1-IRTAs instead of 1-IRTAs. Even though a proof of this equi-expressivity theorem is given in [10, Theorem 1] we provide one in appendix for the sake of being self-contained.

Proposition 25 (see also Theorem 1 [10])**.** *A language accepted by a (deterministic)* 1*-IRTA is also accepted by a (deterministic) strict* 1*-IRTA with the same greatest constant in guards.*

Proof. Given a 1-TA we can always assume that its transitions are of the form given in item 1. We call a transition of the form $(q, q', a, x = m, 1)$ a *bad* transition. Next, we prove by induction on the number of bad transitions that the language of any (deterministic) 1-TA with transitions as in item 1 is also accepted by a (deterministic) 1-TA with no bad transitions. Let $\mathcal{A} = (Q, q_I, T, F)$ be a 1-TA with transitions as in item 1 and assume it has $n + 1$ bad transitions. Let $\theta = (p, q, a, x = m, 1) \in T$ be a bad transition such that the constant m is the largest among all the constants associated with bad transitions.

For a guard ϕ with a constant greater than m, define

$$\mathit{modify}(\phi) = \begin{cases} x = c - m & \text{if } \phi \text{ is } x = c \\ c - m < x \wedge x < c - m + 1 & \text{if } \phi \text{ is } c < x \wedge x < c + 1 \\ K - m < x & \text{else } (\phi \text{ is } K < x)\ . \end{cases}$$

Define the 1-TA $\overline{\mathcal{A}}$ as follows: The set of states is given by Q and a copy $\overline{Q}$ of Q and the initial state is $q_I \in Q$. The transitions are given by

1. the transitions of $T \setminus \{\theta\}$,
2. the transition $(p, \overline{q}, a, x = m, 0)$,
3. for every non bad transition $\eta = (s, s', a, \phi, 1) \in T$ the transition $\overline{\eta} = (\overline{s}, \overline{s}', a, \mathit{modify}(\phi), 1)$,
4. for every bad transition $\eta = (s, s', a, x = m, 1) \in T$ of guard $x = m$ (including the transition θ) the transition $\eta = (\overline{s}, \overline{s}', a, x = 0, 0)$,
5. for every resetting transition $(s, s', a, \phi, 0) \in T$ the transition $(\overline{s}, s', a, \mathit{modify}(\phi), 0)$.

The set of final states is $F \cup \{\overline{f} \in \overline{Q} \mid f \in F\}$. Note that if $\mathcal{A}$ is deterministic then $\overline{\mathcal{A}}$ is also deterministic. All the bad transitions of $\overline{\mathcal{A}}$ are in $T \setminus \{\theta\}$, thus $\overline{\mathcal{A}}$ has n bad transitions. By induction hypothesis it is accepted by a 1-TA without any bad transitions. It remains to show

that $L(\mathcal{A}) = L(\overline{\mathcal{A}})$. The runs of $\mathcal{A}$ that don't use the transition θ correspond to the runs of $\overline{\mathcal{A}}$ that don't use the transition of item 2. $\overline{\mathcal{A}}$ simulates a run of $\mathcal{A}$ that uses θ as follows: Until the first occurrence of θ the run is simulated by the transitions in item 1. Every occurrence of θ in the run is simulated either by the transition in item 2 (in particular this is the case for the first occurrence of θ) or by the "good" version of θ given in item 4. After an occurrence of θ the automaton $\overline{\mathcal{A}}$ uses the transitions of items 3 and 4 until a resetting transition occurs. Such a resetting transition is simulated with the corresponding transition from item 5 which forces $\overline{\mathcal{A}}$ back to the states of Q until a next occurrence of θ. □

8.2 Strict 1-IRDTA from Equivalence on Timed Words

We now start our search for a Nerode-style equivalence for timed languages with integer resets. In this section, we define a strict 1-IRDTA from an equivalence on timed words to accept a given language. As expected, we need some conditions on the equivalence.

Definition 8.2.1. Given a constant $K \in \mathbb{N}$, an equivalence relation $\approx \subseteq \mathbb{T}\Sigma^* \times \mathbb{T}\Sigma^*$ is *K-monotonic* when

(a) $u \approx v \implies c^K(u) \equiv^K c^K(v)$,

(b) $u \approx v \implies \forall a \in \Sigma, \forall t \in \mathbb{R}_{\geq 0}, \exists t' \in [t]_{\equiv^K},\ u(t \cdot a) \approx v(t' \cdot a)$,

(c) $\forall u \in \mathbb{T}\Sigma^*, \forall t, t' \in \mathbb{R}_{\geq 0},\ c^K(u) + t \equiv^K c^K(u) + t' \implies u(t \cdot a) \approx u(t' \cdot a)$ for all $a \in \Sigma$.

Here is some intuition behind the definition. When we build the strict 1-IRDTA with the equivalence classes as states, we want every word in an equivalence class $[u]_\approx$ to take an outgoing transition along with u. If we do not impose **(a)**, then there could be two words u, v with say $c^K(u) = 0.5$ and $c^K(v) = 0$ within the same class. Then, an outgoing transition with $x = 0$ is taken only by v and not u. This is an unwanted situation. Condition **(b)** says that if u on a "letter" jumps to a class, v on a "similar letter" can jump to the same class. Finally, the third condition **(c)** imposes some kind of completeness w.r.t. guards: every letter obtained by a time delay that satisfies a particular clock constraint should end up in the same equivalence class.

We now move on to the actual strict 1-IRDTA construction. Given a K-monotonic and finite index equivalence $\approx \subseteq \mathbb{T}\Sigma^* \times \mathbb{T}\Sigma^*$ and, moreover, a language $L \subseteq \mathbb{T}\Sigma^*$ we define a 1-TA $\mathcal{A}^L_\approx$ as follows. The set of states is $[\mathbb{T}\Sigma^*]_\approx$, and the initial state is $[\epsilon]_\approx$. To define the transitions we introduce the following notations.

- For a class $[t]_{\equiv^K} \in [\mathbb{R}_{\geq 0}]_{\equiv^K}$, we define the guard $\phi_{[t]_{\equiv^K}}$ as

$$\phi_{[t]_{\equiv^K}} = \begin{cases} x = t & \text{if } t \leq K \wedge t \in \mathbb{N} \\ \lfloor t \rfloor < x \wedge x < \lfloor t \rfloor + 1 & \text{if } t < K \wedge t \notin \mathbb{N} \\ K < x & \text{if } K < t\ . \end{cases}$$

- For a class $[v]_\approx \in [\mathbb{T}\Sigma^*]_\approx$ define $s_{[v]_\approx} \in \{0, 1\}$ as 0 if $c^K(v) = 0$ and 1 otherwise.

It is straightforward that $\phi_{[t]_{\equiv^K}}$ depends solely on the equivalence class $[t]_{\equiv^K}$ i.e., $t \equiv^K t' \implies$

$\phi_{[t]_{\equiv^K}} = \phi_{[t']_{\equiv^K}}$. Regarding $s_{[v]_\approx}$ the same holds due to the K-monotonicity of $\approx$, which guarantees that if $v \approx v'$, then $c^K(v) \equiv^K c^K(v')$.

In turn, we define a transition $([u]_\approx, [v]_\approx, a, \phi_{[x]_{\equiv^K}}, s_{[v]_\approx}) \in T_\approx$ iff there is $t \in \mathbb{R}_{\geq 0}$ such that $u(t \cdot a) \approx v$ and $c^K(u) + t \equiv^K x$. As explained earlier, this transition is independent of the representatives chosen for the classes $[v]_\approx$ and $[x]_{\equiv^K}$. Next we show that it is also independent of the representative chosen for $[u]_\approx$.

Let $u \approx u'$. Since $\approx$ is K-monotonic for every $t \in \mathbb{R}_{\geq 0}$ there is $t' \in [t]_{\equiv^K}$ such that $u(t \cdot a) \approx u'(t' \cdot a)$. By transitivity of $\approx$, we get $u'(t' \cdot a) \approx v$. It remains to show that $c^K(u') + t' \equiv^K x$. By K-monotonicity we have: $c^K(u) \equiv^K c^K(u')$ and $c^K(u(t \cdot a)) \equiv^K c^K(u'(t' \cdot a))$. By our choice of t', we have $t \equiv^K t'$. This implies $c^K(u) + t \equiv^K c^K(u') + t'$. By transitivity of $\equiv^K$, we get $c^K(u') + t' \equiv^K x$. Thus, $([u]_\approx, [v]_\approx, a, \phi_{[x]_{\equiv^K}}, s_{[v]_\approx})$ and $([u']_\approx, [v]_\approx, a, \phi_{[x]_{\equiv^K}}, s_{[v]_\approx})$ coincide.

We conclude the definition $\mathcal{A}^L_\approx$ by giving the final states $F^L_\approx = \{[u]_\approx \in [\mathbb{T}\Sigma^*]_\approx \mid u \in L\}$. This set is well defined when $\approx$ is L-preserving i.e., when $u \approx v \implies (u \in L \iff v \in L)$. In turn we define the 1-TA $\mathcal{A}^L_\approx = ([\mathbb{T}\Sigma^*]_\approx, [\epsilon]_\approx, T_\approx, F^L_\approx)$.

Proposition 26. *Suppose $\approx$ is K-monotonic. Then $\mathcal{A}^L_\approx$ is a 1-IRDTA. Moreover, for every timed word u, $([\epsilon]_\approx, 0) \rightsquigarrow^u ([u]_\approx, c^K(u))$.*

Proof. By construction $\mathcal{A}^L_\approx$ is a strict 1-IRTA. We will show that it is also deterministic. Suppose there are two transitions $[u]_\approx \xrightarrow{a,\phi,r} [u_1]_\approx$ and $[u]_\approx \xrightarrow{a,\phi,r} [u_2]_\approx$ for $u_1 \not\approx u_2$. By definition of the transitions, there are $t_1, t_2 \in \mathbb{R}_{\geq 0}$ (corresponding to the two transitions) such that (1) $c^K(u) + t_1$ and $c^K(u) + t_2$ satisfy the same ϕ, and (2) $u(t_1 \cdot a) \approx u_1$ and $u(t_2 \cdot a) \approx u_2$. Point (1) will imply $c^K(u) + t_1 \equiv^K c^K(u) + t_2$. By condition **(c)** of K-monotonicity, we have $u(t_1 \cdot a) \approx u(t_2 \cdot a)$. From point (2), we get $u_1 \approx u_2$, contradicting $u_1 \not\approx u_2$.

By induction on the length of the timed words we show that for every $u \in \mathbb{T}\Sigma^*$, $([\epsilon]_\approx, 0) \rightsquigarrow^u ([u]_\approx, c^K(u))$. Let $u(t \cdot a) \in \mathbb{T}\Sigma^*$. By induction hypothesis $([\epsilon]_\approx, 0) \rightsquigarrow^u ([u]_\approx, c^K(u))$. Moreover, by definition of $\mathcal{A}^L_\approx$, there is a transition $([u]_\approx, [ua^t]_\approx, a, \phi_{[c^K(u)+t]_{\equiv^K}}, s_{[ua^t]_\approx}) \in T_\approx$. Since $c^K(u(t \cdot a)) = (c^K(u) + t)s_{[u(t \cdot a)]_\approx}$ we deduce that $([\epsilon]_\approx, 0) \rightsquigarrow^u ([u]_\approx, c^K(u)) \rightsquigarrow^{(t \cdot a)} ([u(t \cdot a)]_\approx, c^K(u(t \cdot a)))$. □

Corollary 1. *Let $\approx$ be K-monotonic, L-preserving and of finite index. Then $L = L(\mathcal{A}^L_\approx)$.*

Proof. From Proposition 26, a word is in L iff it has an accepting run. □

8.3 A Myhill-Nerode Theorem for Languages with Integer Resets

In the previous section, we have established generic conditions that we require out of an equivalence in order to obtain an equivalent strict 1-IRTA from it. In this section, we will present a concrete such equivalence: given a language L definable by a 1-IRTA and a constant $K \in \mathbb{N}$ we define a *syntactic equivalence* $\approx^{L,K} \subseteq \mathbb{T}\Sigma^* \times \mathbb{T}\Sigma^*$ for L. By syntactic we mean that this equivalence is independent of a specific representation of L. We then show that

when the constant K is sufficiently large, $\approx^{L,K}$ is the coarsest L-preserving and K-monotonic equivalence. As a consequence of this, we are able to define a canonical 1-IRDTA for L by taking the constant K to be minimal among the greatest constants of 1-IRDTAs accepting L.

The idea for defining $\approx^{L,K}$ is to identify two words u and u' whenever their clock values $c^K(u)$ and $c^K(u')$ are region equivalent and the residuals $u^{-1}L$ and $u'^{-1}L$ are the same modulo some rescaling w.r.t. $c^K(u)$ and $c^K(u')$.

To achieve this, we define a rescaling function $\tau_{u,v} : \mathbb{T} \to \mathbb{T}$ on timestamps. Expectedly, we require $\tau_{u,v}$ to be a length preserving bijection. This function $\tau_{u,v}$ is then extended to timed words by letting $\tau_{u,v}((t_1 \cdot a_1) \dots (t_n \cdot a_n)) = (t'_1 \cdot a_1) \dots (t'_n \cdot a_n)$ where $t'_1 \dots t'_n = \tau_{u,v}(t_1 \dots t_n)$. Hence, we will deem u and u' equivalent iff $c^K(u) \equiv^K c^K(u')$ and $\tau_{u,v}(u^{-1}L) = u'^{-1}L$.

8.3.1 Auxiliary Definitions

Identifying Timed Words. We extend the region equivalence to timestamps of any length and define the equivalence $\equiv^K \subseteq \mathbb{T} \times \mathbb{T}$ for timestamps, which plays a crucial role in our analysis of timed languages with integer resets.

Formally, for timestamps $d = t_1 \cdots t_n$ and $d' = t'_1 \cdots t'_n$, both of the same length n, we have $d \equiv^K d'$ iff every 1-IRTA with greatest constant less than or equal to K that accepts $(t_1 \cdot a_1) \cdots (t_n \cdot a_n)$ also accepts $(t'_1 \cdot a_1) \cdots (t'_n \cdot a_n)$, and vice versa. Alternatively, we have $d \equiv^K d'$ iff the following conditions hold:

- the integral positions of d and d' coincide: $s_d = s_{d'}$
  where $s_d = \{0 = i_0 < i_1 < \ldots < i_p \leq n\}$ (see Definition 8.1.1),
- for every $j \in \{0, \dots, p-1\}$ and every $s \in \{(i_j) + 1, \dots, i_{(j+1)}\}$,
  $t_{(i_j)+1} + \cdots + t_s \equiv^K t'_{(i_j)+1} + \cdots + t'_s$ and, moreover, if $i_p < n$ then the following holds
  $t_{(i_p)+1} + \cdots + t_{s'} \equiv^K t'_{(i_p)+1} + \cdots + t'_{s'}$ for all $s' \in \{(i_p) + 1, \dots, n\}$.

It is an easy exercise to check that for timestamps of length one this definition coincides with the region equivalence on $\mathbb{R}_{\geq 0}$.

We extend $\equiv^K$ to timed words by defining for $u = (t_1 \cdot a_1) \dots (t_n \cdot a_n)$ and $u' = (t'_1 \cdot b_1) \dots (t'_n \cdot b_n)$, $u \equiv^K u'$ iff $a_1 \dots a_n = b'_1 \dots b'_n$ and $t_1 \dots t_n \equiv^K t'_1 \dots t'_n$.

Identifying Residual Languages. In the following we fix $x, x' \in \mathbb{R}_{\geq 0}$ such that $x \equiv^K x'$ and define a *length preserving bijection* $\tau_{x,x'} : \mathbb{T} \to \mathbb{T}$ verifying the following properties

1. for every $t, t' \in \mathbb{T}$ there exists $z \in \mathbb{T}$ such that $\tau_{x,x'}(tt') = \tau(t)z$,
2. for every $t \in \mathbb{T}$, $x'\tau_{x,x'}(t) \equiv^K xt$.

Before we provide the construction of the $\tau_{x,x'}$ function, we give an illustrative example. Assume $K = 1$, and $0 < x, x' < 1$. Let us look at $\tau_{x,x'}$ for timestamps $y \in \mathbb{R}_{\geq 0}$ of length 1. To satisfy property 2, we require $x + y \equiv^1 x' + \tau_{x,x'}(y)$: that is, $x + y < 1 \iff x' + y' < 1$ and $x + y = 1 \iff x' + y' = 1$. One can achieve this by rescaling the interval $1 - x$ to $1 - x'$. The picture below depicts the situation for a y smaller than 1. In this case, we can set $y' = \frac{(1-x')}{(1-x)}y$. When $y \geq 1$, we can set y' to have the same integral part as y, and apply the

rescaling for the fractional parts.

We will now give the general construction of $\tau_{x,x'}$. We define $\tau_{x,x'}$ inductively on the length of the timestamps. Set $\tau_{x,x'}(\epsilon) = \epsilon$. Consider $n+1$ length timestamps $t_1 t_2 \dots t_n t_{n+1}$. To satisfy property **1**, we require $\tau_{x,x'}(t_1 t_2 \dots t_n t_{n+1})$ to be of the form $\tau_{x,x'}(t_1 t_2 \dots t_n)z$ where $z \in \mathbb{R}_{\geq 0}$. As mentioned in the example above, we will set the integral part of z to be the same as that of t_{n+1}. For the fractional part, we will do the rescaling. More precisely, we will define a bijection $f_{t_1 \cdots t_n} \colon [0,1) \to [0,1)$ so that we can set $\tau_{x,x'}(t_1 t_2 \dots t_n t_{n+1}) = \tau_{x,x'}(t_1 \dots t_n) t'_{n+1}$ such that $\lfloor t'_{n+1} \rfloor = \lfloor t_{n+1} \rfloor$ and $\mathit{frac}(t'_{n+1}) = f_{t_1 \cdots t_n}(\mathit{frac}(t_{n+1}))$. It now remains to define this bijection $f_{t_1 \cdots t_n}$.

Assume we have $x' \tau_{x,x'}(t_1 t_2 \dots t_n) \equiv^K x t_1 t_2 \dots t_n$. We want $\tau_{x,x'}$ to satisfy property **1** for $t_1 t_2 \dots t_{n+1}$, that is $x' \tau_{x,x'}(t_1 t_2 \dots t_n) t'_{n+1} \equiv^K x t_1 t_2 \dots t_n t_{n+1}$. Since $x' \tau_{x,x'}(t_1 t_2 \dots t_n) \equiv^K x t_1 t_2 \dots t_n$ by our inductive assumption, the timestamps $x t_1 \cdots t_n$ and $x' \tau_{x,x'}(t_1 t_2 \dots t_n)$ have the same integral positions given by the sequence $s = \{0 = i_0 < i_1 < \ldots < i_p \leq n\}$ (Def. 8.1.1). Hence, according to the characterization of the equivalence classes of $\equiv^K$ given in the first part of Section 8.3.1, it suffices to show the following property:

$$t_{(i_p)+1} + \cdots + t_n + t_{n+1} \equiv^K t'_{(i_p)+1} + \cdots + t'_n + t'_{n+1} \text{ where } t'_1 \dots t'_n = \tau_{x,x'}(t_1 \dots t_n) \ .$$

If $i_p = n$ then we can take $t'_{n+1} = t_{n+1}$. If $i_p < n$ then this property is satisfied if we take $f_{t_1 \cdots t_n} : t \in [0,1) \mapsto \frac{1 - \mathit{frac}(t'_{(i_p)+1} + \cdots + t'_n)}{1 - \mathit{frac}(t_{(i_p)+1} + \cdots + t_n)} t$.

Note that when $x, x' \in \{0, \dots, K\}$, $\tau_{x,x'}$ is the identity function. Also we have $\tau_{x',x} = \tau^{-1}_{x,x'}$.

Lemma 10. *Let $\mathcal{A}$ be a 1-IRTA with greatest constant less than or equal to K and $x, x' \in \mathbb{R}_{\geq 0}$ such that $x \equiv^K x'$. For every state q of $\mathcal{A}$ we have $\tau_{x,x'}(\mathcal{L}(q,x)) = \mathcal{L}(q,x')$.*

Proof. Let $\mathcal{A} = (Q, q_I, T, F)$ be the 1-IRTA with greatest constant less than or equal to K, with $q \in Q$. Let x, x' such that $x \equiv^K x'$. If $x \in \{0, 1, \dots, K\}$, then $x = x'$ and the lemma is trivial since $\tau_{x,x'}$ is the identity function. Assume $x \neq x'$, and therefore neither of them is an integer smaller than or equal to K.

Define $\overline{\mathcal{A}}$ to be the 1-TA obtained by adding a new state $\overline{q} \notin Q$ to $\mathcal{A}$ and the transition $(\overline{q}, q, a, \phi_{[x]_{\equiv^K}}, 1)$. $\overline{\mathcal{A}}$ is a 1-IRTA and for every $w \in \mathbb{T}\Sigma^*$ and $y \in [x]_{\equiv^K}$ we have $w \in \mathcal{L}(q,y) \iff (y \cdot a)w \in \mathcal{L}(\overline{q}, 0)$. Pick $w \in \mathcal{L}(q,x)$. We will show that $\tau_{x,x'}(w) \in \mathcal{L}(q,x')$. This will establish $\tau_{x,x'}(\mathcal{L}(q,x)) \subseteq \mathcal{L}(q,x')$. To deduce the reverse inclusion, we can use the fact that $\mathcal{L}(q,x) = \tau^{-1}_{x,x'}(\mathcal{L}(q,x'))$ and apply a symmetric argument.

To show $\tau_{x,x'}(w) \in \mathcal{L}(q,x')$. Notice that $(x \cdot a)w \in \mathcal{L}(\overline{q}, 0)$. We have $(x \cdot a)w \equiv^K (x' \cdot a)\tau_{x,x'}(w)$ by definition of $\tau_{x,x'}$. Thus, by definition of $\equiv^K$ we have $(x \cdot a)w \in \mathcal{L}(\overline{q}, 0) \iff (x' \cdot a)\tau_{x,x'}(w) \in \mathcal{L}(\overline{q}, 0)$. Overall, $w \in \mathcal{L}(q,x) \iff (x \cdot a)w \in \mathcal{L}(\overline{q}, 0) \iff (x' \cdot a)\tau_{x,x'}(w) \in \mathcal{L}(\overline{q}, 0) \iff \tau_{x,x'}(w) \in \mathcal{L}(q,x')$. □

Lemma 11. *Assume $x \equiv^K x'$, $c \in \mathbb{R}_{\geq 0}$ and $d \in \mathbb{T}$. Then $c^K(xc) \equiv^K c^K(x' \tau_{x,x'}(c))$ and $\tau_{x,x'}(cd) = \tau_{x,x'}(c)\tau(d)$ where $\tau = \tau_{c^K(xc), c^K(x' \tau_{x,x'}(c))}$.*

Proof. Let $x \equiv^K x'$ and $c \in \mathbb{R}_{\geq 0}$. By the property **2** of $\tau_{x,x'}$ we have $c^K(xc) \equiv^K c^K(x'\tau_{x,x'}(c))$. By induction on the length of the timestamps we show that for every $d \in \mathbb{T}$ we have $\tau_{x,x'}(cd) = \tau_{x,x'}(c)\tau_{y,y'}(d)$ where $y = c^K(xc)$ and $y' = c^K(x'\tau_{x,x'}(c))$. It is straightforward for $d = \epsilon$. By the property **1** for $\tau_{x,x'}$, $\tau_{x,x'}(cdt) = \tau_{x,x'}(cd)t'$ for some $t' \in \mathbb{R}_{\geq 0}$. Thus, by the induction hypothesis, $\tau_{x,x'}(cdt) = \tau_{x,x'}(c)\tau_{y,y'}(d)t'$. Similarly, by the property **1** for $\tau_{y,y'}$, $\tau_{x,x'}(c)\tau_{y,y'}(dt) = \tau_{x,x'}(c)\tau_{y,y'}(d)t''$ for some $t'' \in \mathbb{R}_{\geq 0}$. Let i_p be the last term of the sequence s_{xcd} and, i'_p be the last term of the sequence s_{yd}. It is an easy exercise to check that $i_p = 3 \iff i'_p = 2$, $i_p = 2 \iff i'_p = 1$ and $i_p \in \{0, 1\} \iff i'_p = 0$, and, in any of these cases $t' = t''$. □

8.3.2 Syntactic Equivalence

Given L and K we are now in position to define a syntactic equivalence $\approx^{L,K} \subseteq \mathbb{T}\Sigma^* \times \mathbb{T}\Sigma^*$.

To simplify our notations, for any $u, v \in \mathbb{T}\Sigma^*$ such that $c^K(u) \equiv^K c^K(v)$, we write $\tau_{u,v}$ for the bijection $\tau_{c^K(u),c^K(v)}$.

Definition 8.3.1. We define $u \approx^{L,K} v$ iff $c^K(u) \equiv^K c^K(v)$ and $\tau_{u,v}(u^{-1}L) = v^{-1}L$.

Note that the equivalence $\approx^{L,K}$ is L-preserving. Assume $u \approx^{L,K} v$. We have $u \in L \iff \epsilon \in u^{-1}L$ and $\epsilon \in u^{-1}L \iff \epsilon \in v^{-1}L$ since $\tau_{u,v}(\epsilon) = \epsilon$. Thus, $u \in L \iff v \in L$.

Proposition 27. *For every $L \subseteq \mathbb{T}\Sigma^*$ and $K \in \mathbb{N}$ the equivalence $\approx^{L,K}$ verifies the K-monotonicity* **(a)** *and* **(b)**.

Proof. The K-monotonicity **(a)** holds by definition of $\approx^{L,K}$. Next we show the K-monotonicity **(b)**. Assume $u \approx^{L,K} v$. Let $a \in \Sigma$ and $x \in \mathbb{R}_{\geq 0}$. By definition of $\approx^{L,K}$ we have $c^K(u) \equiv^K c^K(v)$. We use Lemma 11. We have $c^K(u(x \cdot a)) \equiv^K c^K(v(\tau_{u,v}(x) \cdot a))$. Let $\tau = \tau_{u(x\cdot a),v(\tau_{u,v}(x)\cdot a)}$. We have:

$$
\begin{aligned}
& \tau(w) \in \tau((u(x \cdot a))^{-1}L) \\
\iff\ & u(x \cdot a)w \in L \iff \tau_{u,v}((x \cdot a)w) \in \tau_{u,v}(u^{-1}L) && \text{[since } \tau \text{ is a bijection]} \\
\iff\ & (\tau_{u,v}(x) \cdot a)\tau(w) \in \tau_{u,v}(u^{-1}L) && \text{[by Lemma 11]} \\
\iff\ & (\tau_{u,v}(x) \cdot a)\tau(w) \in v^{-1}L && \text{[since } \tau_{u,v}(u^{-1}L) = v^{-1}L] \\
\iff\ & \tau(w) \in (v(\tau_{u,v}(x) \cdot a)^{-1}L\ .
\end{aligned}
$$

Hence, $\tau((u(x \cdot a))^{-1}L) = (v(\tau_{u,v}(x) \cdot a))^{-1}L$. Thus, $u(x \cdot a) \approx^{L,K} v(\tau_{u,v}(x) \cdot a)$. Since $\tau_{u,v}(x) \equiv^K x$ (by definition of $\tau_{u,v}$) we conclude that $\approx^{L,K}$ verifies the **(b)**. □

The next theorem makes use of the above equivalence to present the Myhill-Nerode style characterization for IRTA languages.

Theorem 14.

- *$L \subseteq \mathbb{T}\Sigma^*$ is a language definable by a 1-IRTA if and only if there is a constant $K \in \mathbb{N}$ such that $\approx^{L,K}$ is K-monotonic and has finite index.*
- *$\approx^{L,K}$ is coarser than any K-monotonic and L-preserving equivalence.*

Proof.

- Since $\approx^{L,K}$ is L-preserving, Corollary 1 establishes the "if" part of the proof. By Proposition 27, $\approx^{L,K}$ always verifies the K-monotonicity **(a)** and **(b)**, no matter which L we choose. Next, we show that when $L \subseteq \mathbb{T}\Sigma^*$ is definable by a deterministic 1-IRTA with greatest constant less than or equal to K then $\approx^{L,K}$ has finite index and also verifies the K-monotonicity **(c)**. We fix a strict 1-IRDTA $\mathcal{A}$ with greatest constant less than or equal to K that accepts L, which exists by Proposition 25.

  - We show K-monotonicity **(c)**. Let $u \in \mathbb{T}\Sigma^*$ and $t, t' \in \mathbb{R}_{\geq 0}$ such that $c^K(u) + t \equiv^K c^K(u) + t'$. We have $c^K(u(t \cdot a)) \equiv^K c^K(u(t' \cdot a))$ for all $a \in \Sigma$. Let $x = c^K(u(t \cdot a))$ and $x' = c^K(u(t' \cdot a))$. Let q be the state of $\mathcal{A}$ reached by the word $u(t \cdot a)$. Since $c^K(u) + t \equiv^K c^K(u) + t'$, we deduce that q is the state reached on the word $u(t' \cdot a)$ as well. Hence, we have $(u(t \cdot a))^{-1}L = \mathcal{L}(q, x)$, and $(u(t' \cdot a))^{-1}L = \mathcal{L}(q, x')$. Since $x \equiv^K x'$, by Lemma 10, we deduce that $\tau_{x,x'}((u(t \cdot a))^{-1}L) = (u(t' \cdot a))^{-1}L$. Thus, $u(t \cdot a) \approx^{L,K} u(t' \cdot a)$.

  - Finally ,we show that $\approx^{L,K}$ has finite index. From the automaton $\mathcal{A}$ we can define a natural equivalence $\approx_{\mathcal{A}}$ as follows: $u \approx_{\mathcal{A}} v$ if $c^K(u) \equiv^K c^K(v)$ and $\mathcal{A}$ reaches the same (control) state on reading u and v from its initial configuration. It is easy to see that the number of equivalence classes of $\approx_{\mathcal{A}}$ is bounded by the number of states of $\mathcal{A}$ multiplied by the number of K regions. We will now prove that $\approx^{L,K}$ is coarser than $\approx_{\mathcal{A}}$, implying the finiteness of $\approx^{L,K}$.

    Suppose $u \approx_{\mathcal{A}} v$. Let $c^K(u) = x$ and $c^K(v) = x'$, and let q be the state reached by $\mathcal{A}$ on reading u and v. We have $u^{-1}L = \mathcal{L}(q, x)$ and $v^{-1}L = \mathcal{L}(q, x')$. Since $x \equiv^K x'$, Lemma 10 entails $\tau_{x,x'}(u^{-1}L) = v^{-1}L$, thereby proving $u \approx^{L,K} v$.

- To show the second part of the theorem we need the following preliminary result:

  Claim 1. *Let $\approx$ be K-monotonic. Then $u \approx v$ implies $uw \approx v\tau_{u,v}(w)$ for all $w \in \mathbb{T}\Sigma^*$.*

  Proof. By induction on the length of the words we show that for every $w \in \mathbb{T}\Sigma^*$ we have $uw \approx v\tau_{u,v}(w)$. Since $u \approx v$ it is true for ϵ. Assume it holds for $w \in \mathbb{T}\Sigma^*$ i.e., $uw \approx v\tau_{u,v}(w)$. Let $a \in \Sigma$ and $x \in \mathbb{R}_{\geq 0}$. We will show that $uw(x \cdot a) \approx v\tau_{u,v}(w(x \cdot a))$. By the property **1** of $\tau_{u,v}$ there is $y \in \mathbb{R}_{\geq 0}$ such that $\tau_{u,v}(w(x \cdot a)) = \tau_{u,v}(w)(y \cdot a)$. From the property **2** of $\tau_{u,v}$ we deduce $c^K(uw(x \cdot a)) \equiv^K c^K(v\tau_{u,v}(w)(y \cdot a))$. Since $\approx$ is K-monotonic there is $x' \in [y]_{\equiv^K}$ and $uw(x' \cdot a) \approx v\tau_{u,v}(w)(y \cdot a)$, thus $c^K(uw(x' \cdot a)) \equiv^K c^K(v\tau_{u,v}(w)(y \cdot a))$. Thus, by transitivity of $\equiv^K$ we deduce $c^K(uw(x \cdot a)) \equiv^K c^K(uw(x' \cdot a))$. Since $x' \equiv^K y$ and $y \equiv^K x$ (the last equivalence holds by definition of $\tau_{u,v}$) we have $x \equiv^K x'$. From $c^K(uw(x \cdot a)) \equiv^K c^K(uw(x' \cdot a))$ and $x \equiv^K x'$ we deduce that $c^K(uw) + x \equiv^K c^K(uw) + x'$ (if $c^K(uw(x \cdot a)) \in \mathbb{R}_{\geq 0} \setminus \{0\}$ it is straightforward, otherwise we have $c^K(uw) + x \in \mathbb{N}$ and $c^K(uw) + x' \in \mathbb{N}$ and we use that $x \equiv^K x'$). Hence, by K-monotonicity, $uw(x \cdot a) \approx uw(x' \cdot a)$. Finally, by transitivity of $\approx$ we find that $uw(x \cdot a) \approx v\tau_{u,v}(w(x \cdot a))$. □

  Let $\approx$ be a K-monotonic and L-preserving equivalence. Assume $u \approx v$. By K-monotonicity, $c^K(u) \equiv^K c^K(v)$. To prove that $u \approx^{L,K} v$ it remains to show that

$\tau_{u,v}(u^{-1}L) = v^{-1}L$. We have:

$$\begin{aligned}
\tau_{u,v}(w) \in \tau_{u,v}(u^{-1}L) &\iff w \in u^{-1}L && \text{[because } \tau_{u,v} \text{ is a bijection]}\\
&\iff uw \in L && \text{[by definition]}\\
&\iff v\tau_{u,v}(w) \in L && \text{[by Claim 1 and } L\text{-preservation]}\\
&\iff \tau_{u,v}(w) \in v^{-1}L\ . && \square
\end{aligned}$$

Assume L is definable by an 1-IRDTA with greatest constant less than or equal to K. The proof of Theorem 14 shows that $\approx^{L,K}$ is a K-monotonic finite index equivalence. Since it is also L-preserving we have $L = L(\mathcal{A}^L_{\approx^{L,K}})$. In the rest of the section, we provide examples that apply the above theorem.

Example 27. *Consider the language $L = \{(x \cdot a) \mid x \in \mathbb{N}\}$. It can intuitively be seen that this language cannot be accepted by any IRTA. We will now show that the right-hand-side of the characterization does not hold. There is no $K \in \mathbb{N}$ such that $\approx^{L,K}$ verifies the K-monotonicity* (c)*: For every $K \in \mathbb{N}$ we have $c^K(\epsilon) + K + 1 \equiv^K c^K(\epsilon) + K + 1.1$. Since $(K + 1 \cdot a) \in L$ and $(K + 1.1 \cdot a) \notin L$, by L-preservation, $(K + 1 \cdot a) \not\approx^{L,K} (K + 1.1 \cdot a)$. By Theorem 14, L is not 1-IRTA definable.*

Example 28. *Consider the language $L = \{(x \cdot a)(1 \cdot b) \mid 0 < x < 1\}$. This language is accepted by a 1-TA that reads a on a guard $0 < x < 1$, resets the clock x and reads b at $x = 1$. This is clearly not an IRTA. We will once again see that K-monotonicity does not hold for any K. For $u = (0.2 \cdot a)$, $c^K(u) + 1 \equiv^K c^K(u) + 1.1$ holds for every constant $K \in \mathbb{N}$. Since $u(1 \cdot b) \in L$ and $u(1.1 \cdot b) \notin L$, we have $u(1 \cdot b) \not\approx^{L,K} u(1.1 \cdot b)$. Thus, there is no K such that $\approx^{L,K}$ verifies K-monotonicity* (c)*. By Theorem 14, L is not definable by an 1-IRTA.*

Example 29. *Consider the language $L = \{(0 \cdot a)^n(0 \cdot b)^n \mid n \in \mathbb{N}\}$. There is no K such that $\approx^{L,K}$ has finite index, although $\approx^{L,K}$ is 0-monotonic: for a word u with $0 < c^0(u)$, no extension of u belongs to L and hence monotonicity* (c) *is trivially true; pick a word u with $c^0(u) = 0$, and let t, t' such that $c^0(u) + t \equiv^0 c^0(u) + t'$. When $t = 0$, we have $t' = 0$ and monotonicity is trivial. Suppose $0 < t, t'$. Then $u(t \cdot \alpha) \notin L$ and $u(t' \cdot \alpha) \notin L$ for any letter α. Once again,* (c) *holds.*

We now argue the infinite index, similar to the case of untimed languages. For distinct integers n and m we have $((0 \cdot a)^n)^{-1}L \neq ((0 \cdot a)^m)^{-1}L$. Since $\tau_{0,0}$ is the identity we have $\tau_{0,0}(((0 \cdot a)^n)^{-1}L) \neq ((0 \cdot a)^m)^{-1}L$. Thus, $(0 \cdot a)^n \not\approx^{L,K} (0 \cdot a)^m$.

Example 30. *Consider the language $M = \{u \in \mathbb{T}\Sigma^* \mid c^1(u) = 0\}$ with alphabet $\Sigma = \{a\}$ which has the following residual languages. For $u \in \mathbb{T}\Sigma^*$,*

$$u^{-1}M = M \quad \text{when } c^1(u) = 0, \quad u^{-1}M = \{v \in \mathbb{T}\Sigma^* \mid \sigma(uv) = 1\} \quad \text{when } 0 < c^1(u) < 1,$$
$$u^{-1}M = \emptyset \quad \text{when } 1 < c^1(u).$$

The equivalence classes for $\approx^{M,1}$ are the following:

$$[\epsilon]_{\approx^{M,1}} = M, \quad [(\tfrac{3}{2} \cdot a)]_{\approx^{M,1}} = \{u \in \mathbb{T}\Sigma^* \mid 1 < c^1(u)\}, \quad [(\tfrac{1}{2} \cdot a)]_{\approx^{M,1}} = \{u \in \mathbb{T}\Sigma^* \mid 0 < c^1(u) < 1\}.$$

The 1-TA $\mathcal{A}^M_{\approx^{M,1}}$ is depicted in Figure 8.1. In this automaton, $[\epsilon]_{\approx^{M,1}}$ serves as both the initial and final state, $[(\frac{3}{2} \cdot a)]_{\approx^{M,1}}$ is the sink state, while $[(\frac{1}{2} \cdot a)]_{\approx^{M,1}}$ is the rightmost state.

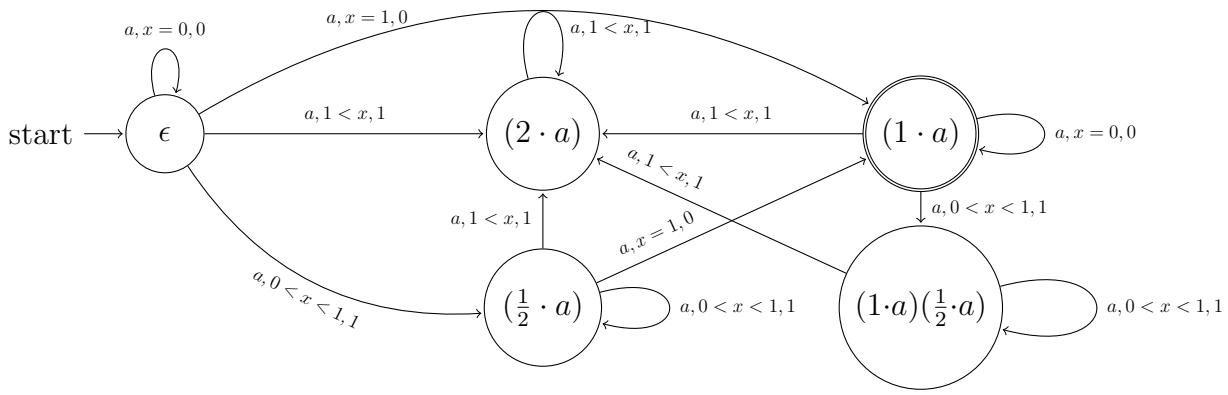


Figure 8.2: A strict 1-IRDTA $\mathcal{A}^{L_1}_{\approx_{L,1}}$ accepting $L_1 = \{u \in \mathbb{T}\Sigma^* \mid \sigma(u) = 1\}$.

Example 31. *Consider the language $L_1 = \{u \in \mathbb{T}\Sigma^* \mid \sigma(u) = 1\}$ given in Figure 8.2. Its residual languages are the following. For $u \in \mathbb{T}\Sigma^*$,*

$$u^{-1}L_1 = \emptyset \quad \text{when } 1 < \sigma(u), \quad u^{-1}L_1 = \{v \in \mathbb{T}\Sigma^* \mid \sigma(v) = 0\} \quad \text{when } \sigma(u) = 1,$$
$$u^{-1}L_1 = L_1 \quad \text{when } \sigma(u) = 0, \quad u^{-1}L_1 = \{v \in \mathbb{T}\Sigma^* \mid \sigma(u) + \sigma(v) = 1\} \quad \text{when } 0 < \sigma(u) < 1.$$

The strict 1-IRDTA $\mathcal{A}^{L_1}_{\approx_{L_1,1}}$ is depicted in Figure 8.2 and its equivalence classes for $\approx^{L_1,1}$ are:

$$[(1 \cdot a)]_{\approx_{L_1,1}} = L_1, \qquad [(1 \cdot a)(\tfrac{1}{2} \cdot a)]_{\approx_{L_1,1}} = \{u \in \mathbb{T}\Sigma^* \mid 0 < c^1(u) < 1 \wedge 1 < \sigma(u)\},$$
$$[\epsilon]_{\approx_{L_1,1}} = \{u \in \mathbb{T}\Sigma^* \mid \sigma(u) = 0\}, \qquad [(\tfrac{1}{2} \cdot a)]_{\approx_{L_1,1}} = \{u \in \mathbb{T}\Sigma^* \mid 0 < c^1(u) < 1 \wedge 0 < \sigma(u) < 1\},$$
$$[(2 \cdot a)]_{\approx_{L_1,1}} = \{u \in \mathbb{T}\Sigma^* \mid 1 < c^1(u)\}.$$

Example 32. *As a last example, consider $L = \{(t_1 \cdot a)(t_2 \cdot b) \mid t_1 = 0$ or $t_1 \geq 1$, and $t_2 \geq 0\}$. This language can be recognized by a 1-IRTA with greatest constant 1. Let $u = (0 \cdot a)$ and $v = (2 \cdot a)$. We would like to see the relation between these words for different constants $K = 1$ and $K = 2$. Notice that $c^2(u) = c^2(v) = 0$ and the residuals $u^{-1}L$ and $v^{-1}L$ are identical. Therefore, $u \approx^{L,2} v$. However, $c^1(u) = 0$, but $c^1(v) = 2$ and therefore $u \not\approx^{L,1} v$. Therefore, $\approx^{L,2}$ is not a refinement of $\approx^{L,1}$. The example indicates that if we have two constants K_1, K_2 such that $\approx^{L,K_1}$ and $\approx^{L,K_2}$ are both monotonic and have finite index, a smaller constant does not mean a coarser equivalence.*

8.4 Effectively Computing $\mathcal{A}^L_{\approx_{L,K}}$

In this section we show that given a 1-IRDTA with greatest constant less than or equal to K accepting L we can effectively compute $\mathcal{A}^L_{\approx_{L,K}}$. No matter the 1-IRDTA we start from, as long as it has greatest constant less than or equal to K, we get the same automaton $\mathcal{A}^L_{\approx_{L,K}}$. It is therefore a canonical automaton for L, for a given constant K. However, it is not necessarily the canonical automaton $\mathcal{A}^L_{\approx_{L,K_L}}$ of L defined in the previous section.

Our goal is to compute a finite set of representatives covering all the equivalence classes of $\approx^{L,K}$. The representatives we choose will have rational timestamps. We start with a

preliminary notion. We say that an equivalence $\approx \subseteq \mathbb{T}\Sigma^* \times \mathbb{T}\Sigma^*$ is *decidable* when given two timed words $u, v \in \mathbb{T}\Sigma^*$ with rational timestamps we can decide whether $u \approx v$.

Proposition 28. *If $\mathcal{A}$ is a 1-IRDTA with greatest constant less than or equal to K then $\approx^{L(\mathcal{A}),K}$ is decidable.*

Proof. We show that given $u, v \in \mathbb{T}\Sigma^*$ with rational timestamps, we can decide $u \approx^{L(\mathcal{A}),K} v$. Since the construction of the proof of Proposition 25 does not change the constant K, we can assume that $\mathcal{A}$ is a strict 1-IRDTA. Let $(q, c^K(u))$ and $(q', c^K(v))$ be the two configurations such that $(q_I, 0) \rightsquigarrow^u (q, c^K(u))$ and $(q_I, 0) \rightsquigarrow^v (q', c^K(v))$. By Lemma 10, $\tau_{u,v}(\mathcal{L}(q, c^K(u)) = \mathcal{L}(q, c^K(v))$. Thus, $u \approx^{L(\mathcal{A}),K} v$ if and only if $c^K(u) \equiv^K c^K(v)$ and $\mathcal{L}(q, c^K(v)) = \mathcal{L}(q', c^K(v))$. The first condition is easy. We focus on deciding $\mathcal{L}(q, c^K(v)) = \mathcal{L}(q', c^K(v))$. Checking equality of 1-IRTA is decidable, since they can be determinized. The extra difficulty here is that we want to look at words that start with clock value $c^K(v)$. Below, we reduce it to checking equality of IRTA languages, in the usual sense.

Add states $\overline{q}$ and $\overline{q}'$ in $\mathcal{A}$, and transitions $(\overline{q}, q, a, \phi_{[c^K(v)]_{\equiv^K}}, 1)$ and $(\overline{q}', q', a, \phi_{[c^K(v)]_{\equiv^K}}, 1)$. We claim that $\mathcal{L}(q, c^K(v)) = \mathcal{L}(q', c^K(v))$ iff $\mathcal{L}(\overline{q}, 0) = \mathcal{L}(\overline{q}', 0)$. It is straightforward to show that $\mathcal{L}(\overline{q}, 0) = \mathcal{L}(\overline{q}', 0) \implies \mathcal{L}(q, c^K(v)) = \mathcal{L}(q', c^K(v))$. Next, we show the converse.

Assume $\mathcal{L}(q, c^K(v)) = \mathcal{L}(q', c^K(v))$ and let $x = c^K(v)$. Any word $u \in \mathcal{L}(\overline{q}, 0)$ is of the form $u = (y \cdot a)w$ for some $y \equiv^K x$. Consider the bijection $\tau_{y,x}$. By property **2** for $\tau_{y,x}$ we have $(x \cdot a)\tau_{y,x}(w) \equiv^K (y \cdot a)w$. By definition of $\equiv^K$, a 1-IRTA with greatest constant at most K accepts $(x \cdot a)\tau_{y,x}(w)$ iff it accepts $(y \cdot a)w$. Therefore, we have $(x \cdot a)\tau_{y,x}(w) \in \mathcal{L}(\overline{q}, 0)$. Hence, $\tau_{y,x}(w) \in \mathcal{L}(q, x)$. The hypothesis $\mathcal{L}(q, x) = \mathcal{L}(q', x)$ implies $\tau_{y,x}(w) \in \mathcal{L}(q', x)$. Hence, $(x \cdot a)\tau_{y,x}(w) \in \mathcal{L}(\overline{q}', 0)$. Once again, since $(x \cdot a)\tau_{y,x}(w) \equiv^K (y \cdot a)w$, we get $(y \cdot a)w \in \mathcal{L}(\overline{q}', 0)$. □

Proposition 29. *If $\approx$ is K-monotonic, decidable and has finite index then we can effectively compute a set of representatives for $[\mathbb{T}\Sigma^*]_\approx$.*

Proof. Given a finite subset $Y \subseteq \mathbb{Q}_{\geq 0}$ of representatives for $[\mathbb{R}_{\geq 0}]_{\equiv^K}$, define the function $f \colon \wp(\mathbb{T}\Sigma^*) \to \wp(\mathbb{T}\Sigma^*)$ by

$$f(X) = \{w(t_{w,y} \cdot a) \in \mathbb{T}\Sigma^* \mid w \in X, a \in \Sigma, y \in Y, \exists t \in \mathbb{R}_{\geq 0}, c^K(w) + t \equiv^K y\} \cup \{\epsilon\} \ ,$$

where $t_{w,y} = \max(0, y - c^K(w))$.

Next we show that a set of representatives for $[\mathbb{T}\Sigma^*]_\approx$ is given by the iterate $f^n(\emptyset)$ with least n s.t. $[f^{n+1}(\emptyset)]_\approx = [f^n(\emptyset)]_\approx$ and that we can compute this iterate.

- Because $\approx$ has finite index there exists $n \in \mathbb{N}$ such that $[f^{n+1}(\emptyset)]_\approx = [f^n(\emptyset)]_\approx$. Given a finite set X of timed words with rational timestamps we can compute $f(X)$, thereby we can compute each iterate of f. For every $m \in \mathbb{N}$ we have $f^m(\emptyset) \subseteq f^{m+1}(\emptyset)$. Thus, to decide $[f^{m+1}(\emptyset)]_\approx = [f^m(\emptyset)]_\approx$ it suffices to check $[f^{m+1}(\emptyset)]_\approx \subseteq [f^m(\emptyset)]_\approx$. This inclusion translates to checking $\forall x \in f^{m+1}(\emptyset), \exists y \in f^m(\emptyset), x \approx y$. Since $f^n(\emptyset)$ is a finite set of rational timed words and $\approx$ is decidable, we can decide this last check, thus also $[f^{m+1}(\emptyset)]_\approx = [f^m(\emptyset)]_\approx$.

- Next, we show by induction on the length of the words, that $f^n(\emptyset)$ is a set of representatives for $[\mathbb{T}\Sigma^*]_\approx$ i.e., for every $u \in \mathbb{T}\Sigma^*$ there is $v \in f^n(\emptyset)$ such that $u \approx v$. It is true for ϵ since it belongs in $f^n(\emptyset)$. Let $w \in \mathbb{T}\Sigma^*$, $a \in \Sigma$ and $t \in \mathbb{R}_{\geq 0}$. As an induction hypothesis assume there is $w' \in f^n(\emptyset)$ such that $w \approx w'$. We now look at a one-step extension $w(t \cdot a)$. By K-monotonicity **(b)** there is $t' \in \mathbb{R}_{\geq 0}$ such that $w(t \cdot a) \approx w'(t' \cdot a)$. Since Y is a set of representatives for $[\mathbb{R}_{\geq 0}]_{\equiv^K}$, there is $y \in Y$ such that $c^K(w') + t' \equiv^K y$. It is an easy exercise to check that, since $c^K(w') + t' \equiv^K y$, we also have $c^K(w') + t_{w',y} \equiv^K y$. Hence, by K-monotonicity **(c)**, $w'(t' \cdot a) \approx w'(t_{w',y} \cdot a)$. Thus, by transitivity $w(t \cdot a) \approx w'(t_{w',y} \cdot a)$. □

If $\mathcal{C}$ is a 1-IRDTA with greatest constant less than or equal to K then $\approx^{L(\mathcal{C}),K}$ is K-monotonic, decidable and has finite index (Theorem 14 and Proposition 28). Thus, by Proposition 29 we can compute a set of representatives for $[\mathbb{T}\Sigma^*]_{\approx^{L(\mathcal{C}),K}}$. Completing the construction of $\mathcal{A}^{L(\mathcal{C})}_{\approx^{L(\mathcal{C}),K}}$ is straightforward, and we can readily compute the set of final states, as membership in $L(\mathcal{C})$ of timed word with rational timestamps is decidable.

8.5 Languages with No Resets

In this section we consider the subclass of 1-TA with transitions that never reset i.e., the class of 1-TA with transitions in $Q \times Q \times \Sigma \times \Phi \times \{1\}$. We denote this class by 1-NRTA. In this context, where the clock is never reset, any run on a word $u = (t_1 \cdot a_1) \dots (t_n \cdot a_n)$, results in the final clock value $\sigma(u) = t_1 + \dots + t_n$. A language $L \subseteq \mathbb{T}\Sigma^*$ *with no resets* is a language accepted by a 1-NRTA. The class of languages definable by 1-NRTAs is strictly included in that of 1-IRTA. Strict inclusion is shown by $M = \{u \in \mathbb{T}\Sigma^* \mid c^1(u) = 0\}$. The 1-TA in Figure 1.4 is a 1-NRTA. Next, we adapt the reasoning of Sections 8.2 and 8.3 to establish a canonical representation specific to this subclass. Essentially, to achieve this, we replace $c^K(u)$ by $\sigma(u)$ everywhere and prevent transitions of the canonical automaton from resetting.

8.5.1 1-NRTA from Equivalence on Timed Words

We replace $c^K(u)$ by $\sigma(u)$ in the definition of K-monotonicity.

Given a K-monotonic and L-preserving finite index equivalence $\approx \subseteq \mathbb{T}\Sigma^* \times \mathbb{T}\Sigma^*$ we define a transition $([u]_\approx, [v]_\approx, a, \phi_{[\sigma(v)]_{\equiv^K}}, 1) \in \overline{T}_\approx$ iff there is $t \in \mathbb{R}_{\geq 0}$ such that $u(t \cdot a) \approx v$.

Proposition 30. *Suppose $\approx$ is a K-monotonic, L-preserving finite index equivalence. Then $\mathcal{B}^L_\approx = ([\mathbb{T}\Sigma^*]_\approx, [\epsilon]_\approx, \overline{T}_\approx, F^L_\approx)$ is a 1-NRDTA and we have $L = L(\mathcal{B}^L_\approx)$.*

Proof. By K-monotonicity and L-preservation, $\mathcal{B}^L_\approx$ defines a 1-NRTA. A similar proof to the proof of Proposition 26 shows that for every timed word u, the set of configurations reached in $\mathcal{B}^L_\approx$ on reading u starting from the initial configuration $([\epsilon]_\approx, 0)$ is given by $\{([u]_\approx, \sigma(u))\}$. Thus, $\mathcal{B}^L_\approx$ is deterministic and accepts L by definition of F_L. □

8.5.2 A Myhill-Nerode Theorem for Languages with No Resets

Given L and K we define the syntactic equivalence $\asymp^{L,K} \subseteq \mathbb{T}\Sigma^* \times \mathbb{T}\Sigma^*$ for 1-NRTAs by $u \asymp^{L,K}$ iff $\sigma(u) \equiv^K \sigma(v)$ and $\tau_{\sigma(u),\sigma(v)}(u^{-1}L) = v^{-1}L$.

Theorem 15 (Counterpart of Theorem 14.)**.**

- *$L \subseteq \mathbb{T}\Sigma^*$ is a language definable by a 1-NRTA if and only if there is a constant $K \in \mathbb{N}$ such that $\asymp^{L,K}$ is K-monotonic and has finite index.*
- *$\asymp^{L,K}$ is coarser than any K-monotonic and L-preserving equivalence.*

Proof. We adapt the proof of Theorem 14, without the use of strict 1-IRTAs. □

We can easily adapt Proposition 28 and Proposition 29 to show that given a 1-NRDTA with greatest constant less than or equal to K accepting L, we can compute $\mathcal{B}^L_{\asymp^{L,K}}$.

Example 33. *Figure 8.3 depicts the 1-NRTA $\mathcal{B}^L_{\asymp^{L,1}}$ for $L = \{u \in \mathbb{T}\Sigma^* \mid \sigma(u) = 1\}$ and the equivalence classes of $\asymp^{L,1}$ are:*

$$[\epsilon]_{\asymp^{L,1}} = \{u \in \mathbb{T}\Sigma^* \mid \sigma(u) = 0\}, \qquad [(2 \cdot a)]_{\asymp^{L,1}} = \{u \in \mathbb{T}\Sigma^* \mid 1 < \sigma(u)\},$$
$$[(1 \cdot a)]_{\asymp^{L,1}} = L, \qquad [(\tfrac{1}{2} \cdot a)]_{\asymp^{L,1}} = \{u \in \mathbb{T}\Sigma^* \mid 0 < \sigma(u) < 1\}.$$

Example 32 showed that for IRTA, one cannot compare $\approx^{L,K_1}$ and $\approx^{L,K_2}$ even if both are monotonic. The situation is different in the NRTA case, as shown by the next lemma.

Lemma 12. *Assume $K \in \mathbb{N}$ such that $\asymp^{L,K}$ is K-monotonic and has finite index. Let $K < m$. Then $\asymp^{L,m} \subseteq \asymp^{L,K}$.*

Proof. Assume $u \asymp^{L,m} v$. Let τ^m be the "$\tau_{u,v}$" obtained using the constant m and τ^K be the one obtained with K. By hypothesis, we have (1) $\sigma(u) \equiv^m \sigma(v)$ and (2) $\tau^m(u^{-1}L) = v^{-1}L$. To show $u \asymp^{L,K} v$. From $\sigma(u) \equiv^m \sigma(v)$ we deduce $\sigma(u) \equiv^K \sigma(v)$ (because $\equiv^m \subseteq \equiv^K$). It remains to show $\tau^K(u^{-1}L) = v^{-1}L$.

For that we use a 1-NRDTA with greatest constant less than or equal to K accepting L which exists since $\asymp^{L,K}$ is K-monotonic and has finite index (take for example $\mathcal{B}^L_{\asymp^{L,K}}$). Let q be the state reached on u. Therefore, $u^{-1}L = \mathcal{L}(q, \sigma(u))$. By Lemma 10, $\tau^K(u^{-1}L) = \mathcal{L}(q, \sigma(v))$. Since $K < m$, we can apply Lemma 10 once again with the constant m, to get $\tau^m(u^{-1}L) = \mathcal{L}(q, \sigma(v))$. This shows $\tau^K(u^{-1}L) = \tau^m(u^{-1}L)$ and from (2), we deduce $\tau^K(u^{-1}L) = v^{-1}L$. □

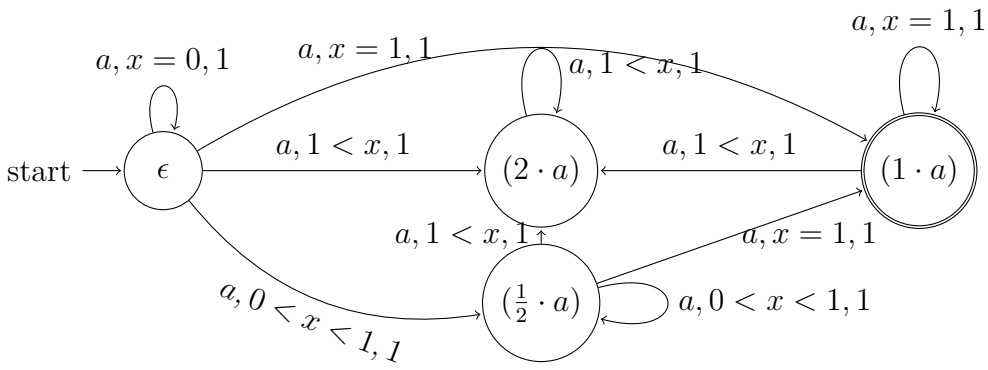


Figure 8.3: A 1-NRTA $\mathcal{B}^L_{\widetilde{<}_{L,1}}$ accepting $L = \{u \in \mathbb{T}\Sigma^* \mid \sigma(u) = 1\}$.

Chapter 9

CONCLUSIONS

In this thesis, we have introduced a unified approach for solving the inclusion problem, leveraging quasiorders to prune the search for counterexamples to inclusion. Additionally, we have established a Myhill-Nerode theorem for timed languages with integer resets.

9.1 A Uniform Approach for Inclusion Problems

We presented a uniform approach for solving the language inclusion problem $L^\omega(\mathcal{A}) \subseteq L^\omega(\mathcal{B})$ between languages of infinite words given by automata, for different decidable cases.

Our approach is conceptually simple: Using quasiorders on finite words we compute a finite subset S of ultimately periodic words of $L^\omega(\mathcal{A})$ that is sufficient for solving the inclusion problem. We compute S through least fixpoint computations for the languages of finite prefixes and periods of ultimately periodic words. The functions to iterate for these fixpoints are readily derived from the automaton $\mathcal{A}$ and the fixpoints converge in finitely many iterations thanks to the "well"-property of our quasiorders. Finally, we decide the inclusion by a straightforward membership check on the elements of S. Additionally, to guarantee that this approach is feasible the well-quasiorders need to be $L^\omega(\mathcal{B})$-preserving, a property ensuring that counterexamples to inclusion are preserved, and they need to verify some monotonicity conditions w.r.t. word concatenation ensuring computability of the subset S.

We demonstrated the generality of our approach by applying it to various decidable inclusion problems. Specifically, we defined algorithms for the inclusion problem between Büchi Automata, the inclusion of a Büchi Pushdown Automaton into a Büchi Automaton and, the inclusion between Büchi Visibly Pushdown Automata.

We categorize our algorithms into two types. The first type uses a pair of separate quasiorders, where the first quasiorder compares prefixes and the second one compares periods of ultimately periodic words. The second type is our FORQ-based algorithm using one quasiorder for the prefixes and a family of quasiorders for the periods, each of them depending on a distinct prefix.

9.1.1 Two-Quasiorders Algorithms

We defined algorithms parameterized by a pair of quasiorders to address the inclusion problem between Büchi Automata, the inclusion of a Büchi Pushdown Automaton into a Büchi Automaton, and the inclusion between Büchi Visibly Pushdown Automata.

We presented different pairs of quasiorders to be used in our algorithms, particularly focusing on state-based quasiorders. State-based quasiorders are defined from the automaton $\mathcal{B}$ and compare words based on the states the words connect in the automaton. These quasiorders are interesting as they enable algorithms that exclusively manipulate the states of the automata $\mathcal{A}$ and $\mathcal{B}$.

Our algorithms were implemented for the inclusion between Büchi Automata and the inclusion between Büchi Visibly Pushdown Automata. In both cases, we instantiated the algorithms with state-based pairs of quasiorders. Our implementations were competitive against state-of-the-art tools, thus showing the benefits of having separate quasiorders for prefixes and periods.

9.1.2 FORQ-based Algorithm

We further refined the two-quasiorders approach by replacing the quasiorder for the periods by a family of right quasiorders, paving the way for even more efficient inclusion algorithms. Families of right quasiorders, extend the notion of family of right congruences introduced in the nineties by Maler and Staiger [60, 61].

A significant difference of our FORQ-based inclusion algorithm compared to the two-quasiorders approach is the increased number of fixpoint computations that, counterintuitively, yield better scalability. Indeed our prototype, FORKLIFT, which implements the FORQ-based algorithm, scales up well on benchmarks taken from real applications in verification and theorem proving. Notably, it outperforms the implementation of our two-quasiorders algorithm on these benchmarks.

Given that the inclusion problem between Büchi automata is PSpace-complete, it is important to have different heuristics for solving it. The FORQ heuristic stands out as a promising approach and we expect the notion of FORQ to have impact beyond the inclusion problem, e.g., in learning [7] and complementation [57].

9.1.3 Future Work

There are various quasiorders that can be used in our algorithms, offering flexibility within the framework. For instance, a class of relations that yields good results in practice are the simulation relations [3] on the states of the automata. Parolini [70] defined several quasiorders enhanced with simulation relations on the states of $\mathcal{B}$ to instantiate our two-quasiorders algorithm for the inclusion between Büchi Automata. A future direction would be to also define FORQs enhanced with simulations and empirically evaluate the resulting algorithms.

Regarding the inclusion between ω-visibly pushdown languages, it would be interesting to study whether FORQs can be extended to this setting.

Finally, there are other decidable inclusion problems that we can consider, such as the inclusion of a context-free language into a superdeterministic context-free language [73].

9.2 A Myhill-Nerode Theorem for Timed Languages

We have presented a Myhill-Nerode style characterization for timed languages accepted by timed automata with integer resets. This characterization leads to the construction of a canonical form for the class.

The three main ingredients of our characterization are: (1) the unique value $c^K(u)$ attained by the clock after reading a word u in any strict 1-IRTA of greatest constant K (Definition 8.1.1), (2) the K-monotonicity conditions (Definition 8.2.1) which enable building a strict 1-IRTA from an equivalence on timed words and (3) the final equivalence $\approx^{L,K}$ (Definition 8.3.1) which in turn uses a rescaling function. We have shown that a timed language is accepted by an IRTA iff there is a K for which $\approx^{L,K}$ is K-monotonic and has finite index. From the equivalence $\approx^{L,K}$ we can define a strict 1-IRTA $\mathcal{A}^L_{\approx^{L,K}}$ that can be effectively computed starting from an IRTA with maximum constant at most K. We have also adapted these ingredients to the class of timed automata with no resets.

9.2.1 Future Work

At the moment, there is no learning algorithm that can generate an IRTA for systems that are known to satisfy the integer reset assumption. We hope that our study of a Nerode-style equivalence for this class lays the foundation for learning algorithms.

Another direction for future work is to study more in depth how the equivalences $\approx^{L,K}$ for different constants K compare and to determine the smallest K for which the right-hand-side of the characterization works.

9.3 Quasiorders in Action

In conclusion, this thesis has demonstrated that quasiorders are both of theoretical and practical interest.

Indeed, we solved several inclusion problems thanks to quasiorders. Compared to the classical approach that complements, intersects and checks for emptiness, quasiorders are more versatile as they can be applied to solve inclusion problems even when complementation is not possible [69]. They are at the core of all algorithms presented in this thesis and, as demonstrated by our empirical evaluation, they are impactful in practice.

Moreover, using strengthened quasiorders (equivalence relations), we presented a computable canonical form for a subclass of regular timed languages. This result contributes to our understanding of timed languages. It also holds potential applications such as learning algorithms and even language inlcusion problems.

BIBLIOGRAPHY

[1] Parosh Aziz Abdulla et al. "Simulation Subsumption in Ramsey-Based Büchi Automata Universality and Inclusion Testing". In: *CAV'10: Proc. 20th Int. Conf. on Computer Aided Verification*. 2010, pp. 132–147.

[2] Parosh Aziz Abdulla et al. "When Simulation Meets Antichains". In: *TACAS'10: Proc. 16th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*. 2010, pp. 158–174.

[3] Parosh Aziz Abdulla et al. "Advanced Ramsey-Based Büchi Automata Inclusion Testing". In: *CONCUR'11: Proc. 22nd Int. Conf. on Concurrency Theory*. 2011, pp. 187–202.

[4] Rajeev Alur and David L. Dill. "A Theory of Timed Automata". In: *Theoretical Computer Science* 126.2 (1994), pp. 183–235. DOI: 10.1016/0304-3975(94)90010-8.

[5] Rajeev Alur and P. Madhusudan. "Visibly Pushdown Languages". In: *STOC'04: Proc. 36th Ann. ACM Symp. on Theory of Computing*. ACM, 2004, pp. 202–211.

[6] Jie An et al. "Learning Nondeterministic Real-Time Automata". In: *ACM Transactions on Embedded Computing Systems* 20.5s (2021), pp. 1–26. DOI: 10.1145/3477030.

[7] Dana Angluin, Udi Boker, and Dana Fisman. "Families of DFAs as Acceptors of ω-Regular Languages". In: *Logical Methods in Computer Science* 14 (2018), 11:1–11:14.

[8] Christel Baier et al. "When Are Timed Automata Determinizable?" In: *International Colloquium on Automata, Languages and Programming*. 2009, pp. 43–54.

[9] *BAIT: An ω-regular language inclusion checker*. https://github.com/parof/bait. Accessed: 2022-01-17.

[10] Devendra Bhave and Shibashis Guha. "Adding Dense-Timed Stack to Integer Reset Timed Automata". In: *RP'17: Proc. 11th International Conference on Reachability Problems*. Vol. 10506. LNCS. Springer, 2017, pp. 9–25. DOI: 10.1007/978-3-319-67089-8_2.

[11] Mikołaj Bojańczyk and Sławomir Lasota. "A Machine-Independent Characterization of Timed Languages". In: *ICALP'12: Proc. of the 39th Int. Colloquium of Automata, Languages and Programming*. Vol. 7392. Springer, 2012, pp. 92–103. DOI: 10.1007/978-3-642-31585-5_12.

[12] Filippo Bonchi and Damien Pous. "Checking NFA Equivalence with Bisimulations up to Congruence". In: *Proc. of the 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*. ACM, 2013, 457–468.

[13] Ahmed Bouajjani et al. "Antichain-Based Universality and Inclusion Testing over Nondeterministic Finite Tree Automata". In: *CIAA'08: Proc. Int. Conf. on Implementation*

and Applications of Automata. LNCS. Springer, 2008, pp. 57–67. DOI: 10.1007/978-3-540-70844-5_7.

[14] Patricia Bouyer et al. "Updatable timed automata". In: *Theoretical Computer Science* 321.2-3 (2004), pp. 291–345. DOI: 10.1016/j.tcs.2004.04.003.

[15] Véronique Bruyère, Marc Ducobu, and Olivier Gauwin. "Visibly Pushdown Automata: Universality and Inclusion via Antichains". In: *LATA'13: Proc. Int. Conf. on Language and Automata Theory and Applications*. LNCS. Springer, 2013. DOI: 10.1007/978-3-642-37064-9_18.

[16] S. Tripakis C. Daws A. Olivero and S. Yovine. "The tool KRONOS". In: *In Proc. Hybrid Systems III: Verification and Control (1995), volume 1066 of LNCS*, pp. 208–219.

[17] Hugues Calbrix, Maurice Nivat, and Andreas Podelski. "Ultimately Periodic Words of Rational ω-Languages". In: *Proc. Int. Symp. on Mathematical Foundations of Programming Semantics (MFPS)*. LNCS. Springer, 1994, pp. 554–566.

[18] Lorenzo Clemente and Richard Mayr. "Efficient reduction of nondeterministic automata with application to language inclusion testing". In: *Log. Methods Comput. Sci.* 15.1 (2019). DOI: 10.23638/LMCS-15(1:12)2019.

[19] Rina S. Cohen and Arie Y. Gold. "Theory of ω-languages I: Characterizations of ω-context-free languages". In: *Journal of Computer and System Sciences* 15.2 (1977), pp. 169–184. DOI: 10.1016/S0022-0000(77)80004-4.

[20] Patrick Cousot and Radhia Cousot. "Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints". In: Jan. 1977, pp. 238–252. DOI: 10.1145/512950.512973.

[21] Patrick Cousot and Radhia Cousot. "Systematic Design of Program Analysis Frameworks." In: Jan. 1979, pp. 269–282. DOI: 10.1145/567752.567778.

[22] Stefano Crespi Reghizzi and Dino Mandrioli. "Operator Precedence and the Visibly Pushdown Property". In: *Journal of Computer and System Sciences* 78.6 (2012), pp. 1837–1867. DOI: 10.1016/j.jcss.2011.12.006.

[23] Aldo de Luca and Stefano Varricchio. "Well Quasi-Orders and Regular Languages". In: *Acta Informatica* 31.6 (1994), pp. 539–557. DOI: 10.1007/BF01213206.

[24] M. De Wulf et al. "Antichains: A New Algorithm for Checking Universality of Finite Automata". In: *CAV'06: Proc. 16th Int. Conf. on Computer Aided Verification*. Springer, 2006, pp. 17–30.

[25] K. Doveri et al. *Büchi Automata benchmarks for language inclusion*. https://github.com/parof/buchi-automata-benchmark. 2021.

[26] Kyveli Doveri, Pierre Ganty, and Luka Hadzi-Djokic. "Antichains Algorithms for the Inclusion Problem Between omega-VPL". In: *Tools and Algorithms for the Construction and Analysis of Systems*. Ed. by Sriram Sankaranarayanan and Natasha Sharygina. Cham: Springer Nature Switzerland, 2023, pp. 290–307.

[27] Kyveli Doveri, Pierre Ganty, and Luka Hadzi-Djokic. *omegaVPLinc v1.1*. Version v1.1. Jan. 2023. DOI: 10.5281/zenodo.7506895.

[28] Kyveli Doveri, Pierre Ganty, and Nicolas Mazzocchi. "FORQ-Based Language Inclusion Formal Testing". In: *CAV'22: Proc. 32nd Int. Conf. on Computer Aided Verification*. Springer, 2022, 109–129.

[29] Kyveli Doveri et al. “Inclusion Testing of Büchi Automata Based on Well-Quasiorders”. In: *CONCUR’21: Proc. 32nd Int. Conf. on Concurrency Theory*. LIPIcs. Schloss Dagstuhl, 2021, 3:1–3:22.

[30] Laurent Doyen and Jean-François Raskin. “Antichains for the Automata-Based Approach to Model-Checking”. In: *Logical Methods in Computer Science* 5.1 (2009). Ed. by Orna Grumberg. DOI: 10.2168/lmcs-5(1:5)2009.

[31] Laurent Doyen and Jean-François Raskin. “Antichain Algorithms for Finite Automata”. In: *Tools and Algorithms for the Construction and Analysis of Systems*. LNCS. Springer, 2010, pp. 2–22.

[32] Alexandre Duret-Lutz et al. “From Spot 2.0 to Spot 2.10: What’s New?” In: *CAV: 34th International Conference on Computer Aided Verification*. LNCS, pp. 174–187.

[33] Alexandre Duret-Lutz et al. “Spot 2.0 - A Framework for LTL and ω-Automata Manipulation”. In: *ATVA: 14th International Conference on Automated Technology for Verification and Analysis*. LNCS. 2016. DOI: 10.1007/978-3-319-46520-3_8.

[34] Javier Esparza. “Automata theory – An algorithmic approach.” In: (2017).

[35] Robert W. Floyd. “Syntactic Analysis and Operator Precedence”. In: *J. ACM* 10.3 (1963), 316–333. DOI: 10.1145/321172.321179.

[36] Seth Fogarty and Moshe Y. Vardi. “Efficient Büchi Universality Checking”. In: *TACAS’10: Proc. 16th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*. LNCS. Springer, 2010, pp. 205–220.

[37] Oliver Friedmann, Felix Klaedtke, and Martin Lange. “Ramsey Goes Visibly Pushdown”. In: *ICALP’13: Proc. 40th Int. Coll. on Automata, Languages, and Programming*. LNCS. Springer, 2013, pp. 224–237. DOI: 10.1007/978-3-642-39212-2_22.

[38] Oliver Friedmann, Felix Klaedtke, and Martin Lange. “Ramsey-Based Inclusion Checking for Visibly Pushdown Automata”. In: *ACM Transactions on Computational Logic* 16.4 (2015), pp. 1–24. DOI: 10.1145/2774221.

[39] Pierre Ganty, Francesco Ranzato, and Pedro Valero. “Language Inclusion Algorithms as Complete Abstract Interpretations”. In: *Static Analysis*. Springer, 2019, pp. 140–161.

[40] Pierre Ganty, Francesco Ranzato, and Pedro Valero. “Complete Abstractions for Checking Language Inclusion”. In: *ACM Trans. Comput. Logic* 22.4 (2021), pp. 1–40.

[41] *GOAL: Graphical Tool for Omega-Automata and Logics*. http://goal.im.ntu.edu.tw/wiki/doku.php. Accessed: 2022-01-17.

[42] Olga Grinchtein, Bengt Jonsson, and Martin Leucker. “Learning of Event-Recording Automata”. In: *Theoretical Computer Science* 411.47 (2010), pp. 4029–4054. DOI: 10.1016/j.tcs.2010.07.008.

[43] Matthias Heizmann, Jochen Hoenicke, and Andreas Podelski. “Software Model Checking for People Who Love Automata”. In: *CAV: 25th International Conference on Computer Aided Verification*. LNCS. 2013, 36–52. DOI: 10.1007/978-3-642-39799-8_2.

[44] Matthias Heizmann et al. “Ultimate Automizer and the Search for Perfect Interpolants - (Competition Contribution)”. In: *TACAS’18: Proc. 24th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*. LNCS. Springer, 2018. DOI: 10.1007/978-3-319-89963-3_30.

[45] Thomas A. Henzinger et al. “Regular Methods for Operator Precedence Languages”. In: *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*. Ed. by Kousha Etessami, Uriel Feige, and Gabriele Puppis. Vol. 261. Leibniz

International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 129:1–129:20.

[46] Philipp Hieronymi et al. "Decidability for Sturmian Words". In: *CSL: 30th EACSL Annual Conference on Computer Science Logic*. LIPIcs. 2022, 24:1–24:23. DOI: 10.4230/LIPIcs.CSL.2022.24.

[47] Martin Hofmann and Wei Chen. "Abstract Interpretation from Büchi Automata". In: *Proc. of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. ACM Press, 2014.

[48] John E. Hopcroft and Jeff D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing Company, 1979.

[49] P. Pettersson K. G. Larsen and W. Yi. "UPPAAL in a nutshell". In: *Journal of Software Tools for Technology Transfer*. 1997, pp. 134–152.

[50] Takumi Kasai and Shigeki Iwata. *Some Problems in Formal Language Theory Known as Decidable are Proved EXPTIME Complete*. 1992.

[51] Igor Konnov. "Handbook of Model Checking by Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem (eds), published by Springer International Publishing AG, Cham, Switzerland, 2018." In: (Jan. 2019).

[52] Denis Kuperberg, Laureline Pinault, and Damien Pous. "Coinductive Algorithms for Büchi Automata". In: *Fundam. Informaticae* 180.4 (2021), pp. 206–220. DOI: 10.3233/FI-2021-2046.

[53] Orna Kupferman. "Automata Theory and Model Checking". In: *Handbook of Model Checking*. Ed. by Edmund M. Clarke et al. Cham: Springer International Publishing, 2018, pp. 107–151. DOI: 10.1007/978-3-319-10575-8_4.

[54] Orna Kupferman and Moshe Y. Vardi. "Verification of Fair Transition Systems". In: *CAV: International Conference on Computer Aided Verification*. LNCS. 1996, pp. 372–382. DOI: 10.1007/3-540-61474-5_84.

[55] Yong Li et al. "ROLL 1.0: ω-Regular Language Learning Library". In: *TACAS: 25th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. LNCS. 2019, pp. 365–371. DOI: 10.1007/978-3-030-17462-0_23.

[56] Yong Li et al. "A novel learning algorithm for Büchi automata based on family of DFAs and classification trees". In: *Information and Computation* (2020), pp. 208–226. DOI: 10.1016/j.ic.2020.104678.

[57] Yong Li et al. "Congruence Relations for Büchi Automata". In: *FM: Formal Methods*. LNCS. 2021, pp. 465–482.

[58] Oded Maler and Amir Pnueli. "On Recognizable Timed Languages". In: *FoSSaCS'04: Proc. of the Int. Conf. on Foundations of Software Science and Computation Structures*. Vol. 2987. LNCS. Springer, 2004, pp. 348–362. DOI: 10.1007/978-3-540-24727-2_25.

[59] Oded Maler and Ludwig Staiger. "On Syntactic Congruences for ω-Languages". In: *STACS: 10th Annual Symposium on Theoretical Aspects of Computer Science*. LNCS. 1993, pp. 93–112. DOI: 10.1007/3-540-56503-5_58.

[60] Oded Maler and Ludwig Staiger. "On Syntactic Congruences for ω-Languages". In: *Theor. Comput. Sci.* 183.1 (1997). DOI: 10.1016/S0304-3975(96)00312-X.

[61] Oded Maler and Ludwig Staiger. *On syntactic congruences for ω-languages*. Tech. rep. Verimag, France, 2008.

[62] Lakshmi Manasa and Krishna S. *Integer Reset Timed Automata: Clock Reduction and Determinizability*. 2010. eprint: 1001.1215 (cs).

[63] Roland Meyer, Sebastian Muskalla, and Elisabeth Neumann. *Liveness Verification and Synthesis: New Algorithms for Recursive Programs*. 2017. arXiv: 1701.02947 [cs.FL].

[64] Roland Meyer, Sebastian Muskalla, and Elisabeth Neumann. "Liveness Verification and Synthesis: New Algorithms for Recursive Programs". In: *CoRR* abs/1701.02947 (2017). arXiv: 1701.02947.

[65] M. Michel. *Complementation Is More Difficult with Automata on Infinite Words*. Tech. rep. Paris: CNET, 1988.

[66] Piterman Nir. "From Nondeterministic Büchi and Streett Automata to Deterministic Parity Automata". In: *Logical Methods in Computer Science* 3.3 (2007), pp. 1–21. DOI: 10.2168/lmcs-3(3:5)2007.

[67] Reed Oei et al. *Pecan: An Automated Theorem Prover for Automatic Sequences using Büchi Automata*. 2021.

[68] Reed Oei et al. "Pecan: An Automated Theorem Prover for Automatic Sequences using Büchi Automata". In: *CoRR* abs/2102.01727 (2021).

[69] J. Ouaknine and J. Worrell. "On the language inclusion problem for timed automata: closing a decidability gap". In: *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, 2004.* 2004, pp. 54–63. DOI: 10.1109/LICS.2004.1319600.

[70] Francesco Parolini. "Simulation-based Inclusion Checking Algorithms for omega-Languages". In: *Master Thesis, Università degli Studi di Padova*. 2020.

[71] *RABIT/Reduce: Tools for language inclusion testing and reduction of nondeterministic Büchi automata and NFA*. http://www.languageinclusion.org/doku.php?id=tools. Accessed: 2022-01-17.

[72] *ROLL library: Regular Omega Language Learning library*. https://github.com/ISCAS-PMC/roll-library. Accessed: 2022-01-17.

[73] E. P. Freidman S. A. Greibach and. "Superdeterministic PDAs: A subcase with a decidable inclusion problem". In: *Journal of the ACM* 27.4 (1980), pp. 675–700.

[74] *Spot: a platform for LTL and ω-automata manipulation*. https://spot.lrde.epita.fr/. Accessed: 2022-01-17.

[75] Jan Springintveld and Frits W. Vaandrager. "Minimizable Timed Automata". In: *FTRTFT'96: Proc. of 4th Int. Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*. Vol. 1135. LNCS. Springer, 1996, pp. 130–147. DOI: 10.1007/3-540-61648-9_38.

[76] P. Vijay Suman et al. "Timed Automata with Integer Resets: Language Inclusion and Expressiveness". In: *FORMATS'08: Proc. of the Int. Conf. on Formal Modeling and Analysis of Timed Systems*. Vol. 5215. Springer, 2008, pp. 78–92. DOI: 10.1007/978-3-540-85778-5_7.

[77] Ming-Hsien Tsai, Yih-Kuen Tsay, and Yu-Shiang Hwang. "GOAL for Games, Omega-Automata, and Logics". In: *CAV: 25th International Conference on Computer Aided Verification*. LNCS. 2013, pp. 883–889. DOI: 10.1007/978-3-642-39799-8_62.

[78] Ming-Hsien Tsai et al. "State of Büchi Complementation". In: *Logical Methods in Computer Science* 10.4 (2014). Ed. by Pierre Wolper, pp. 261–271.

[79] Pedro Valero. "On the Use of Quasiorders in Formal Language Theory". In: *Ph.D Thesis, Universidad Politécnica de Madrid*. 2020.

[80] Masaki Waga. "Active Learning of Deterministic Timed Automata with Myhill-Nerode Style Characterization". In: *CAV'23: Proc. of the 35th Int. Conf. on Computer Aided Verification*. Vol. 13964. LNCS. Springer, 2023, pp. 3–26. DOI: 10.1007/978-3-031-37706-8_1.

[81] William M. Waite and Gerhard Goos. "Properties of Programming Languages". In: *Compiler Construction*. New York, NY: Springer New York, 1984, pp. 15–45. DOI: 10.1007/978-1-4612-5192-7_2.