Università degli Studi di Padova

Department of Mathematics "Tullio Levi-Civita"

Master of Science in Computer Science

Simulation-based Inclusion Checking Algorithms for ω-Languages

Master Thesis

Supervisor

Prof. Francesco Ranzato, Università degli Studi di Padova

Co-supervisor

Prof. Pierre Ganty, IMDEA Software Institute, Madrid, Spain

Candidate

Francesco Parolini

Academic Year 2019-2020

Ai miei genitori.

"You keep on learning and learning, and pretty soon you learn something no one has learned before."

— Richard Feynman

Acknowledgements

Firstly, I have to thank, in Italian, the people who made this work possible.

Per primo vorrei ringraziare il relatore della mia Tesi, Prof. Francesco Ranzato, che non solo ha pazientemente seguito lo sviluppo di questo elaborato, ma che mi ha anche trasmesso la passione per l'Abstract Interpretation, argomento centrale del progetto di Dottorato che inizieró a breve. Ringrazio inoltre i membri del team di Madrid, Prof. Pierre Ganty e Kyveli Doveri, senza la loro conoscenza e i loro preziosi consigli questo lavoro non avrebbe mai visto la luce. Grazie ancora a tutto il "team azzurro" che ha reso la mia prima esperienzea di ricerca non solo formativa, ma anche divertente.

Desidero ringraziare anche le figure che durante gli anni hanno acceso in me la passione per l'Informatica, Prof. Gilberto Filé, Prof. Massimiliano Masetti e Gregorio Piccoli. Senza chi mi mostrasse la bellezza non solo dell'Informatica, ma anche del piacere di scoprire, non starei scrivendo questa Tesi.

Inoltre, non potrei tralasciare i miei amici, in particolare Alessandro, Andrea, Dario, Dennis, Maria, Marta, Matteo, Paolo e Valerio con i quali ho condiviso momenti felici, e che mi hanno aiutato a superare quelli difficili.

Devo poi ringraziare Silvia, con la quale sono stato lontano ma vicino *e che é stata una preziosa compagna di avventure alla scoperta dell'Europa.*

Grazie a mio zio Giorgio che é da sempre una figura speciale e che ha il dono di avere ogni volta il consiglio giusto per me.

Come ultimi, ma per i quali le parole ancora una volta non bastano, ringrazio i miei genitori per aver sempre creduto in me. Con tutti i vostri sacrifici avere reso possibile realizzare i miei sogni. A voi dedico questa Tesi.

Desenzano del Garda, Luglio 2020

Francesco Parolini

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The *language inclusion problem* is a fundamental and classical problem which consists in deciding, given two languages L_1 and L_2, whether $L_1 \subseteq L_2$. This problem has a wide variety of applications, ranging from *automata-based verification* [Kup18], reasoning for *logical theories* [Esp17] to *type systems* [HC14]. Whether the language inclusion problem is computable or not, depends on the nature of the languages and, also if it turns out to be computable, it is usually computationally intensive.

We will formally see that a language is ultimately a set of *words*, that are just concatenated symbols from one *alphabet* Σ. In general, languages are *not finite*, so that it is not possible to simply compute all the words in two languages and compare them. A vast assortment of techniques and algorithms have been proposed in the past to solve the problem for certain classes of languages [GRV19; OW04; Abd+11]. In particular, we are interested in the language inclusion problem between *ω-regular languages*, that are languages of *infinite words* recognized by *Büchi automata* [CG77]. We consider the framework put forward in [DG20], that relies on *abstract interpretation* [CC77] techniques in order to solve the language inclusion problem.

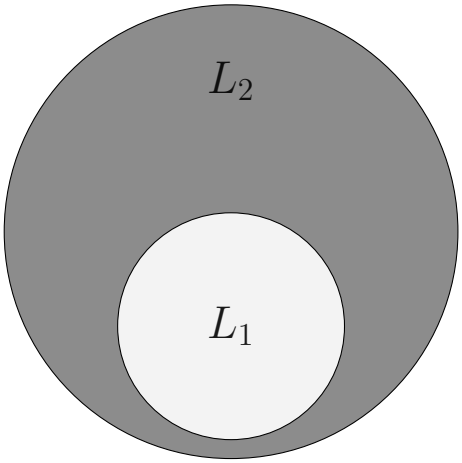

Figure 1.1: Representation of two languages L_1, L_2 such that $L_1 \subseteq L_2$

The theory of Abstract Interpretation, introduced in [CC77], is a general theory of the approximation of formal program semantics. It is an invaluable tool to prove

the correctness of a static analysis, as it makes it possible to express mathematically the link between the output of a practical, approximate analysis, and the original, uncomputable program semantics [Min17]. Although the most well-known application of abstract interpretation theory is *program verification* [Cou+05], during the years it has been exploited in many different fields: from efficient algorithms to compute the simulation equivalence [RT07], to artificial intelligence [RZ19]. Most of the time, the abstractions are *sound* but not *complete*, giving up precision in order to gain decidability. As we will see, the framework described in [DG20] is *sound and complete*, meaning that exactly solves the language inclusion problem between ω-regular languages, while not raising *false alarms*, as they are known in abstract interpretation terminology.

In particular, the idea is to *abstract* the language L_1 with an *over approximation* function ρ. As long as ρ satisfies a *completeness condition*, that intuitively corresponds to not losing precision while abstracting the language, we show how to use ρ in order to checking the inclusion between two languages.

The framework to solve the inclusion between ω-regular languages described in [DG20] is parameterized by a pair of *quasiorders* $\leq_1, \leq_2$ on words: as long as $\leq_1, \leq_2$ satisfy a list of requirements related to computability and completeness, they can be plugged into the framework in order to check the inclusion between two languages. As one could imagine, the performance of the algorithm described in [DG20] depends on the choice of the two quasiorders. In particular, *coarser* relations lead to algorithms that converge in less iterations. They also suggest two families of quasiorders that are suitable for the described framework: the *syntactic* and the *state-based* quasiorders. Our ultimate goal is to put forward a number of relations on words that are based on the *simulation relation* on the states of an automaton. We explore an ample mix of simulation relations, from the well-known *direct simulation* [DHWT91], up to less celebrated simulations, as the *k-lookahead simulation* [CM17]. We will see that the *simulation-based quasiorders on words* result in *coarser* relations than the state-based quasiorders, while being *finer* than the syntactic ones, effectively lying in the middle between the already proposed relations.

In Chapter 2 we formally define the concepts that will be needed in the rest of our work. We also describe the framework put forward in [DG20] on which we rely in order to check the inclusion between ω-regular languages; in Chapter 3 we define various simulation-based quasiorders on words and we prove that they meet the requirements of the framework, and, finally, in Chapter 4 we give several examples that show the practical advantages of using the simulation-based quasiorders.

Chapter 2

Background

In this chapter we formally describe the concepts and the background information needed to understand what follows. In Section 2.1 we give some basic mathematical background; in Section 2.2 we introduce the concept of *formal language* and we characterize some classes of languages in which we are interested in; in Section 2.3 we present basic notions of *order theory*; in Section 2.4 we define a number of *simulation preorders* on the states of an automaton and, finally, in Section 2.5 we describe how to solve the language inclusion problem for certain classes of languages using *complete abstractions*.

2.1 Mathematical background

Let X and Y be two sets. We denote by $|X|$ the *cardinality* of X and by $\wp(X)$ its *powerset*. We define $\wp_f(X) \triangleq \{S \in \wp(X) \mid |S| < \infty\}$. If X is a subset of some universe set U then X^C denotes the complement of X with respect to U when U is implicitly given by the context. A *partition* P of X is a set of non empty subsets of X, called *blocks*, that are pairwise disjoint and whose union gives X. If $f : X \to Y$ is a function between sets and $S \in \wp(X)$ then $f(S) \triangleq \{f(x) \mid x \in S\}$ denotes its image on a subset S. A composition of two functions f and g is denoted both by fg and $f \circ g$. We define $id : X \to X$ as the *identity function*, namely $id(x) \triangleq x$. Let $f : X \to X$ be a function. For all $n \in \mathbb{N}$ we inductively define:

$$f^n \triangleq \begin{cases} id & \text{if } n = 0 \\ f \circ f^{n-1} & \text{if } n > 0 \end{cases}$$

One *relation* $\mathscr{R}$ over X is a subset of $X \times X$. The *composition* of two relations $\mathscr{R}_1, \mathscr{R}_2$ is $\mathscr{R}_1 \circ \mathscr{R}_2 \triangleq \{(x, z) \mid \exists (x, y) \in \mathscr{R}_1 \ \wedge \ \exists (y, z) \in \mathscr{R}_2\}$. Let $\mathscr{R}$ be a relation on X.

- $\mathscr{R}$ is *reflexive* if for all $x \in X, x \mathscr{R} x$;
- $\mathscr{R}$ is *transitive* if for all $x, y, z \in X, x \mathscr{R} y \ \wedge \ y \mathscr{R} z \implies x \mathscr{R} z$;
- $\mathscr{R}$ is *symmetric* if for all $x, y \in X, x \mathscr{R} y \iff y \mathscr{R} x$;
- $\mathscr{R}$ is *antisymmetric* if for all $x, y \in X, x \mathscr{R} y \ \wedge \ y \mathscr{R} x \implies x = y$.

If $\mathscr{R}$ is reflexive, transitive and symmetric we say that $\mathscr{R}$ is an *equivalence*. If $\mathscr{R}$ is reflexive, transitive and antisymmetric we say that $\mathscr{R}$ is a *partial order*. If $\mathscr{R}$ is reflexive and transitive we say that $\mathscr{R}$ is a *quasiorder*, and we will recall this concept in Section 2.3. We define $[x]_{\mathscr{R}} \triangleq \{y \in X \mid x\mathscr{R}y \wedge y\mathscr{R}x\}$. The *transitive closure* of one relation $\mathscr{R}$ on one set X is defined as the smallest relation on X that contains $\mathscr{R}$ and is transitive. Let $\mathscr{R}_1, \mathscr{R}_2$ be two relations. If $\mathscr{R}_1 \subseteq \mathscr{R}_2$ we say that $\mathscr{R}_1$ is *finer* than $\mathscr{R}_2$ and if $\mathscr{R}_2 \subseteq \mathscr{R}_1$ we say that $\mathscr{R}_1$ is *coarser* than $\mathscr{R}_2$.

Let $k \in \mathbb{N}$ and $x_1, \ldots, x_k \in X$. We denote by $\vec{x}$ the k-dimensional vector $\langle x_i \rangle_{i \in [1,k]} \in X^k$. We denote by $(\vec{x})_i$ the element x_i. In what follows we abuse notations by implicitly lifting them to vectors. For instance, $\vec{x} \leq \vec{y} \overset{\triangle}{\iff} \forall i \in [1,k], x_i \leq y_i$. Sometimes we also implicitly lift the empty set $\emptyset$ to a vector, writing $\emptyset$ to refer to $\vec{\emptyset} \triangleq \langle \emptyset \rangle_{i \in [1,k]} \in \wp(X)^k$.

2.2 Formal languages

In mathematics, computer science, and linguistics, a *formal language* consists of words whose letters are taken from an alphabet Σ and are well-formed according to a specific set of rules. The alphabet Σ of a formal language consist of symbols that concatenate into strings of the language. Each string concatenated from symbols of this alphabet is called a *word*, and the words that belong to a particular formal language are sometimes called well-formed words or well-formed formulas. A formal language is often defined by means of a formal grammar, such as a regular grammar or context-free grammar, which consists of its formation rules. We now describe these concepts more formally.

Let Σ be a finite set of symbols. A finite sequence of elements of Σ is called a *finite word*. We denote the sequence $(a_1, a_2, \ldots, a_n)$ by mere juxtaposition:

$$a_1 a_2 \ldots a_n$$

The set of words is endowed with the operation of *concatenation product*, which associates two words $u = a_1 a_2 \ldots a_n$ and $v = b_1 b_2 \ldots b_m$ the word $uv = a_1 a_2 \ldots a_n b_1 b_2 \ldots b_m$. We denote with ϵ the *empty word*. We denote by Σ^* the set of words on Σ and by Σ^+ the set of nonempty words, that is $\Sigma^+ \triangleq \Sigma^* \setminus \{\epsilon\}$. An *infinite word* on the alphabet Σ is an infinite sequence of elements of Σ, which we also denote by juxtaposition:

$$a_1 a_2 \ldots a_n \ldots$$

We denote with Σ^ω the set of infinite words over the alphabet Σ. One *formal language*, or simply a *language*, is a subset of Σ^* or Σ^ω. Let L be a language, $w \in \Sigma^*$, we define the *context* of w as $ctx_L(w) \triangleq \{(x, y) \in \Sigma^* \times \Sigma^* \mid xwy \in L\}$.

In the following sections we describe three classes of languages: *regular*, *context-free* and *ω-regular*.

2.2.1 Regular languages and finite automata

There are many different ways to defines regular languages. A *regular language* is a formal language that can be expressed using a *regular expression* [HMU13]. The words of a regular language are sequences of finite length of symbols in the alphabet Σ . An alternative definition of regular language is that a regular language is the language accepted by *finite automaton* (FA). A FA can either be *deterministic* or *nondeterministic*.

A *deterministic finite automaton* (DFA) is a tuple $\mathcal{A} = \langle Q, \Sigma, \delta, I, F \rangle$ where Σ is a finite alphabet, Q is a finite set of states, $I \subseteq Q$ is the set of *initial* states, $F \subseteq Q$ is the set of *accepting* states, and $\delta \subseteq Q \times \Sigma \times Q$ is the transition *function*. Figure 2.1 shows an example of DFA. A *nondeterministic finite automaton* (NFA) is a tuple $\mathcal{A} = \langle Q, \Sigma, \delta, I, F \rangle$ where Σ is a finite alphabet, Q is a finite set of states, $I \subseteq Q$ is the set of *initial* states, $F \subseteq Q$ is the set of *accepting* states, and $\delta \subseteq Q \times \Sigma \times Q$ is the transition *relation*. Figure 2.2 shows an example of NFA. Observe that the difference between a DFA and a NFA is that in the former δ is a function, while in the latter δ is a relation.

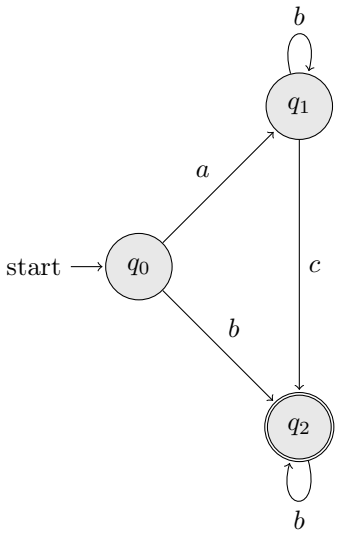

Figure 2.1: Example of a DFA

Let $q, q' \in Q$ and $a \in \Sigma$. If $q' \in \delta(q, a)$ we write that $q \xrightarrow{a} q'$ and if $\nexists q' \in \delta(q, a)$ we write that $q \xrightarrow{a}$. If $w \in \Sigma^*$, then $q \overset{w}{\rightsquigarrow} q'$ means that the state q' is reachable from q by following the string w. More formally, by induction on the length of w: (i) if $w = \epsilon$ then $q \overset{\epsilon}{\rightsquigarrow} q' \iff q = q'$; (ii) if $w = av$ with $a \in \Sigma$, $v \in \Sigma^*$ then $q \overset{av}{\rightsquigarrow} q' \iff \exists q'' \in \delta(q, a), q'' \overset{v}{\rightsquigarrow} q'$. We define $q \underset{u,q''}{\overset{uv}{\rightsquigarrow}} q' \overset{\triangle}{\iff} \exists q'' \in Q, q \overset{u}{\rightsquigarrow} q'' \wedge q'' \overset{v}{\rightsquigarrow} q'$. When $q \overset{u}{\rightsquigarrow} q' \wedge q' \overset{v}{\rightsquigarrow} q''$ we write $q \overset{u}{\rightsquigarrow} q' \overset{v}{\rightsquigarrow} q''$. We write $q \overset{w}{\rightarrowtail} q'$ to denote the fact that the state q' is reachable from q by following the string w, reaching at some point one final state. More formally, by induction on the length of the word w: (i) if $w = \epsilon$, then $q \overset{\epsilon}{\rightarrowtail} q'$ iff $q = q'$ and $q \in F$; (ii) if $w = av$, then $q \overset{av}{\rightarrowtail} q'$ iff $\exists q'' \in Q$ such that $((q \in F \vee q'' \in F) \wedge q \xrightarrow{a} q'' \wedge q'' \overset{v}{\rightsquigarrow} q')$ or $(q \xrightarrow{a} q'' \wedge q'' \overset{v}{\rightarrowtail} q')$. The *language accepted by the FA* $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) \overset{\triangle}{=} \{w \in \Sigma^* \mid \exists q_i \in I, \exists q_f \in F, q_i \overset{w}{\rightsquigarrow} q_f\}$. Figure 2.3 shows an example of an automaton accepting the language $a(b^*(ca)^*)^*$.

Let $S, T \subseteq Q$, we define $W^{\mathcal{A}}_{S,T} \overset{\triangle}{=} \{w \in \Sigma^* \mid \exists p \in S, \exists q \in T, p \overset{w}{\rightsquigarrow} q\}$. When $S = \{p\}$ or $T = \{q\}$ we slightly abuse the notation writing $W^{\mathcal{A}}_{p,T}$, $W^{\mathcal{A}}_{S,q}$ or $W^{\mathcal{A}}_{p,q}$. Let $w \in \Sigma^*$ and $S \subseteq Q$, we define $ctx_{\mathcal{A}}(w) \overset{\triangle}{=} \{(p, q) \in Q \times Q \mid p \overset{w}{\rightsquigarrow} q\}$, $ctx^F_{\mathcal{A}}(w) \overset{\triangle}{=} \{(p, q) \in Q \times Q \mid \exists p_f \in F, w_1, w_2 \in \Sigma^* : p \overset{w_1}{\rightsquigarrow} p_f \wedge p_f \overset{w_2}{\rightsquigarrow}, w = w_1 w_2\}$, $pre^{\mathcal{A}}_w(S) \overset{\triangle}{=} \{q \in Q \mid w \in W^{\mathcal{A}}_{q,S}\}$ and $post^{\mathcal{A}}_w(S) \overset{\triangle}{=} \{q \in Q \mid w \in W^{\mathcal{A}}_{S,q}\}$.

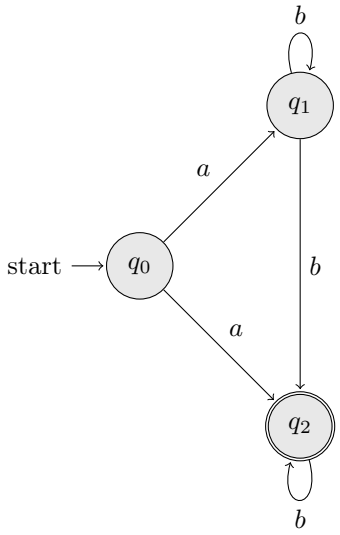

Figure 2.2: Example of a NFA

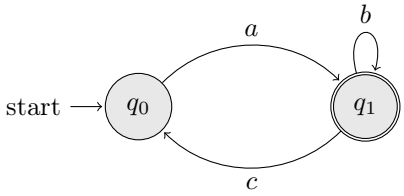

Figure 2.3: DFA accepting the language $a(b^*(ca)^*)^*$

$\mathcal{A}^R \triangleq \langle Q, \Sigma, \delta^R, F, I\rangle$ is the *reverse* of $\mathcal{A}$, where $q \in \delta^R(q', a) \iff q' \in \delta(q, a)$. If $q' \in \delta^R(q, a)$ we write that $q \xrightarrow{a}_R q'$. By induction on the length of w we define what means $q \overset{w}{\rightsquigarrow}_R q'$: (i) if $w = \epsilon$ then $q \overset{\epsilon}{\rightsquigarrow}_R q' \iff q = q'$; (ii) if $w = av$ with $a \in \Sigma$, $v \in \Sigma^*$ then $q \overset{av}{\rightsquigarrow}_R q' \iff \exists q'' \in \delta^R(q, a), q'' \overset{v}{\rightsquigarrow}_R q'$. Figure 2.4 shows the reverse of the DFA in Figure 2.3.

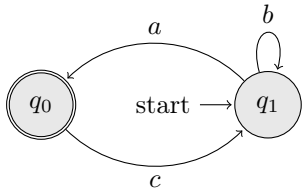

Figure 2.4: Reverse of the DFA in Figure 2.3

Lemma 2.2.1. Let $w \in \Sigma^*$ and $p, q \in Q$, then:

$$p \overset{w}{\rightsquigarrow} q \iff q \overset{w^R}{\rightsquigarrow}_R p$$

Proof. It follows from an induction on $|w|$. If $|w| = 0$, then $w = \epsilon$ and by the definition of $\rightsquigarrow$, $p = q$. If $|w| > 0$, then $w = av = ub$, $w^R = v^R a = bu^R$ for $a, b \in \Sigma$, $v, u \in \Sigma^*$

such that $av = ub$. If $p \overset{av}{\rightsquigarrow} q$, then $\exists p' \in Q$ such that $p \overset{a}{\rightarrow} p'$ and $p' \overset{v}{\rightsquigarrow} q$. For inductive hypothesis $q \overset{v^R}{\rightsquigarrow}_R p'$, and since $p \overset{a}{\rightarrow} p'$ implies $p' \overset{a}{\rightarrow}_R p$, $q \overset{v^R a}{\rightsquigarrow}_R p$. If $q \overset{bu^R}{\rightsquigarrow}_R p$, then $\exists q' \in Q$ such that $q \overset{b}{\rightarrow}_R q'$ and $q' \overset{u^R}{\rightsquigarrow}_R p$. For inductive hypothesis $p \overset{u}{\rightsquigarrow} q'$, and since $q \overset{b}{\rightarrow}_R q'$ implies $q' \overset{b}{\rightarrow} q$, $p \overset{ub}{\rightsquigarrow} q$. □

2.2.2 Context-free languages and context-free grammars

In order to define context-free languages we have to introduce the concept of context-free grammar. A *context-free grammar* (CFG) is a tuple $\mathcal{G} = \langle \mathcal{V}, \Sigma, P \rangle$ where $\mathcal{V} = \{X_0, X_1, \ldots, X_n\}$ is the finite set of *variables* including the start symbol X_0, Σ is the finite alphabet of *terminals* and P is is the set of productions $X_i \rightarrow \beta$ where $\beta \in (\mathcal{V} \cup \Sigma)^*$. We assume, for simplicity and without loss of generality, that CFGs are in *Chomsky Normal Form* (CNF), that is, every production $X_i \rightarrow \beta \in P$ is such that $\beta \in (\mathcal{V} \times \mathcal{V}) \cup \Sigma \cup \{\epsilon\}$ and if $\beta = \epsilon$ then $i = 0$ [Cho59]. We also assume that for all $X_i \in \mathcal{V}$ there exists a production $X_i \rightarrow \beta \in P$, otherwise X_i can be safely removed from $\mathcal{V}$. Given two words $w, w' \in (\mathcal{V} \cup \Sigma)^*$ we write $w \rightarrow w'$ iff there exists $u, v \in (\mathcal{V} \cup \Sigma)^*$ and $X \rightarrow \beta$ such that $w = uXv$ and $w' = u\beta v$. We denote by $\rightsquigarrow$ the reflexive-transitive closure of $\rightarrow$. The language accepted by a CFG $\mathcal{G}$ is $\mathcal{L}(\mathcal{G}) \triangleq \{w \in \Sigma^* \mid X_0 \rightsquigarrow w\}$. Figure 2.5 shows an example of CFG generating the language $\{a^n b^n \mid n \geq 0\}$, while Figure 2.6 shows a CFG that accepts the same language but is in CNF. The class of languages accepted by CFGs is called *contex-free languages*.

$$X_0 \rightarrow aX_0b$$
$$X_0 \rightarrow \epsilon$$

Figure 2.5: CFG that accepts the language $\{a^n b^n \mid n \geq 0\}$

$$X_0 \rightarrow X_2X_1$$
$$X_0 \rightarrow \epsilon$$
$$X_1 \rightarrow X_0X_3$$
$$X_2 \rightarrow a$$
$$X_3 \rightarrow b$$

Figure 2.6: CFG in *Chomsky Normal Form* that accepts the language $\{a^n b^n \mid n \geq 0\}$

2.2.3 ω-regular languages and Büchi automata

In order to define ω-regular languages we have to introduce the concept of Büchi automaton (BA). The definition of BA is analogous to the definition of FA, they differ just on how to accept words. BAs can be *deterministic* or *nondeterministic*.

A *deterministic Büchi automaton* (DBA) is tuple $\mathcal{B} = \langle Q, \Sigma, \delta, I, F \rangle$ where Σ is a finite alphabet, Q is a finite set of states, $I \subseteq Q$ is the set of *initial* states, $F \subseteq Q$ is the set of *accepting* states, and $\delta \subseteq Q \times \Sigma \times Q$ is the transition *function*. A

nondeterministic Büchi automaton (NBA) is tuple $\mathcal{B} = \langle Q, \Sigma, \delta, I, F \rangle$ where Σ is a finite alphabet, Q is a finite set of states, $I \subseteq Q$ is the set of *initial* states, $F \subseteq Q$ is the set of *accepting* states, and $\delta \subseteq Q \times \Sigma \times Q$ is the transition *relation*.

In order to define the language accepted by BAs we have to introduce the concept of *trace*. One *infinite trace* of the automaton $\mathcal{B} = \langle Q, \Sigma, \delta, I, F \rangle$ on an infinite word $w = a_0 a_1 \cdots \in \Sigma^\omega$ starting in a state $q_0 \in Q$ is an infinite sequence of transitions $\pi = q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} \cdots$. Similarly, a *finite trace* on a finite word $w = a_0 a_1 \ldots a_n \in \Sigma^*$ starting in a state $q_0 \in Q$ is a finite sequence of transitions $\pi = q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} \cdots \xrightarrow{a_n} q_{n+1}$. A trace is *initial* if it starts in an initial state $q_0 \in I$, and a finite trace is *final* if it ends in an accepting state $q_f \in F$. A trace is *fair* if it is infinite and $q_i \in F$ for infinitely many i's. Differently from FAs, Büchi automata recognize languages of *infinite* words. In particular, the *language* of a BA $\mathcal{B}$ is $\mathcal{L}(\mathcal{B}) \triangleq \{w \in \Sigma^\omega \mid \mathcal{B} \text{ has an initial and fair trace on } w\}$. Figure 2.7 shows an example of a Büchi automaton that accepts the language a^ω. An *ω-regular language* is a language recognized by a Büchi automaton.

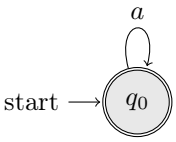

Figure 2.7: Büchi automaton that accepts the language a^ω

When the distinction between BAs and FAs is important we specify it, otherwise assume that $\mathcal{A}$ is an automaton such that $\mathcal{A} = \langle Q, \Sigma, \delta, I, F \rangle$. As a convertion, we will refer to FAs and generic automata as $\mathcal{A}$ and to BAs as $\mathcal{B}$.

2.2.4 The language inclusion problem

Deciding whether a formal language contains another one is a fundamental problem in computer science with diverse applications including automata-based verification [Kup18], reasoning for logical theories [Esp17] or type systems [HC14]. Typically, inclusion problems are computationally intensive.

Let L_1, L_2 be two languages. The *language inclusion problem* consists in checking if $L_1 \subseteq L_2$ holds. Whether $L_1 \subseteq L_2$ is computable or not, depends on the nature of L_1 and L_2.

It is well-known that if they are both regular the problem is computable, and many algorithms have been proposed to check the inclusion [GRV19; DW+06; Abd+10]. This problem is PSPACE-complete [Abd+10]. If L_1 is context-free and L_2 is regular the problem is still decidable. In Section 2.5.2 we present the framework put forward in [GRV19] to check this inclusion.

We are particularly in interested in the language inclusion problem between ω-regular languages. It is known to be a PSPACE-complete [KV96] problem and becomes EXPTIME-complete [MMN17] when the "smaller" language is ω-context free. In Section 2.5.1 we describe the framework for checking the language inclusion between ω-regular languages proposed in [DG20].

2.3 Order theory

Let D be a set. $\langle D, \leq \rangle$ is a *quasiordered set* (qoset) when $\leq$ is a *quasiorder* (qo) relation on D, that is, reflexive and transitive. We sometimes refer to quasiorders as

preorders. A qoset $\langle D, \leq \rangle$ is a *partially ordered set* (poset) when $\leq$ is antisymmetric. The *powerset* of D is one example of poset. A qoset satisfies the *ascending* (resp. *descending*) *chain condition* (ACC, resp. DCC) if there is no countably infinite sequence of distinct elements $\{x_i\}_{i \in \mathbb{N}}$ such that, for all $i \in \mathbb{N}, x_i \leq x_{i+1}$ (resp. $x_{i+1} \leq x_i$). An *antichain* in a qoset $\langle D, \leq \rangle$ is a subset $X \subseteq D$ such that any two distinct element in X are incomparable. We denote the set of antichains of a qoset $\langle D, \leq \rangle$ by $AC_{\langle D, \leq \rangle} \stackrel{\triangle}{=} \{X \subseteq D \mid X \text{ is an antichain}\}$. A qoset is called ACC (DCC) when it satisfies the ACC (DCC). A qoset $\langle D, \leq \rangle$ is a *well-quasiordered set* (wqoset) when $\leq$ is a *well-quasiorder* (wqo) relation on D, that is, when for every countably infinite sequence of elements $\{x_i\}_{i \in \mathbb{N}}$ there exist $i, j \in \mathbb{N}$ such that $i < j$ and $x_i \leq x_j$. For every qoset $\langle D, \leq \rangle$, let us define the following binary relation $\sqsubseteq$ on the $\wp(D)$: given $X, Y \in \wp(D)$, $X \sqsubseteq Y \stackrel{\triangle}{\iff} \forall x \in X, \exists y \in Y, y \leq x$. A *minor* of a subset $X \subseteq D$, denoted by $\lfloor X \rfloor$, is a subset of X satisfying: (i) $X \sqsubseteq \lfloor X \rfloor$ and (ii) $\lfloor X \rfloor$ is an *antichain*. Let us recall that every subset of a wqoset $\langle D, \leq \rangle$has at least one minor set, all minor sets are finite and if $\langle D, \leq \rangle$ is additionally a poset then there exists *exactly one* minor set.

Let $\langle D, \leq \rangle$ be a qoset. An *(upper) closure operator* on D, is a function in $D \to D$ that is monotone (i.e. $x \leq y \implies \rho(x) \leq \rho(y)$), idempotent (i.e., $\rho = \rho\rho$) and increasing (i.e., $x \leq \rho(x)$). The qoset D induces a closure operator $\rho_{\leq}$ defined by $\rho_{\leq}(X) \stackrel{\triangle}{=} \{y \in D \mid \exists x \in X, x \leq y\}$. We call $\rho_{\leq}(X)$ the *upward closure* of X and if $x \in X$ we abuse the notation and write $\rho_{\leq}(x)$ for $\rho_{\leq}(\{x\})$.

Let $\langle D, \leq \rangle$ be a poset. Given two elements x and y in D, we call *upper bound* of x and y any element $z \in D$ such that $x \leq z$ and $y \leq z$. Likewise, a *lower bound* z of x and y satisfies $z \leq x$ and $z \leq y$. Moreover, z is the *least upper bound*, also called *lub* or *join*, if it is the smallest element grater than both x and y. Such a upper bound does not necessarily exist but, when it does, it is unique. It is written as $x \sqcup y$. Likewise, the unique *greatest lower bound* of x and y, also called *glb* or *meet*, if it exists, is the greatest element smaller than x and y, and it is denoted as $x \sqcap y$. As $\sqcup$ and $\sqcap$ are associative operators, we will employ also the notations $\sqcup A$ and $\sqcap A$ to compute lubs an glbs on arbitrary (possibly infinite) sets A of elements. We denote respectively as $\bot$ (called *bottom*) and $\top$ (called *top*) the *least element* and the *greatest element* in the poset, if they exist.

A *chain* $C \subseteq D$ in a poset $\langle D, \leq \rangle$ is a subset C of D such that is totally ordered: $\forall x, y \in C : (x \leq y) \vee (y \leq x)$. A *complete partial order* (CPO) is a poset such that every chain has a lub.

A *lattice* $\langle D, \leq, \sqcup, \sqcap \rangle$ is a poset such that $\forall x, y, \in D : x \sqcup y$ and $x \sqcap y$ exist. A *join-semilattice* $\langle D, \leq, \sqcup, \bot \rangle$ is a poset that has the lub of all its arbitrary nonempty finite subsets. A *complete lattice* $\langle D, \leq, \sqcup, \sqcap, \bot, \top \rangle$ is a poset that has the lub of all its arbitrary (possibly empty) subsets.

2.3.1 Fixpoints

Let $\langle D, \leq \rangle$ be a poset, $f : D \to D$ a function and $x \in D$.

- x is a *fixpoint* of f if $f(x) = x$. We denote as $fp(f) \stackrel{\triangle}{=} \{x \in D \mid f(x) = x\}$ the set of fixpoints of f;
- x is a *prefixpoint* of f is $x \leq f(x)$;
- x is a *postfixpoint* of f is $f(x) \leq x$;

- $\text{lfp}_x\ f \stackrel{\triangle}{=} min\{y \in fp(f) \mid x \leq y\}$, if it exists, is the *least fixpoint* of f greater than x;
- $\text{lfp}\ f \stackrel{\triangle}{=} \text{lfp}_\bot f$, is it exists, is the *least fixpoint* of f;
- $\text{gfp}_x\ f \stackrel{\triangle}{=} max\{y \in fp(f) \mid y \leq x\}$, if it exists, is the *greatest fixpoint* of f greater than x;
- $\text{gfp}\ f \stackrel{\triangle}{=} \text{gfp}_\top f$, is it exists, is the *greatest fixpoint* of f.

In order to guarantee the existence of fixpoints we need some extra hypotheses on f and $\langle D, \leq \rangle$. A function $f : \langle D_1, \leq_1 \rangle \to \langle D_2, \leq_2 \rangle$ between two posets is *monotonic* if $\forall x, y \in D_1 : x \leq_1 y \Longrightarrow f(x) \leq_2 f(y)$. Monotonicity is related to the existence of fixpoints, as the following Theorem states [Tar+55].

Theorem 2.3.1 (Tarski's Theorem)**.** *If $f : D \to D$ is a monotonic operator is a complete lattice $\langle D, \leq, \sqcup, \sqcap, \bot, \top \rangle$, then the set of fixpoints $fp(p)$ is a non-empty complete lattice. In particular, lfp f exists. Furthermore, lfp $f = \sqcap\{x \in D \mid f(x) \leq x\}$.*

A function $f : \langle D_1, \leq_1, \sqcup_1 \rangle \to \langle D_2, \leq_2, \sqcup_2 \rangle$ between two CPOs is *continuous* if for every chain $C \subseteq D_1$, $\{f(c) \mid c \in C\}$ is also a chain and the limits coincide: $f(\sqcup_1 C) = \sqcup_2 \{f(c) \mid c \in C\}$. Another useful fixpoint theorem, found, e.g., in [CC77], is inspired by Kleene[Kle+52]'s star construction:

Theorem 2.3.2 (Kleene's Theorem)**.** *If $f : D \to D$ is a continuous function in a CPO $\langle D, \leq, \sqcup, \top \rangle$, then lfp f exists. Moreover, lfp $f = \sqcup\{f^i(\bot) \mid i \in \mathbb{N}\}$.*

This theorem requires stronger hypotheses on the function f than Taski's Theorem, but it does not require a complete lattice, only a CPO. More interestingly, the iteration scheme makes the theorem *constructive*: one can imagine the process of computing iterations one by one from $\bot$. Due to Kleene's Theorem, the following procedure provides the possibly infinite sequence of so-called *Kleene iterates* of the function f starting from the basis x.

$$\texttt{Kleene(}f\texttt{,}x\texttt{)} \stackrel{\triangle}{=} \begin{cases} y \leftarrow x; \\ \textbf{while} f(y) \neq y \textbf{ do} \\ \quad\quad y \leftarrow f(y); \\ \textbf{return } y \end{cases}$$

When D is an ACC CPO and f is monotonic, `Kleene(`f`,`x`)` terminates and returns the least fixpoint of the function f which is greater than or equal to x. Moreover, under the same hypotheses `Kleene(`f`,`$\bot$`)` returns the lfp of the function f.

2.3.2 Quasiorders on words

A quasiorder $\leq$ on Σ^* is *left-monotonic* (*right-monotonic*) if $\forall y, x_1, x_2 \in \Sigma^*, x_1 \leq x_2 \Longrightarrow yx_1 \leq yx_2$ ($x_1 y \leq x_2 y$). Also, $\leq$ is called *monotonic* if it is both left and right-monotonic.

Definition 2.3.3 (L-consistency)**.** Let $L \in \wp(\Sigma^*)$ and $\leq_L$ be a quasiorder on Σ^*. Then, $\leq_L$ is called left (resp. right) $L-$consistent if and only if: (i) $\leq_L \cap (L \times \neg L) = \varnothing$; (ii) $\leq_L$ is left (resp.right) monotonic. Furthermore, $\leq_L$ is called $L-$consistent when it is both left and right $L-$consistent.

We see in Section 2.5.2 that if one qo monotonic and L-consistent, then it can be used in the framework described in [GRV19] to solve the language inclusion problem $L_1 \subseteq L_2$ where L_1 is context-free and L_2 is regular.

We present some wqos defined in [GRV19]. Observe that $\preceq^{di}$ is the direct simulation relation on states, while $\preceq^{r}$ is the reverse simulation on states (see Section 2.4). Let $L \in \wp(\Sigma^*)$, $\mathcal{A}$ be an automaton, $u, v \in \Sigma^*$:

$$u \leqq_L v \overset{\triangle}{\iff} ctx_L(u) \subseteq ctx_L(v)$$

$$u \leq^1_{\mathcal{A}} v \overset{\triangle}{\iff} ctx_{\mathcal{A}}(u) \subseteq ctx_{\mathcal{A}}(v)$$

$$u \sqsubseteq^r_{\mathcal{A}} v \overset{\triangle}{\iff} \forall q \in post^{\mathcal{A}}_u(I)\ \exists p \in post^{\mathcal{A}}_v(I) \text{ such that } q \preceq^{di} p$$

$$u \sqsubseteq^l_{\mathcal{A}} v \overset{\triangle}{\iff} \forall q \in pre^{\mathcal{A}}_u(F)\ \exists p \in pre^{\mathcal{A}}_v(F) \text{ such that } q \preceq^{r} p$$

The first is known as the *Myhill quasiorder*, while the second is a *state-based quasiorder*. If $\mathcal{A}$ is a FA, they are both decidable monotonic wqos [GRV19] and $\leqq_A$ is $L-$consistent. Furthermore, also $\leq^1_{\mathcal{A}}$ is $\mathcal{L}(\mathcal{A})$-consistent. $\sqsubseteq^r_{\mathcal{A}}$ is right-monotonic and $\sqsubseteq^l_{\mathcal{A}}$ is left-monotonic. If $\mathcal{A}$ is a FA they are respectively right $\mathcal{L}(\mathcal{A})$-consistent and left $\mathcal{L}(\mathcal{A})$-consistent. They are both decidable wqos as well.

Some of these qos will be referenced frequently, in particular $\leq^1_{\mathcal{A}}$ will be used in the framework for solving the language inclusion problem for ω-regular languages described in Section 2.5.1. Furthermore, $\sqsubseteq^r_{\mathcal{A}}$ is the first right-monotonic simulation-based qo that we present. In Chapter 3 we will generalize $\sqsubseteq^r_{\mathcal{A}}$ in order to obtain a number of different right-monotonic simulation-based qos.

2.4 Simulation quasiorders on states

We assume that the automata are *forward* and *backward* complete (or simply *complete*), i.e., for any state $p \in Q$ and symbol $a \in \Sigma$, there exist states $q_0, q_1 \in Q$ such that $q_0 \xrightarrow{a} p \xrightarrow{a} q_1$. In our examples not every automaton is backward and forward complete, because otherwise they would be too complicated. Observe that in the examples we can consider incomplete automata without loss of generality: as Proposition 2.4.1 and 2.4.2 state, every automaton can be converted into an equivalent complete one.

Proposition 2.4.1. Let $\mathcal{A} = \langle Q, \Sigma, \delta, I, F \rangle$ be a FA. There exists a backward and forward complete FA $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.

Proof. We define $\mathcal{A}'$ in terms of the components Q, δ, I, F of $\mathcal{A}$:

- $Q' = Q \cup \{b, f\}$, where b and f are two new states;
- $\delta' = \delta \cup \{(b, a, q) \mid \nexists (q', a, q) \in \delta\} \cup \{(b, a, b) \mid a \in \Sigma\} \cup \{(q, a, f) \mid \nexists (q, a, q') \in \delta\} \cup \{(f, a, f) \mid a \in \Sigma\}$;
- $I' = I$;
- $F' = F$.

The runs of $\mathcal{A}'$ are a superset of those of $\mathcal{A}$ since we have added states and transitions. Furthermore, on any input word w the accepting runs of $\mathcal{A}$ and $\mathcal{A}'$ correspond, because any run that reaches f stays in f, and since $f \notin F'$, such a run can't reach a final state. □

Proposition 2.4.2. Let $\mathcal{B} = \langle Q, \Sigma, \delta, I, F \rangle$ be a BA. There exists a backward and forward complete BA $\mathcal{B}'$ such that $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{B}')$.

Proof. We define $\mathcal{B}'$ in terms of the components Q, δ, I, F of $\mathcal{B}$:

- $Q' = Q \cup \{b, f\}$, where b and f are two new states;
- $\delta' = \delta \cup \{(b, a, q) \mid \nexists (q', a, q) \in \delta\} \cup \{(b, a, b) \mid a \in \Sigma\} \cup \{(q, a, f) \mid \nexists (q, a, q') \in \delta\} \cup \{(f, a, f) \mid a \in \Sigma\}$;
- $I' = I$;
- $F' = F$.

The runs of $\mathcal{B}'$ are a superset of those of $\mathcal{B}$ since we have added states and transitions. Furthermore, on any infinite input word w the accepting runs of $\mathcal{B}$ and $\mathcal{B}'$ correspond, because any run that reaches f stays in f, and since $f \notin F'$, such a run is not fair. □

In the following example we show how to transform one incomplete FA in a complete one.

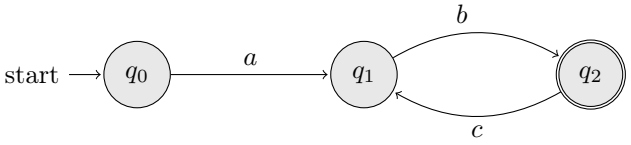

Figure 2.8: Incomplete FA accepting the language $ab(cb)^*$

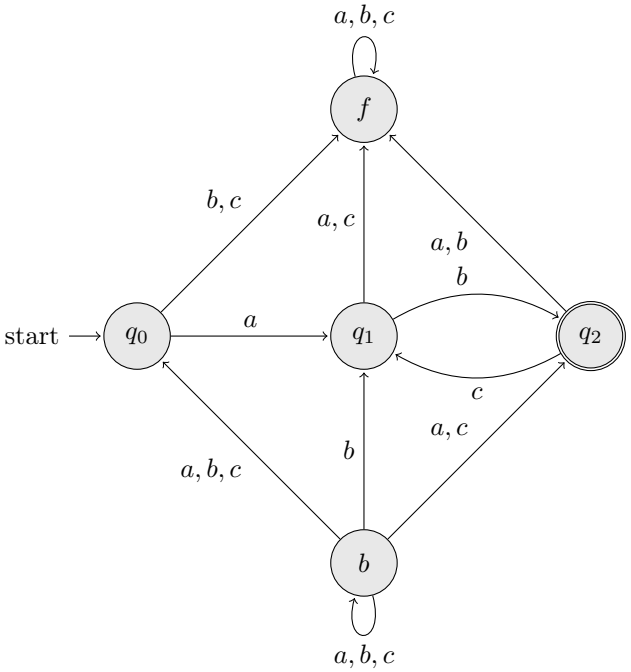

Figure 2.9: Complete FA accepting the language $ab(cb)^*$

Example 2.4.3. Let $\mathcal{A}$ be the FA in Figure 2.8. If we add states b and f and the transitions described in Proposition 2.4.1 we obtain the FA $\mathcal{A}'$ in Figure 2.9. It is easy to check that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.

The *simulation relation* between two states p_0 and q_0 can be described in terms of a game between two players, Spoiler (he) and Duplicator (she), where the latter wants to prove that q_0 can step-wise mimic any behaviour of p_0, and the former wants to disprove it. The game starts in the initial configuration (p_0, q_0). Inductively, given a game configuration (p_i, q_i) at the i-th round of the game, Spoiler chooses a symbol $a_i \in \Sigma$ and a transition $p_i \xrightarrow{a_i} p_{i+1}$. Then, Duplicator responds by choosing a matching transition $q_i \xrightarrow{a_i} q_{i+1}$, and the next configuration is (p_{i+1}, q_{i+1}). The game goes on forever, and the two players build two infinite traces $\pi_0 = p_0 \xrightarrow{a_0} p_1 \xrightarrow{a_1} \cdots$ and $\pi_1 = q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} \cdots$. The winning condition for Duplicator is a predicate on the two traces π_0, π_1 and it depends on the type of simulation. For our purposes, we firstly consider direct[DHWT91], delayed[EWS05] and fair[HHK95] simulations. Let $k \geq 0$, $x \in \{di, de, f\}$. Duplicator wins the play if $\mathcal{C}^x(\pi_0, \pi_1)$ holds, where:

$$\mathcal{C}^{di}(\pi_0, \pi_1) \iff \forall(i \geq 0) \cdot p_i \in F \implies q_i \in F$$
$$\mathcal{C}^{de}(\pi_0, \pi_1) \iff \forall(i \geq 0) \cdot p_i \in F \implies \exists(j \geq i) \cdot q_j \in F$$
$$\mathcal{C}^{f}(\pi_0, \pi_1) \iff \text{if } \pi_0 \text{ is fair, then } \pi_1 \text{ is fair}$$

We define x-simulation relation $\preceq^x \subseteq Q \times Q$, for $x \in \{di, de, f\}$, by stipulating that $p_0 \preceq^x q_0$ holds if Duplicator has a winning strategy in the x-simulation game, starting from configuration (p_0, q_0). Thus, $\preceq^{di} \subseteq \preceq^{de} \subseteq \preceq^{f}$. Furthermore $\preceq^{di}, \preceq^{de}$ and $\preceq^{f}$ are PTIME computable *quasiroders* [DHWT91; EWS05; HHK95; HKR02].

Example 2.4.4. In this example we show the difference between the delayed and the fair simulation. Consider the family of automata $\mathcal{A}_n$ in Figure 2.10 (taken from [EWS05]). Observe that while $q_0 \preceq^{f} q_1$, $q_0 \not\preceq^{de} q_1$. We are playing the delayed simulation game: Spoiler starts at q_0, Duplicator at q_1. Spoiler plays an infinite word *not accepted* by $\mathcal{A}_n$, e.g. a_2^ω. Since $\mathcal{A}_n$ is deterministic (and complete), Duplicator has no choice but to self loop on her starting state. Spoiler visited one accepting state but at no point in future Duplicator will match that visit. □

We also consider the *reverse* and the *backward* simulation, that rely on the reverse transition $\rightarrow_R$. Let $y \in \{r, b\}$, $\preceq^y$ simulation is defined like ordinary simulation except that transitions are taken backwards: from configuration (p_i, q_i), Spoiler selects a transition $p_{i+1} \xrightarrow{a_i} p_i$, Duplicator replies with a transition $q_{i+1} \xrightarrow{a_i} q_i$, and the next configuration is (p_{i+1}, q_{i+1}). Let $\pi_0 = \cdots \xrightarrow{a_1} p_1 \xrightarrow{a_0} p_0$ and $\pi_1 = \cdots \xrightarrow{a_1} q_1 \xrightarrow{a_0} q_0$ be the infinite backward traces built in this way. Duplicator wins the play if $\mathcal{C}^y(\pi_0, \pi_1)$ holds, where:

$$\mathcal{C}^{r}(\pi_0, \pi_1) \iff \forall(i \geq 0) \cdot p_i \in I \implies q_i \in I$$
$$\mathcal{C}^{b}(\pi_0, \pi_1) \iff \forall(i \geq 0) \cdot p_i \in I \implies q_i \in I \text{ and } p_i \in F \implies q_i \in F$$

Then, $p_0 \preceq^y q_0$ holds if Duplicator has a winning strategy in the above described game starting from (p_0, q_0) with winning condition $\mathcal{C}^y$. It holds that $\preceq^b \subseteq \preceq^r$. Furthermore, $\preceq^b$ and $\preceq^r$ are efficiently computable *quasiorders* [SB00].

We also consider the *k-lookahead* simulation [CM17]. Intuitively, we put the lookahead under Duplicator's control, who can choose at each round and depending on Spoiler's move how much lookahead she needs (up to k). Formally, configurations are

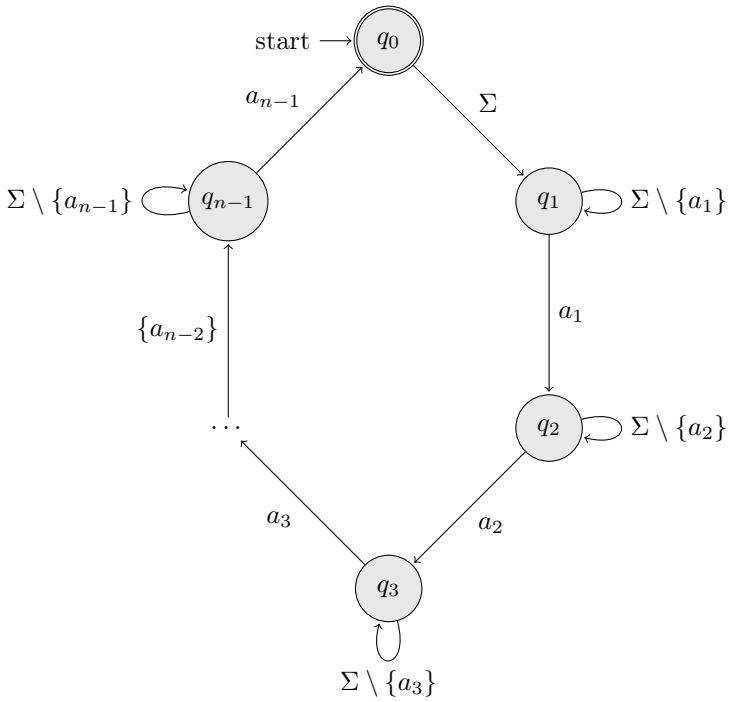

Figure 2.10: Family of automata $\mathcal{A}_n$

pairs (p_i, q_i) of states. From configuration (p_i, q_i), one round of the game is played as follows.

- Spoiler chooses a sequence of k consecutive transitions $p_i \xrightarrow{a_i} p_{i+1} \xrightarrow{a_{i+1}} \cdots \xrightarrow{a_{i+k-1}} p_{i+k}$;
- Duplicator chooses a degree of lookahead m such that $1 \leq m \leq k$;
- Duplicator responds with a sequence of m transitions $q_i \xrightarrow{a_i} q_{i+1} \xrightarrow{a_{i+1}} \cdots \xrightarrow{a_{i+m-1}} q_{i+m}$.

The remaining $k - m$ moves of Spoiler $p_{i+m} \xrightarrow{a_{i+m}} p_{i+m+1} \xrightarrow{a_{i+m+1}} \cdots \xrightarrow{a_{i+k-1}} p_{i+m}$ are forgotten, and the next configuration is (p_{i+m}, q_{i+m}); in particular, in the next round Spoiler can choose a different attack from p_{i+m}. In this way, the players build as usual two infinite traces π_0 from p_0 and π_1 from q_0. For any acceptance condition $x \in \{di, de, f\}$, Duplicator wins this play if $\mathcal{C}^x(\pi_0, \pi_1)$ holds. We define the *k-lookahead x-simulation* $\sqsubseteq^{k-x} \subseteq Q \times Q$, by stipulating that $p_0 \sqsubseteq^{k-x} q_0$ holds if Duplicator has a winning strategy in the above game, starting from configuration (p_0, q_0). Since for $k > 1$ $\sqsubseteq^{k-x}$ is not transitive (see Example 2.4.5), we consider instead its *transitive clouse* (see Section 2.3), and we denote it by $\preceq^{k-x}$. Even a moderate lookahead often yields much coarser relation than normal simulation quasiorder [CM17].

Example 2.4.5. Let $\mathcal{A}$ be the automaton in Figure 2.11 (taken from [CM17]). We are playing the 2-lookahead game: Spoiler starts from q_0, while Duplicator from q_1. If Spoiler plays $q_0 \xrightarrow{a} q_0 \xrightarrow{a} q_0$, Duplicator can reply with $q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_1$. The other cases are similar, so that $q_0 \sqsubseteq^{2-x} q_1$. Consider another configuration: Spoiler starts from q_1

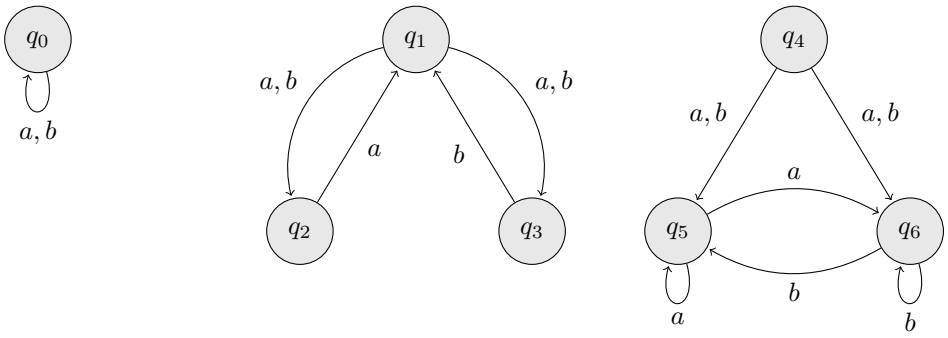

Figure 2.11: A k-lookahead simulation example.

and Duplicator from q_4. If Spoiler goes to q_2 or q_3, then Duplicator goes to q_5 or q_6, respectively. If Spoiler plays $q_2 \xrightarrow{a} q_1 \xrightarrow{a} q_2$, Duplicator does $q_5 \xrightarrow{a} q_5 \xrightarrow{a} q_5$. If Spoiler plays $q_2 \xrightarrow{a} q_1 \xrightarrow{b} q_2$, then Duplicator responds with $q_5 \xrightarrow{a} q_6 \xrightarrow{b} q_5$. The other cases are similar, so that $q_1 \sqsubseteq^{2-x} q_4$. Consider the last configuration: Spoiler starts from q_0 and Duplicator from q_4. If Spoiler plays $q_0 \xrightarrow{a} q_0 \xrightarrow{a} q_0$, Duplicator can play $q_4 \xrightarrow{a} q_5$, $q_4 \xrightarrow{a} q_6$, $q_4 \xrightarrow{a} q_5 \xrightarrow{a} q_5$ or $q_4 \xrightarrow{a} q_5 \xrightarrow{a} q_6$. In the first and the third case Spoiler plays $q_0 \xrightarrow{b} q_0$ and Duplicator loses, in the second and in the fourth $q_0 \xrightarrow{a} q_0$. To conclude, $q_0 \sqsubseteq^{2-x} q_4$ doesn't hold. This example shows that $\sqsubseteq^{2-x}$ is not transitive, so that in general, for $k > 1$, $\sqsubseteq^{k-x}$ is not transitive.

The last relations that we consider are the *trace inclusions*, which are obtained through the following modification of the simulation game. In a simulation game, the players build two paths π_0, π_1 by choosing single transitions in an alternating fashion. That is, Duplicator moves by a single transition by knowing only the next single transition chosen by Spoiler. We can obtain coarser relations by allowing Duplicator a certain amount of lookahead on Spoiler's chosen transitions. In the extremal case of *infinite lookahead*, i.e., where Spoiler has to reveal his entire path in advance, we obtain *trace inclusions*. Analogously to simulations, we define direct, delayed, and fair trace inclusion, as binary relations on Q. Formally, for $x \in \{di, de, f\}$, x-trace inclusion holds between p and q, written $p \preceq^{t-x} q$ if, for every word $w = a_1 a_2 \cdots \in \Sigma^\omega$, and for every infinite w-trace $\pi_0 = p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} \cdots$ starting at $p_0 = p$, there exists an infinite w-trace $\pi_1 = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots$ starting at $q_0 = q$ such that $\mathcal{C}^x(\pi_0, \pi_1)$ holds.

Like simulations, trace inclusions are preorders. Clearly, direct trace inclusion $\preceq^{t-di}$ is a subset of delayed trace inclusion $\preceq^{t-de}$, which, in turn, is a subset of fair trace inclusion $\preceq^{t-f}$. Moreover, since Duplicator has more power in the trace inclusion game than in the corresponding simulation game, trace inclusions subsume the corresponding simulation. This is true also for the corresponding k-lookahead simulation. Furthermore, trace inclusions are PSPACE-computable quasiorders [Ete02; CM18].

2.4.1 Computing simulations

Here we describe some of the leading ideas than have been proposed to compute simulation quasiorders. At the end of this section we also give a reference to some tools to compute the simulations.

Direct simulation. In what follows for simplicity we will refer to the direct

simulation just as simulation. To compute the simulation $\preceq^{di}$ the algorithm that has mostly influenced the literature is that described in [HHK95]. It runs in $O(|Q| \cdot | \preceq^{di} |)$ time and uses $O(|Q|^2 \cdot log(|Q|))$ space. There are many known algorithms that improve the time and space complexity of [HHK95], for example [BG03], [RT07] and [CRT11]. Here we present the leading ideas of the approach taken in [Céc17], since it is the one with the best published time and space complexities. Observe that since $\preceq^{di}$ is a quasiorder, it can be efficiently represented as a *partition-relation pair* (P, R) with P a *partition* of the states Q, whose blocks are classes of the simulation equivalence relation and with $R \subseteq P \times P$ a quasiorder over the blocks of P [Céc17]. The first observation is that a relation $\mathscr{S}$ is a simulation if:

$$\mathscr{S} \circ \delta^R \subseteq \delta^R \circ \mathscr{S}$$

This is an alternative characterization that helps us design efficient algorithms to compute the simulation. In particular, if a relation $\mathscr{R}$ is not a simulation we have $\mathscr{R} \circ \delta^R \nsubseteq \delta^R \circ \mathscr{R}$. This implies the existence of a relation *Remove* such that $\mathscr{R} \circ \delta^R \subseteq (\delta^R \circ \mathscr{R}) \cup Remove$. Many of the previously cited works rely on this equation in order to compute the simulation preorder. In the described paper they take advantage of the equation in a different way: when $\mathscr{R}$ is not a simulation, it is possible to reduce the problem of finding the coarsest simulation inside $\mathscr{R}$ to the case where there is a relation *NotRel* such that $\mathscr{R} \circ \delta^R \subseteq \delta^R \circ (\mathscr{R} \cup NotRel)$. Let us denote $\mathscr{U} \triangleq \mathscr{R} \cup NotRel$. We will say that $\mathscr{R}$ is $\mathscr{U}$*-stable* since we have:

$$\mathscr{R} \circ \delta^R \subseteq \delta^R \circ \mathscr{U}$$

Consider Figure 2.12 (taken from [Céc17]) and the transition $c \to b$. The quasiorder $\mathscr{R}$

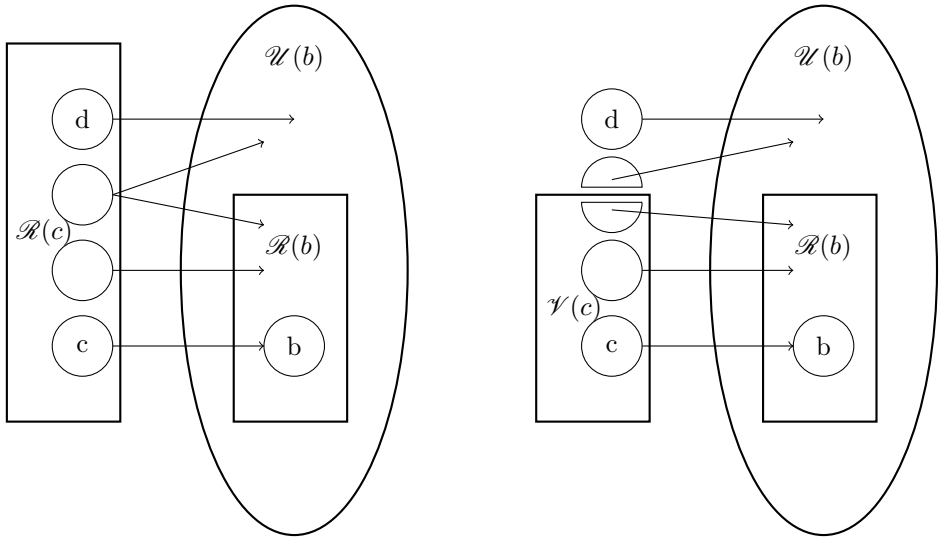

Figure 2.12: $\mathscr{R}$ is $\mathscr{U}$-stable and $\mathscr{V}$, obtained after a split of blocks of $\mathscr{R}$ and refinement of $\mathscr{R}$, is $\mathscr{R}$-stable

is assumed to be $\mathscr{U}$-stable and we want to find the coarsest simulation included in $\mathscr{R}$. Observe that since $\mathscr{R}$ is a quasiorder the set $\mathscr{R}(c)$ is a union of blocks of $\mathscr{R}$. A state d in $\mathscr{R}(c)$ which doesn't have an outgoing transition to $\mathscr{R}(b)$ belongs to $\delta^R \circ \mathscr{U}(b)$, but cannot simulate c. In order to compute the coarsest simulation included in $\mathscr{R}$ we

can remove d from $\mathscr{R}(c)$. To be efficient we want to manage *blocks* of states, rather than individual ones. Then, the first step is a *split step*, in which we split the blocks of $\mathscr{R}$ in two groups: the ones that are are completely included in $\delta^R \circ \mathscr{R}(b)$ and hence have a chance to simulate c, and ones that are completely outside $\delta^R \circ \mathscr{R}(b)$ that then cannot simulate c. Let $\mathscr{P}$ be the equivalence relation associated to the resulting partition. Since $\mathscr{P}$ is an equivalence, for each block of the partition that has one outgoing transition to $\delta^R \circ (\mathscr{U} \setminus \mathscr{R})(b)$ it is sufficient to test *only for one element* of the block if it is included in $\delta^R \circ \mathscr{R}(b)$. For each block E we call this element the *representative* of E and we denote it by $E.rep$. To perform the test in constant time we manage a counter that at first counts the number of transitions from $E.rep$ to $\mathscr{U}(b) = \mathscr{U}([b]_{\mathscr{P}})$. Then, we get the following equivalences: there is no transition from E to $\mathscr{R}(b)$ iff there is no transition from $E.rep$ to $\mathscr{R}(b)$ iff this counter is null. We can safely remove from $\mathscr{R}(c)$ those blocks of $\delta^R \circ (\mathscr{U} \setminus \mathscr{R})(b)$ which do not have an outgoing transition to $\mathscr{R}(b)$. We call this step the *refinement step*. After performing it $\mathscr{R}(c)$ has been reduced to $\mathscr{V}(c)$. Doing these split and refinement steps for all transitions from c to b results in the relation $\mathscr{V}$ that is a $\mathscr{R}$-stable quasiorder.

In summary, from an initial preorder we build a strictly decreasing sequence of preorders $\{\mathscr{R}_i\}_{i\geq 0}$ such that $\mathscr{R}_{i+1}$ is $\mathscr{R}_i$-stable and contains all the simulations included in $\mathscr{R}_i$. We observe that all the relations are finite and hence this sequence has a limit that is reached after a finite number of steps: and we call this limit $\mathscr{R}_{sim}$. We have that $\mathscr{R}_{sim}$ is $\mathscr{R}_{sim}$-stable. Therefore, we observe that $\mathscr{R}_{sim}$ is a simulation and by construction contains all simulations included in the initial quasiorder, hence is the coarsest one and exactly corresponds to $\preceq^{di}$. We denote by $P_{\preceq^{di}}$ the partition of Q induced by $\preceq^{di}$. The time complexity of the algorithm proposed in [Céc17] is $O(|P_{\preceq^{di}}| \cdot |\delta|)$, while it uses $O(|P_{\preceq^{di}}|^2 \cdot log(|P_{\preceq^{di}}|) + |Q| \cdot log(|Q|))$ space.

Delayed and fair simulation. Also for the fair and the delayed simulations a wide variety of algorithms have been proposed [CM17; HKR02]. Here we briefly discuss approach taken in [EWS05]. The first step is to define a *parity game graph* $G_{\mathcal{B}}$ on one Büchi automaton $\mathcal{B}$ such that the winning vertices in $G_{\mathcal{B}}$ for Even (one of the two players) in the corresponding parity game determine precisely the pairs of states (q, q') of $\mathcal{B}$ where q' simulates q.

A *parity game graph* $G = \langle V_0, V_1, E, p\rangle$ has two disjoint sets of vertices, V_0 and V_1 whose union is denoted by V. There is an edge set $E \subseteq V \times V$, and $p : V \to \{0, \ldots, d-1\}$ is a mapping that assigns a *priority* to each vertex. A *parity game* on G, starting at vertex $v_0 \in V$, is denoted by $P(G, v_0)$, and is played by two players, *Even* and *Odd*. The play starts by placing a pebble on vertex v_0. Thereafter, the pebble is moved according to the following rule: with the pebble currently on a vertex v_i, and $v_i \in V_0(V_1)$, Even (Odd, respectively) plays and moves the pebble to a neighbour v_{i+1}, that is, such that $(v_i, v_{i+1}) \in E$.

If ever the above rule cannot be applied, i.e., someone can't move because there are no outgoing edges, the game ends, and the player who cannot move loses. Otherwise, the game goes on forever and defines a path $\pi = v_0 v_1 v_2$ in G, called a *play* of the game. The winner of the play is determined as follows. Let k_π be the minimum priority that occurs infinitely often in the play π, i.e., so for infinitely many i, $p(v_i) = k_\pi$ and k_π is the least number with this property. Even wins if k_π is even, whereas Odd wins if k_π is odd.

We can now show how to build the parity game graph $G^f_{\mathcal{B}}$ for the fair simulation. We define $G^f_{\mathcal{B}} \triangleq \langle V^f_0, V^f_1, E^f_{\mathcal{B}}, p^f_{\mathcal{B}}\rangle$, where:

$$V^f_0 \triangleq \{v_{(q,q',a)} \mid q, q' \in Q \ \wedge \ \exists q'' : q \in \delta(q'', a)\}$$

$$V_1^f \triangleq \{v_{(q,q')} \mid q, q' \in Q\}$$

$$E_{\mathcal{B}}^f \triangleq \{(v_{(q_1,q_1',a)}, v_{(q_1,q_2)}) \mid q_2' \in \delta(q_1', a)\} \cup \{(v_{(q_1,q_1')}, v_{(q_2,q_1',a)}) \mid q_2 \in \delta(q_1, a)\}$$

$$p_{\mathcal{B}}^f(v) = \begin{cases} 0 & \text{if } v = v_{(q,q')} \text{ and } q' \in F \\ 1 & \text{if } v = v_{(q,q')} \text{ and } q \in F, \text{ and } q' \notin F \\ 2 & \text{otherwise} \end{cases}$$

We show that the defined parity game mimics the fair simulation game. Here Odd takes the role of Spoiler, while Even takes the role of Duplicator. When in the game the current position is node $v_{(q,q')}$, this corresponds to the situation in which is Spoiler's turn to move, while $v_{(q,q',a)}$ denotes the situation in which is Duplicator's turn to match Spoiler's play, that played a. The priority function is defined in such a way that every time Duplicator visits a final state, the priority function returns 0. It returns 1 only if Spoiler visits a final state, but Duplicator does not. In all other cases, 2 is returned. That is, Spoiler wins iff he visits an accept state infinitely often but Duplicator does not. It turns out that for every pair of states $q, q' \in Q$, $q \preceq^f q'$ iff Even has a winning strategy in $P(G_{\mathcal{B}}^f, v_{(q,q')})$.

Similarly, it is possible to define the parity game graph $G_{\mathcal{B}}^{de} \triangleq \langle V_0^{de}, V_1^{de}, E_{\mathcal{B}}^{de}, p_{\mathcal{B}}^{de} \rangle$ for the delayed simulation, where:

$$V_0^{de} \triangleq \{v_{(b,q,q',a)} \mid q, q' \in Q \ \wedge \ b \in \{0,1\} \ \wedge \ \exists q'' : q \in \delta(q'', a)\}$$

$$V_1^{de} \triangleq \{v_{(b,q,q')} \mid q, q' \in Q \ \wedge \ b \in \{0,1\} \ \wedge \ (q' \in F \implies b = 0)\}$$

$$\begin{aligned} E_{\mathcal{B}}^{de} \triangleq & \{(v_{(b,q_1,q_1',a)}, v_{(b,q_1,q_2')}) \mid q_2' \in \delta(q_1', a) \ \wedge \ q_2' \notin F\} \cup \\ & \{(v_{(b,q_1,q_1',a)}, v_{(0,q_1,q_2')}) \mid q_2' \in \delta(q_1', a) \ \wedge \ q_2' \in F\} \cup \\ & \{(v_{(b,q_1,q_1')}, v_{(b,q_2,q_1',a)}) \mid q_2 \in \delta(q_1, a) \ \wedge \ q_2 \notin F\} \cup \\ & \{(v_{(b,q_1,q_1')}, v_{(1,q_2,q_1',a)}) \mid q_2 \in \delta(q_1, a) \ \wedge \ q_2 \in F\} \end{aligned}$$

$$p_{\mathcal{B}}^{de}(v) = \begin{cases} b & \text{if } v = v_{(b,q,q')} \\ 2 & \text{if } v \in V_{\mathcal{B}}^{de} \end{cases}$$

When the extra bit b is equal to 1, this corresponds to the situation in which Spoiler encountered a final state that has not yet matched by Duplicator, while if b is equal to 0 she matched Spoiler's final states. The priority function assigns priority 1 to only those vertices in V_1 that signify that an unmatched final state has been visited by Spoiler. The priority function makes sure that that the smallest number occurring infinitely many often is 1 iff from some point onwards the bit in the first component is 1. Now observe that this bit is 1 iff a final state has been visited by Spoiler but not yet matched by Duplicator. In this way the winning condition of the delayed simulation game is transferred to the parity game. Similarly to the fair simulation parity game, it turns out that for each pair of states $q, q' \in Q$, $q \preceq^{de} q'$ iff Even has a winning strategy in $P(G_{\mathcal{B}}^{de}, v_{(q,q')})$. We precise that in [EWS05] they also define the *direct simulation parity game*.

We now describe the algorithm put forward in [Jur00] that solves parity games. Later, it has been improved in [EWS05]. It uses *progress measures* [Kla94; Wal00] to compute the set of vertices in a parity game from which Even has a winning strategy, that in our case will correspond the pairs of states in the fair or delayed simulation

relation. Let G be a parity game graph with n' its number of vertices and m' its number of edges. For simulations we only need three priorities, but since the algorithm is more general we will assume $p : V \to \{0, \ldots, d-1\}$.

Let $[n] = \{0, \ldots, n-1\}$ and $n_i = |p^{-1}(i)|$. We assign to each vertex a *progress measure* from $M_G^\infty \triangleq M_G \cup \{\infty\}$, where

$$M_G \triangleq \begin{cases} [1] \times [n_1 + 1] \times [1] \times [n_3 + 1] \times \cdots \times [1] \times [n_{d-1}1] & \text{if } d \text{ is even} \\ [1] \times [n_1 + 1] \times [1] \times [n_3 + 1] \times \cdots \times [1] \times [n_{d-2}1] & \text{if } d \text{ is odd} \end{cases}$$

That is, the progress measure is either ∞ or a length d vector which at even index positions is 0, and at odd index positions i ranges over $\{0, \ldots, n_i\}$. Initially, every vertex is assigned 0, the all-zero vector. The measures are repeatedly "incremented" in a certain fashion until a fixed point is reached. The *increment operation* represents the heart of the algorithm. For $i < d$ and $x \in M_G^\infty$ we define $\langle x \rangle_i$ as follows. For $x = (l_0, \ldots, l_{d-1})$, $\langle x \rangle_i = (l_0, \ldots, l_i, 0, 0, \ldots, 0)$. If $x = \infty$, then $\langle \infty \rangle_i = \infty$. We define a lexicographic total order on M_G^∞, denoted $>$. Here, index 0 is the most significant position, and ∞ is greater than all other vectors. In addition, for d-vectors x and y, we write $x >_i y$ iff $\langle x \rangle_i > \langle y \rangle_i$ according to the above ordering. Observe that $x > y$ iff $x >_{d-1} y$. We now can define the increment operation on a progress measure: for each $i \in [d]$, let

$$\mathit{incr}_i(x) \triangleq \begin{cases} \langle x \rangle_i & \text{if } i \text{ is even, } x \neq \infty \\ \mathit{min}\{y \in M_G^\infty \mid y >_i x\} & \text{if } i \text{ is odd, } x \neq \infty \\ \infty & \text{if } x = \infty \end{cases}$$

We observe that incr_i is monotone with respect to the ordering $<$. When $v \in V$ we abuse notation and write $\langle x \rangle_v$ and $\mathit{incr}_v(x)$ for $\langle x \rangle_{p(v)}$ and $\mathit{incr}_{p(v)}$ respectively. For every assignment $\rho : V \to M_G^\infty$ of progress measures to the vertices of a game graph, and for $v \in V$, let

$$\mathit{best\text{-}nghb\text{-}ms}(\rho, v) \triangleq \begin{cases} \langle \mathit{min}(\langle \rho(w) \mid (v, w) \in E \rangle) \rangle_v & \text{if } v \in V_0 \\ \langle \mathit{max}(\langle \rho(w) \mid (v, w) \in E \rangle) \rangle_v & \text{if } v \in V_1 \end{cases}$$

best-nghb-ms stand for *best neighbour* of v with respect to the measure we defined. We also define a *lifting operator* that given an assignment ρ and $v \in V$, gives a new assignment. First, define:

$$\mathit{update}(\rho, v) \triangleq \mathit{incr}_v(\mathit{best\text{-}nghb\text{-}ms}(\rho, v))$$

The lifted assignment, $\mathit{lift}(\rho, v) : V \to D$, is then defined as follows:

$$\mathit{lift}(\rho, v)(u) \triangleq \begin{cases} \mathit{update}(\rho, v) & \text{if } u = v \\ \rho(u) & \text{otherwise} \end{cases}$$

Then, we can finally present the algorithm described in [Jur00]. First, for each $v \in V$ we assign to $\rho(v)$ the value 0. Then, while there exists a v such that $\mathit{update}(\rho, v) \neq \rho(v)$, assign to ρ the value $\mathit{lift}(\rho, v)$. Let G be a parity game. In [Jur00] they prove that Even has a winning strategy from precisely the vertices v such that, after running the lifting algorithm, $\rho(v) < \infty$.

In [EWS05] they present a more efficient version of the lifting algorithm that maintains a set L of pending vertices v whose measure needs to be considered for lifting, because a successor has recently been updated resulting in a requirement to update $\rho(v)$. The also maintain arrays B and C that store, for each vertex v, the value $best\text{-}nghb\text{-}ms(\rho, v)$ and the number of successors u of v with $\langle \rho(u) \rangle_v = best\text{-}nghb\text{-}ms(\rho, v)$. They show that the efficient implementation runs in $O(m'|M_G^\infty|d)$ time and uses $O(d'm)$ space. Going back to the original problem, that is computing the fair and the delayed simulation, in [EWS05] they show that it is possible to compute the quasiorders in $O(|\delta||Q|^3)$ time and $O(|\delta||Q|)$ space.

As a final note, we point out that there are many tools to compute various simulations, for example RABIT [CM] is a platform that can check the language inclusion between two Büchi automata that heavily relies on simulation relations [Abd+11]. Oink [Dij18] is a modern tool that allows to solve parity games. In particular, it aims to provide high-performance implementations of state-of-the-art algorithms representing different approaches to solving parity games. Since we showed that simulations can be characterized as parity games, Oink can be used to efficiently compute them.

2.5 Solving the language inclusion problem with complete abstractions

In this section we first give an informal introduction of the main ideas on how to solve the language inclusion using *complete abstractions* for regular languages; in Section 2.5.1 we describe a framework for solving the language inclusion problem for ω-regular languages and finally in Section 2.5.2 we describe a framework for solving the language inclusion problem between context-free and regular languages.

We now give a glimpse of the leading ideas that exploit abstract interpretation techniques to design algorithms to solve the inclusion between two regular languages. For a formal description refer to [GRV19]. We remark that the reader will find the same concepts readapted in the following sections.

Let L_1, L_2 be two *regular* languages: we want to determine whether $L_1 \subseteq L_2$ holds or not. In order to do this we rely on one *upper closure operator* ρ (see Section 2.3). Intuitively, we use ρ to *abstract* the language L_1. In fact, $\rho(L_1)$ is an *over approximation* of L_1. Assume that $\rho(L_2) = L_2$: we observe that in this case $L_1 \subseteq L_2 \iff \rho(L_1) \subseteq L_2$.

Let $\mathcal{A} = \langle Q, \Sigma, \delta, I, F \rangle$ be one FA such that $\mathcal{L}(\mathcal{A}) = L_1$. It is well-known (and we will return on one similar concept in Section 2.5.1) that the language recognized by $\mathcal{A}$ can be characterized as follows: $\mathcal{L}(\mathcal{A})$ equals the union of the component languages of the vector $\mathrm{lfp}(\lambda \vec{X}.\vec{\epsilon^F} \cup Pre_{\mathcal{A}}(\vec{X}))$ indexed by the initial states in I, where $Pre_{\mathcal{A}} : \wp(\Sigma^*)^{|Q|} \to \wp(\Sigma^*)^{|Q|}$ is defined by $Pre_{\mathcal{A}}(\vec{X}) \triangleq \langle \bigcup_{a \in \Sigma, q \xrightarrow{a} q'} aX_{q'} \rangle_{q \in Q}$, $\vec{\epsilon^F} \triangleq \langle \{\epsilon \mid q \in F\} \rangle_{q \in Q} \in \wp(\Sigma^*)^{|Q|}$. In Section 2.5.1 we give an alternative characterization of the language recognized by an automaton which is based on a symmetric function called $Post_{\mathcal{A}}$. If ρ is *complete* for $Pre_{\mathcal{A}}$ (i.e. $\rho \circ Pre_{\mathcal{A}} \circ \rho = \rho \circ Pre_{\mathcal{A}}$), we can characterize $\rho(L_1)$ as the union of the component languages of the vector $\mathrm{lfp}(\lambda \vec{X}.\rho(\vec{\epsilon^F} \cup Pre_{\mathcal{A}}(\vec{X})))$ indexed by initial states in I. Intuitively, *completeness* models an ideal situation where no loss of precision is accumulated in the computations of $\rho \circ Pre_{\mathcal{A}}$ when its concrete input objects are approximated by ρ. Then, the requirements for ρ are:

1. $\rho(L_2) = L_2$;
2. ρ *complete* for $Pre_{\mathcal{A}}$.

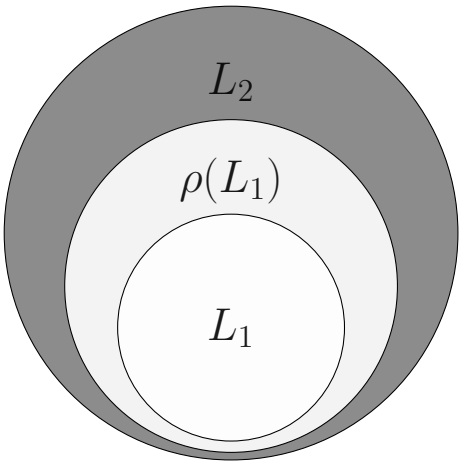

Figure 2.13: Solving the language inclusion using one abstraction ρ

We remark that given a qoset $\langle D, \leq \rangle$, we defined the upper closure operator on D induced by $\leq$ as $\rho_{\leq}(X) \stackrel{\triangle}{=} \{y \in D \mid \exists x \in X, x \leq y\}$. Let $\leq$ be a quasiorder on Σ^*. It turns out that if $\leq$ is a left L_2-consistent qo, then $\rho_{\leq}$ meets the previously stated requirements. Then, if this holds, we have:

$$L_1 \subseteq L_2 \iff \rho_{\leq}(L_1) \subseteq L_2$$

We denote by $min_{\leq}(X)$ the minor set induced by $\leq$. The key observation is that since minor sets are finite, $min_{\leq}(L_1)$ provides a compact representation of $\rho_{\leq}(L_1)$: it holds that $\rho_{\leq}(L_1) \subseteq L_2 \iff \forall w \in min_{\leq}(L_1), w \in L_2$, so that:

$$L_1 \subseteq L_2 \iff \forall w \in min_{\leq}(L_1), w \in L_2$$

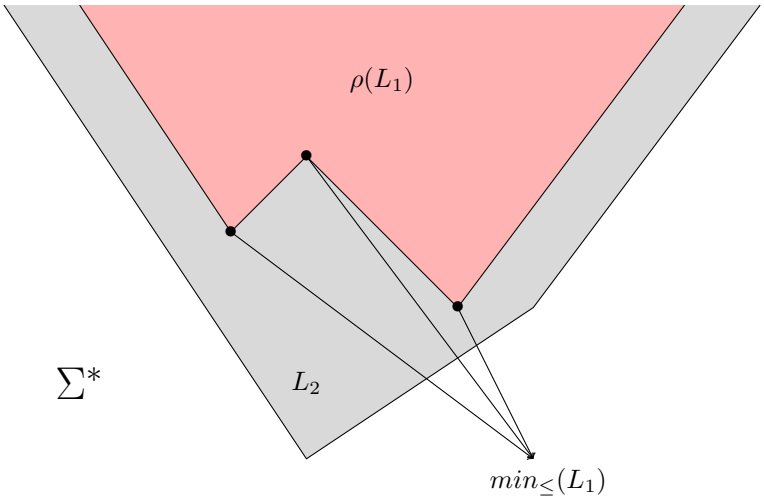

Figure 2.14: Representation of $L_2, \rho(L_1)$ and $min_{\leq}(L_1)$ in the qoset $\langle \Sigma^*, \leq \rangle$

The final remark is that it is possible to compute $min_{\leq}(L_1)$ by using the `Kleene` procedure to compute $\text{lfp}(\lambda \vec{X}.min_{\leq}(\vec{\epsilon^F} \cup Pre_{\mathcal{A}}(\vec{X})))$. Exploiting *complete abstractions*

we have shown how to solve the language inclusion problem: no *false alarms* will be raised.

This informally summarizes the main ideas behind solving the language inclusion for regular languages using complete abstractions. These concepts have been generalized and adapted to solve similar problems, namely checking the inclusion between ω-regular languages and whether the language of a CFG is contained in the one generated by a FA. In what follows, we describe more formally how these inclusions are decided.

2.5.1 Checking the inclusion between ω-regular languages

Let $\mathcal{B}_1, \mathcal{B}_2$ be two Büchi automata. To solve the *language inclusion problem*, namely whether $\mathcal{L}(\mathcal{B}_1) \subseteq \mathcal{L}(\mathcal{B}_2)$ holds or not, numerous algorithms have been proposed [Abd+11; CM17]. In this section we're going to describe the framework put forward by [DG20], which relies on *abstract interpretation* techniques.

One word $w \in \Sigma^\omega$ is *ultimately periodic* iff $w = uv^\omega$ for some $u \in \Sigma^*, v \in \Sigma^+$. For $L \subseteq \Sigma^\omega$, $UP(L)$ denotes its subset of ultimately periodic words. Let L_1, L_2 be two ω-regular languages [CNP93]:

$$L_1 \subseteq L_2 \iff UP(L_1) \subseteq UP(L_2)$$

We can represent $UP(L)$ by $I_L \triangleq \{(u, v) \in \Sigma^* \times \Sigma^+ \mid uv^\omega \in L\}$, so that:

$$UP(L_1) \subseteq UP(L_2) \iff I_{L_1} \subseteq I_{L_2}$$

Now we abstract I_{L_1} using a wqo $\preceq$ on $\Sigma^* \times \Sigma^+$ such that $\rho_\preceq(I_{L_2}) = I_{L_2}$.

$$I_{L_1} \subseteq I_{L_2} \iff \rho_\preceq(I_{L_1}) \subseteq I_{L_2}$$

We define $(u, v) \equiv_{UP} (u', v')$ iff $uv^\omega = u'v'^\omega$, where $(u, v), (u', v') \in \Sigma^* \times \Sigma^+$. Since $\equiv_{UP}$ is a qo on $\Sigma^* \times \Sigma^+$, the closure operator $\rho_{\equiv_{UP}}$ is well-defined. Let $S \subseteq \Sigma^* \times \Sigma^+$ such that $\rho_{\equiv_{UP}}(S) = I_{L_1}$. Then, $I_{L_1} \subseteq I_{L_2} \iff S \subseteq I_{L_2}$ [DG20]. Since $\rho_\preceq(I_{L_2}) = I_{L_2}$ and by *monotonicity* of $\rho_\preceq$ we obtain:

$$I_{L_1} \subseteq I_{L_2} \iff \rho_\preceq(S) \subseteq I_{L_2}$$

We have to choose a suitable set S. In order to do this, we rely on the automaton representation of the language L_1. Let $\mathcal{B} = \langle Q, \Sigma, \delta, \{i\}, F\rangle$ be a BA with $\mathcal{L}(\mathcal{B}) = L_1$. Observe that in [DG20] they consider automata with only one initial state. We remark that it is always possible to construct from a BA $\mathcal{B} = \langle Q, \Sigma, \delta, I, F\rangle$ a BA $\mathcal{B}' = \langle Q \cup \{i\}, \Sigma, \delta \cup \delta', \{i\}, F\rangle$ that has one single initial state and accepts the same language of $\mathcal{B}$: it is enough to add one new state $i \notin Q$, to pick it as the only initial state and to add to δ the new set of transitions $\delta' = \{(i, q) \mid \exists p \in I, q \in Q, a \in \Sigma : q \in \delta(p, a)\}$. For each $p, q \in Q$ we define the FA $\mathfrak{A}_q^p \triangleq \langle Q, \Sigma, \delta, \{p\}, \{q\}\rangle$, and define $S_\mathcal{B} \triangleq \bigcup_{p \in F} \mathcal{L}(\mathfrak{A}_p^i) \times (\mathcal{L}(\mathfrak{A}_p^p) \setminus \{\epsilon\})$. Since $\rho_{\equiv_{UP}}(S_\mathcal{B}) = I_{\mathcal{L}(\mathcal{B})}$ [DG20], by the previous result:

$$I_{\mathcal{L}(\mathcal{B})} \subseteq I_{L_2} \iff \rho_\preceq(S_\mathcal{B}) \subseteq I_{L_2}$$

We now give a *least fixpoint characterization* of $S_\mathcal{B}$. For all $p \in F$, the languages of the FAs $\mathfrak{A}_p^i$ and $\mathfrak{A}_p^p$ can be characterized by $\mathcal{L}(\mathfrak{A}_p^i) = \langle \text{lfp } \lambda\vec{X}.\vec{\epsilon^i} \cup Post_\mathcal{B}(\vec{X})\rangle_p$ and $\mathcal{L}(\mathfrak{A}_p^p) \setminus \{\epsilon\} = \langle \text{lfp } \lambda\vec{X}.\vec{\zeta^p} \cup Post_\mathcal{B}(\vec{X})\rangle_p$ [GRV19], where $Post_\mathcal{B} : \wp(\Sigma^*)^{|Q|} \to \wp(\Sigma^*)^{|Q|}$

is defined by $Post_{\mathcal{B}}(\vec{X}) \triangleq \langle \bigcup_{a\in\Sigma, q' \xrightarrow{a} q} X_{q'} a\rangle_{q\in Q}$, $\vec{\epsilon^i} \triangleq \langle \{\epsilon \mid q = i\}\rangle_{q\in Q} \in \wp(\Sigma^*)^{|Q|}$, and $\vec{\zeta^p} \triangleq \langle \{a \in \Sigma \mid q \in \delta(p, a)\}\rangle_{q\in Q} \in \wp(\Sigma^*)^{|Q|}$. Letting $D_{1,i} \triangleq \lambda\vec{X}.\vec{\epsilon^i} \cup Post_{\mathcal{B}}(\vec{X})$ and $D_{2,p} \triangleq \lambda\vec{X}.\vec{\zeta^p} \cup Post_{\mathcal{B}}(\vec{X})$ we obtain that:

$$S_{\mathcal{B}} = \bigcup_{p\in F} (\text{lfp } D_{1,i})_p \times (\text{lfp } D_{2,p})_p$$

For each $p \in F$, define the vector $\vec{I}^p_{L_2} \in \wp(\Sigma^* \times \Sigma^+)$ as $(\vec{I}^p_{L_2})_q \triangleq \Sigma^* \times \Sigma^+$ for $q \neq p$ and $(\vec{I}^p_{L_2})_p \triangleq I_{L_2}$. Lift $\times$ to vectors as follows: $\vec{X} \times \vec{Y}$ is the vector $\langle X_q \times Y_q\rangle_{q\in Q}$ where $\vec{X} \triangleq \langle X_q\rangle_{q\in Q}$ and $\vec{Y} \triangleq \langle Y_q\rangle_{q\in Q}$. Let $\leq_1 \subseteq \Sigma^* \times \Sigma^*$ and $\leq_2 \subseteq \Sigma^+ \times \Sigma^+$ two wqos such that $\preceq = \leq_1 \times \leq_2$. This equality can be lifted to vectors. Using this we obtain:

$$\rho_{\preceq}(S_{\mathcal{B}}) \subseteq I_{L_2} \iff \forall p \in F, \rho_{\leq_1}(\text{lfp } D_{1,i}) \times \rho_{\leq_2}(\text{lfp } D_{2,p}) \subseteq \vec{I}^p_{L_2}$$

Observe that the least fixed points have no guarantee to converge after finitely many steps. We solve this issue by "pushing" the closure operator ρ inside the fixed point expression, and this induces no loss of precision. This is the consequence of a property of ρ relative to the function of the fixed point expression. A closure ρ is called *backward complete*, or simply *complete*, for a function $f : C \to C$ when $\rho \circ f = \rho \circ f \circ \rho$. Suppose that $\rho_{\leq_1}, \rho_{\leq_2}$ are backward complete for $D_{1,i}$ and $D_{2,p}$ respectively. It implies completeness of least fixed points [CC79]: $\rho_{\leq_1}(\text{lfp } D_{1,i}) = \text{lfp } \rho_{\leq_1} \circ D_{1,i}$ and $\rho_{\leq_2}(\text{lfp } D_{2,p}) = \text{lfp } \rho_{\leq_2} \circ D_{2,p}$. Furthermore, it turns out that if one qo $\leq$ on Σ^* is right-monotonic, then it is also backward complete for $D_{1,i}$ and $D_{2,p}$ [DG20]. This implies that if $\leq_1$ and $\leq_2$ are two right-monotonic wqos:

$$\forall p \in F, \rho_{\leq_1}(\text{lfp } D_{1,i}) \times \rho_{\leq_2}(\text{lfp } D_{2,p}) \subseteq \vec{I}^p_{L_2} \iff \forall p \in F, \text{lfp } \rho_{\leq_1} D_{1,i} \times \text{lfp } \rho_{\leq_2} D_{2,p} \subseteq \vec{I}^p_{L_2}$$

This solves the convergence problem. For simplicity ignore that we are working on vectors by assuming $D_{1,i} : \wp(\Sigma^*) \to \wp(\Sigma^*)$. Observe that since $\leq_1$ is a wqo, $\langle \{\rho_{\leq_1}(S) \mid S \in \wp(\Sigma^*)\}, \subseteq\rangle$ is a complete sublattice verifying the ACC and, moreover, $\rho_{\leq_1} \circ D_{1,i}$ is a monotone function, hence it is routine to check that $\rho_{\leq_1} \circ D_{1,i}$ preserves $\cup$, and finally that $\text{lfp } \rho_{\leq_1} \circ D_{1,i} = \bigcup_n (\rho_{\leq_1} \circ D_{1,i})^n(\emptyset)$ for some $n \in \mathbb{N}$ by the Kleene's iterative fixed point theorem. The same results also hold when considering $\leq_2$ and $D_{2,p}$.

We define $\gamma_{\leq_1} : \wp_f(\Sigma^*) \to \wp(\Sigma^*)$ such that $\gamma_{\leq_1}(Y) \triangleq \rho_{\leq_1}(Y)$, which maps finite sets of words onto upward-closed set. Backward completeness shows that $\rho_{\leq_1} \circ D_{1,i} \circ \gamma_{\leq_1} = \gamma_{\leq_1} \circ D_{1,i}$. By induction we find that $(\rho_{\leq_1} \circ D_{1,i})^n(\emptyset) = \gamma_{\leq_1}(D^n_{1,i}(\emptyset))$ for all $n \in \mathbb{N}$, hence that $\text{lfp } \rho_{\leq_1} D_{1,i} = \gamma_{\leq_1}(D^N_{1,i}(\emptyset))$ for some $N \in \mathbb{N}$ from the convergence result on upward-closed sets previously stated. Observe that $D_{1,i}$ returns a finite set of words when given one, that is, $D_{1,i} : \wp_f(\Sigma^*) \to \wp_f(\Sigma^*)$. By induction we thus find that $D^n_{1,i}(\emptyset)$ is finite and effectively computable for all $n \geq 0$. The above reasoning also holds for $\leq_2$ and $D_{2,p}$. Let $N_1, N_2 \in \mathbb{N}$ be such that $(\gamma_{\leq_1}(D^{N_1+1}_{1,i}(\emptyset)))_q \subseteq (\gamma_{\leq_1}(D^{N_1}_{1,i}(\emptyset)))_q$ and $(\gamma_{\leq_2}(D^{N_2+1}_{2,p}(\emptyset)))_q \subseteq (\gamma_{\leq_2}(D^{N_2}_{2,p}(\emptyset)))_q$ for all $q \in Q$. Then, we obtain:

$$\forall p \in F, \text{lfp } \rho_{\leq_1} D_{1,i} \times \text{lfp } \rho_{\leq_2} D_{2,p} \subseteq \vec{I}^p_{L_2} \iff \forall p \in F, \gamma_{\leq_1}(D^{N_1}_{1,i}(\emptyset)) \times \gamma_{\leq_2}(D^{N_2}_{2,p}(\emptyset)) \subseteq \vec{I}^p_{L_2}$$

We still have to show how to effectively detect when the sequence $\{D^n_{1,i}(\emptyset)\}_{n\in\mathbb{N}}$ has converged, that is when $(\gamma_{\leq_1}(D^{n+1}_{1,i}(\emptyset)))_q \subseteq (\gamma_{\leq_1}(D^n_{1,i}(\emptyset)))_q$ for all $q \in Q$. To this end

define the qos $\leq_1^{\forall\exists}$ on $\wp_f(\Sigma^*)$ such that $X \leq_1^{\forall\exists} Y \overset{\triangle}{\iff} \gamma_{\leq_1}(X) \subseteq \gamma_{\leq_1}(Y)$. Assuming that the ordering $\leq_1$ is decidable, we find that given $X, Y \in \wp_f(\Sigma^*)$ we can decide $X \leq_1^{\forall\exists} Y$ by finding for each element $x \in X$ an element $y \in Y$ such that $y \leq_1 x$. The above reasoning also holds for $\leq_2$ and $D_{2,p}$. Finally, the lifting to vectors is an easy exercise. Let $N_1, N_2 \in \mathbb{N}$ be such that $(D_{1,i}^{N_1+1}(\emptyset))_q \leq_1^{\forall\exists} (D_{1,i}^{N_1}(\emptyset))_q$ and $(D_{2,p}^{N_2+1}(\emptyset))_q \leq_2^{\forall\exists} (D_{2,p}^{N_2}(\emptyset))_q$ for all $q \in Q$ and $p \in F$. Given $p \in F$, let $\vec{T}_p \overset{\triangle}{=} D_{1,i}^{N_1}(\emptyset) \times D_{2,p}^{N_2}(\emptyset)$. Observe that $\vec{T}_p$ is an effectively computable element $(\wp_f(\Sigma^*) \times \wp_f(\Sigma^*))^{|Q|}$. We have:

$$\forall p \in F, \gamma_{\leq_1}(D_{1,i}^{N_1}(\emptyset)) \times \gamma_{\leq_2}(D_{2,p}^{N_2}(\emptyset)) \subseteq \vec{I}_{L_2}^p \iff \forall p \in F, \rho_{\preceq}(\vec{T}_p) \subseteq \vec{I}_{L_2}^p$$

Finally, since $\rho_{\preceq}$ is increasing and $\rho_{\preceq}(I_{L_2}) = I_{L_2}$:

$$\forall p \in F, \rho_{\preceq}(\vec{T}_p) \subseteq \vec{I}_{L_2}^p \iff \forall p \in F, \vec{T}_p \subseteq \vec{I}_{L_2}^p$$

Observing that each element of the vector $\vec{T}_p$ is finite, we can reduce inclusion to one finite number of membership queries:

$$\forall p \in F, \vec{T}_p \subseteq \vec{I}_{L_2}^p \iff \forall p \in F, \forall (u,v) \in (\vec{T}_p)_p, uv^\omega \in L_2$$

Summing up everything, we have:

$$\mathcal{L}(\mathcal{B}) \subseteq L_2 \iff \forall p \in F, \forall (u,v) \in (\vec{T}_p)_p, uv^\omega \in L_2$$

Algorithms `BAPrefixes` and `BAPeriods` compute respectively $D_{1,i}^{N_1}(\emptyset)$ and $D_{2,p}^{N_2}(\emptyset)$ where $N_1, N_2 \in \mathbb{N}$ are the least values such that $(D_{1,i}^{N_1+1}(\emptyset))_q \leq_1^{\forall\exists} (D_{1,i}^{N_1}(\emptyset))_q$ and $(D_{2,p}^{N_2+1}(\emptyset))_q \leq_2^{\forall\exists} (D_{2,p}^{N_2}(\emptyset))_q$ for all $q \in Q$ and $p \in F$.

We slightly abuse the notation calling `BAPrefixes`: by `BAPrefixes(`$\mathcal{B}$`,`$\leq_1$`)` we mean to invoke the procedure `BAPrefixes` with actual parameters the automaton $\mathcal{B}$ and *the procedure* to compute $\leq_1$, and not $\leq_1$ itself because it is an infinite relation. The same holds for `BAPeriods` and $\leq_2$. Finally, Algorithm `BAInc` computes whether $\mathcal{L}(\mathcal{B}) \subseteq L_2$ holds or not.

Discussion. We presented a framework for solving the language inclusion problem for ω-regular languages which relies on the underlying automaton representation of the language. Let $\leq_1$ and $\leq_2$ be two quasiorders on Σ^*. If the following conditions hold:

1. $\leq_1$ and $\leq_2$ are computable well-quasiorders;
2. $\leq_1$ and $\leq_2$ are right-monotonic;
3. $\rho_{\leq_1 \times \leq_2}(I_{L_2}) = I_{L_2}$.

Then, $\leq_1$ and $\leq_2$ can be used in Algorithm `BAInc` to solve the language inclusion problem. Observe that condition (3) can be characterized alternatively as follows:

Proposition 2.5.1. $\rho_{\leq_1 \times \leq_2}(I_{L_2}) = I_{L_2}$ iff $\forall u, s \in \Sigma^*, v, t \in \Sigma^+$ such that $uv^\omega \in L_2$, $u \leq_1 s$ and $v \leq_2 t$, it holds that $st^\omega \in L_2$.

Proof. If $\rho_{\leq_1 \times \leq_2}(I_{L_2}) = I_{L_2}$, then consider $u, s \in \Sigma^*, v, t \in \Sigma^+$ such that $uv \in L_2$, $u \leq_1 s$ and $v \leq_2 t$. By definition of ρ, $s \in \rho_{\leq_1}(u)$ and $t \in \rho_{\leq_2}(v)$, so that $(s,t) \in \rho_{\leq_1 \times \leq_2}(I_{L_2})$ and by hypothesis $\rho_{\leq_1 \times \leq_2}(I_{L_2}) = I_{L_2}$. Since $I_{L_2} = \{(w_1, w_2) \mid w_1 w_2^\omega \in L_2\}$, $st^\omega \in L_2$.

For the other implication we proceed by showing: (i) $\rho_{\leq_1 \times \leq_2}(I_{L_2}) \subseteq I_{L_2}$; (ii) $I_{L_2} \subseteq \rho_{\leq_1 \times \leq_2}(I_{L_2})$. Let $(s,t) \in \rho_{\leq_1 \times \leq_2}(I_{L_2})$, then $\exists (u,v) \in I_{L_2}$ such that $u \leq_1 s$

BAPrefixes: Algorithm that computes $D_{1,i}^{N_1}(\emptyset)$, where $N_1 \in \mathbb{N}$ is the least value such that $(D_{1,i}^{N_1+1}(\emptyset))_q \leq_1^{\forall\exists} (D_{1,i}^{N_1}(\emptyset))_q$ for all $q \in Q$

Data: Büchi automaton $\mathcal{B} = \langle Q, \Sigma, \delta, \{i\}, F\rangle$
Data: Procedure deciding $u \leq_1 v$, given $u, v \in \Sigma^*$
Result: $D_{1,i}^{N_1}(\emptyset)$, where $N_1 \in \mathbb{N}$ is the least value such that $(D_{1,i}^{N_1+1}(\emptyset))_q \leq_1^{\forall\exists} (D_{1,i}^{N_1}(\emptyset))_q$ for all $q \in Q$

```
1  Prev ← ⟨∅⟩_{q∈Q} ;
2  done ← false ;
3  while done = false do
4  |  done ← true ;
5  |  Next ← D_{1,i}(Prev) ;
6  |  forall q ∈ Q do
7  |  |  forall u ∈ (Next)_q do
8  |  |  |  if ∄v ∈ (Prec)_q such that v ≤_1 u then
9  |  |  |  |  done ← false;
10 |  |  |  end
11 |  |  end
12 |  end
13 |  if done = false then
14 |  |  Prev ← Next;
15 |  end
16 end
17 return Prev
```

and $v \leq_2 t$. Then, by hypothesis, $st^\omega \in L_2$, so that $(s,t) \in I_{L_2}$. Let $(s,t) \in I_{L_2}$. We observe that by reflexivity of $\leq_1$ and $\leq_2$ it holds that $\forall w_1 \in \Sigma^*, w_1 \in \rho_{\leq_1}(w_1)$ and $\forall w_2 \in \Sigma^+, w_2 \in \rho_{\leq_2}(w_2)$. This implies that $(s,t) \in \rho_{\leq_1 \times \leq_2}(I_{L_2})$. □

Observe that the performance of Algorithm `BAInc` depends on the choice of the two wqos $\leq_1$ and $\leq_2$. Let $\preceq_1 \subseteq \leq_1$, then `BAPrefixes`$(\mathcal{B}, \leq_1)$ will converge in less iterations than `BAPrefixes`$(\mathcal{B}, \preceq_1)$, because the condition at line 8 will result false more often. The same holds for `BAPeriods`. This implies that we want to consider coarser relations in order to let the algorithm converge faster.

Definition 2.5.2. Let $\leq_1, \leq_2$ a pair of qos on Σ^*. We say that the pair $\leq_1, \leq_2$ *covers* an ω-regular language L iff:

$$L = \bigcup_{uv^\omega \in L} \rho_{\leq_1}(u)\rho_{\leq_2}(v)^\omega$$

In [DG20] they observe that if $\leq_1, \leq_2$ covers one language, then $\rho_{\leq_1 \times \leq_2}(I_{L_2}) = I_{L_2}$. They also propose some wqos that meet the requirements of the framework. First, we define the *state-based* wqos. Let $u, v \in \Sigma^*$ and $\mathcal{B}$ be a BA.

$$u \leq_{\mathcal{B}}^1 v \overset{\triangle}{\iff} ctx_{\mathcal{B}}(u) \subseteq ctx_{\mathcal{B}}(v)$$

$$u \leq_{\mathcal{B}}^2 v \overset{\triangle}{\iff} u \leq_{\mathcal{B}}^1 v \ \wedge \ ctx_{\mathcal{B}}^F(u) \subseteq ctx_{\mathcal{B}}^F(v)$$

BAPeriods: Algorithm that computes $D_{2,p}^{N_2}(\emptyset)$, where $N_2 \in \mathbb{N}$ is the least value such that $(D_{2,p}^{N_2+1}(\emptyset))_q \leq_2^{\forall\exists} (D_{2,p}^{N_2}(\emptyset))_q$ for all $q \in Q$

Data: Büchi automaton $\mathcal{B} = \langle Q, \Sigma, \delta, \{i\}, F\rangle$
Data: Procedure deciding $u \leq_2 v$, given $u, v \in \Sigma^+$
Data: Final state $p \in F$
Result: $D_{2,p}^{N_2}(\emptyset)$, where $N_2 \in \mathbb{N}$ is the least value such that $(D_{2,p}^{N_2+1}(\emptyset))_q \leq_2^{\forall\exists} (D_{2,p}^{N_2}(\emptyset))_q$ for all $q \in Q$

```
Prev ← ⟨∅⟩_{q∈Q} ;
done ← false ;
while done = false do
    done ← true ;
    Next ← D_{2,p}(Prev) ;
    forall q ∈ Q do
        forall u ∈ (Next)_q do
            if ∄v ∈ (Prec)_q such that v ≤_2 u then
                done ← false;
            end
        end
    end
    if done = false then
        Prev ← Next;
    end
end
return Prev
```

$$u \leq_{\mathcal{B}}^{r} v \overset{\triangle}{\iff} post_u^{\mathcal{B}}(I) \subseteq post_v^{\mathcal{B}}(I)$$

Observe that $\leq_{\mathcal{B}}^{1}$ has been already defined in Section 2.3.2. It holds that $\leq_{\mathcal{B}}^{1}$ and $\leq_{\mathcal{B}}^{2}$ are monotonic, while $\leq_{\mathcal{B}}^{r}$ is right-monotonic. Furthermore, the pairs $\leq_{\mathcal{B}}^{1}, \leq_{\mathcal{B}}^{2}$ and $\leq_{\mathcal{B}}^{r}, \leq_{\mathcal{B}}^{2}$ cover $\mathcal{L}(\mathcal{B})$, so that they can be used to instantiate the Algorithm `BAInc`.

Next, we define the *syntactic quasiorders*. Let $L \subseteq \Sigma^\omega$. For $u \in \Sigma^*$ define $c_L \overset{\triangle}{=} \{(x, y, r) \in \Sigma^{*3} \mid xuyr^\omega \in L\}$, $d_L \overset{\triangle}{=} \{(s, t) \in \Sigma^{*2} \mid s(ut)^\omega \in L\}$ and $e_L \overset{\triangle}{=} \{(s, t) \in \Sigma^{*2} \mid ust^\omega \in L\}$. We define:

$$u \leq_L^1 v \overset{\triangle}{\iff} c_L(u) \subseteq c_L(v)$$

$$u \leq_L^2 v \overset{\triangle}{\iff} c_L(u) \subseteq c_L(v) \ \wedge \ d_L(u) \subseteq d_L(v)$$

$$u \leq_L^r v \overset{\triangle}{\iff} e_L(u) \subseteq e_L(v)$$

It holds that $\leq_L^1$ and $\leq_L^2$ are monotonic, while $\leq_L^r$ is right-monotonic. They are computable wqos, and the pairs $\leq_L^1, \leq_L^2$ and $\leq_L^r, \leq_L^2$ cover L, so that they can be used to instantiate the Algorithm `BAInc`. Let $\mathcal{B}$ be a BA such that $L = \mathcal{L}(\mathcal{B})$. It holds that $\leq_{\mathcal{B}}^1 \subseteq \leq_L^1$, $\leq_{\mathcal{B}}^2 \subseteq \leq_L^2$ and $\leq_{\mathcal{B}}^r \subseteq \leq_L^r$. Observe that furthermore in [DG20] they also prove the following:

Proposition 2.5.3. Let $\leq_1, \leq_2$ be a pair of qos covering L such that $\leq_2 \subseteq \leq_1$.

BAInc: Algorithm that computes whether $\mathcal{L}(\mathcal{B}) \subseteq L_2$ holds

Data: Büchi automaton $\mathcal{B} = \langle Q, \Sigma, \delta, \{i\}, F\rangle$
Data: Procedure deciding $uv^\omega \in L_2$ given $u \in \Sigma^*, v \in \Sigma^+$
Data: Decidable right-monotonic wqos $\leq_1, \leq_2$ s.t. $\rho_{\leq_1 \times \leq_2}(I_{L_2}) = I_{L_2}$
Result: Whether $\mathcal{L}(\mathcal{B}) \subseteq L_2$ holds

```
1  Prefixes ← BAPrefixes(B,≤1);
2  forall p ∈ F do
3  |  Periods ← BAPeriods(B, ≤2, p) ;
4  |  forall u ∈ (Prefixes)_p, v ∈ (Periods)_p do
5  |  |  if uv^ω ∉ L2 then
6  |  |  |  return false
7  |  |  end
8  |  end
9  end
10 return true
```

- If $\leq_1, \leq_2$ are both monotonic, then $\leq_1 \subseteq \leq_L^1$ and $\leq_2 \subseteq \leq_L^2$;
- If $\leq_1$ is right-monotonic, and $\leq_2$ is monotonic then $\leq_1 \subseteq \leq_L^1$ and $\leq_2 \subseteq \leq_L^2$.

We observe the following:

Proposition 2.5.4. Let $\mathcal{B}$ be a BA. Let $\leq_1, \leq_2$ be a pair of qos such that $\forall u, s \in \Sigma^*, v, t \in \Sigma^+$, if $uv^\omega \in \mathcal{L}(\mathcal{B})$, $u \leq_1 s$ and $v \leq_2 t$, then $st^\omega \in \mathcal{L}(\mathcal{B})$. Then, $\leq_1, \leq_2$ *covers* $\mathcal{L}(\mathcal{B})$.

Proof. Let $L = \bigcup_{uv^\omega \in \mathcal{L}(\mathcal{B})} \rho_{\leq_1}(u)\rho_{\leq_2}(v)^\omega$. We have to show that $L = \mathcal{L}(\mathcal{B})$. That $\mathcal{L}(\mathcal{B}) \subseteq L$ holds is implied by the fact that $\leq_1$ and $\leq_2$ are qos, so that they are reflexive. That $L \subseteq \mathcal{L}(\mathcal{B})$ holds is implied by the hypothesis that $\forall uv^\omega \in \mathcal{L}(\mathcal{B})$, if $s \in \Sigma^*, t \in \Sigma^+$ are such that $u \leq_1 s$ and $v \leq_2 t$, then $st^\omega \in \mathcal{L}(\mathcal{B})$. This implies, by definition of L, that each element in L is also in $\mathcal{L}(\mathcal{B})$. □

In Chapter 3 we propose some suitable relations on words that can fit the described framework. We will show that defining quasiorders on words that are based on simulations we obtain relations that are coarser than the state-based ones, while being finer than the syntactic qos, effectively lying the in the middle between the two categories of preorders. In Chapter 4 we give some examples that show the practical advantage of using simulation-based qos.

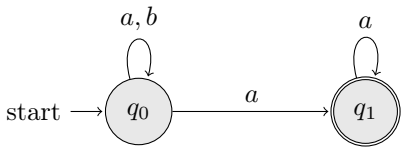

Figure 2.15: Büchi automaton that accepts the language $(a+b)^*a^\omega$

Example 2.5.5. We now give an example of a run of Algorithm BAInc. Let $\mathcal{B}$ be the BA in Figure 2.15 and $\mathcal{B}'$ be the BA in Figure 2.16. We want to decide whether

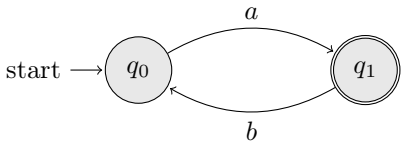

Figure 2.16: Büchi automaton that accepts the language $(ab)^\omega$

$\mathcal{L}(\mathcal{B}) \subseteq \mathcal{L}(\mathcal{B}')$ holds or not. For this example, we instantiate the algorithm with the state-based qos $\leq^1_{\mathcal{B}}, \leq^2_{\mathcal{B}}$.

The first step is to compute the vector $Prefixes$, calling the procedure `BAPrefixes`$(\mathcal{B}, \leq^1_{\mathcal{B}})$. Table 2.1 shows the first three Kleene's iterates of the function D_{1,q_0} relative to $\mathcal{B}$. We point out some of the relations between words based on $\leq^1_{\mathcal{B}}$. Observe that $ctx_{\mathcal{B}}(a) = \{(q_0, q_0), (q_0, q_1), (q_1, q_1)\}$ and $ctx_{\mathcal{B}}(b) = \{(q_0, q_0)\}$. Since $ctx_{\mathcal{B}}(aa) = \{(q_0, q_0), (q_0, q_1), (q_1, q_1)\}$, we observe that $ctx_{\mathcal{B}}(a) \subseteq ctx_{\mathcal{B}}(aa)$ so that $a \leq^1_{\mathcal{B}} aa$. Similarly, $ctx_{\mathcal{B}}(ab) = ctx_{\mathcal{B}}(bb) = \{(q_0, q_0)\}$, hence $b \leq^1_{\mathcal{B}} ab$ and $b \leq^1_{\mathcal{B}} bb$. Finally, $ctx_{\mathcal{B}}(ba) = \{(q_0, q_0), (q_0, q_1)\}$ and then $b \leq^1_{\mathcal{B}} ba$. Consider $(D^3_{1,q_0}(\emptyset))_{q_0}$. By the previous observations we can verify that $\forall u \in (D^3_{1,q_0}(\emptyset))_{q_0}$, $\exists v \in (D^2_{1,q_0}(\emptyset))_{q_0}$ such that $v \leq^1_{\mathcal{B}} u$. Furthermore, it also holds that $\forall u \in (D^3_{1,q_0}(\emptyset))_{q_1}$, $\exists v \in (D^2_{1,q_0}(\emptyset))_{q_1}$ such that $v \leq^1_{\mathcal{B}} u$. Hence, `BAPrefixes`$(\mathcal{B}, \leq^1_{\mathcal{B}})$ returns the vector $(\{a, b\}, \{a\})$.

	q_0	q_1
$D^0_{1,q_0}(\emptyset)$	$\emptyset$	$\emptyset$
$D^1_{1,q_0}(\emptyset)$	$\{\epsilon\}$	$\emptyset$
$D^2_{1,q_0}(\emptyset)$	$\{\epsilon, a, b\}$	$\{a\}$
$D^3_{1,q_0}(\emptyset)$	$\{\epsilon, a, b, aa, ab, ba, bb\}$	$\{a, aa\}$

Table 2.1: The first three Kleene's iterates of D_{1,q_0} on the automaton in Figure 2.15

The second step is to compute, for each final state, the vector $Periods$. Since there is just one final state, we now consider the only call to `BAPeriods`$(\mathcal{B}, \leq^2_{\mathcal{B}})$. Table 2.2 shows the first two Kleene's iterates of the function D_{2,q_1} relative to $\mathcal{B}$. We observe that $ctx^F_{\mathcal{B}}(a) = \{(q_0, q_1), (q_1, q_1)\}$ and $ctx^F_{\mathcal{B}}(aa) = \{(q_0, q_1), (q_1, q_1)\}$ so that $ctx^F_{\mathcal{B}}(a) \subseteq ctx^F_{\mathcal{B}}(aa)$ and $ctx_{\mathcal{B}}(a) \subseteq ctx_{\mathcal{B}}(aa)$. This implies $a \leq^2_{\mathcal{B}} aa$, so that $\forall u \in (D^2_{2,q_1}(\emptyset))_{q_1}$ $\exists v \in (D^1_{2,q_1}(\emptyset))_{q_1}$ such that $v \leq^2_{\mathcal{B}} u$. Hence, `BAPeriods`$(\mathcal{B}, \leq^2_{\mathcal{B}})$ returns the vector $(\emptyset, \{a\})$.

	q_0	q_1
$D^0_{2,q_1}(\emptyset)$	$\emptyset$	$\emptyset$
$D^1_{2,q_1}(\emptyset)$	$\emptyset$	$\{a\}$
$D^2_{2,q_1}(\emptyset)$	$\emptyset$	$\{a, aa\}$

Table 2.2: The first two Kleene's iterates of D_{2,q_1} on the automaton in Figure 2.15

The last step is to compute whether $\forall u \in (D^2_{1,q_0}(\emptyset))_{q_1} = \{a\}, v \in (D^1_{2,q_1}(\emptyset))_{q_1} = \{a\}$ it holds that $uv^\omega \in \mathcal{L}(\mathcal{B}')$. Since $a^\omega \notin \mathcal{L}(\mathcal{B}')$, the algorithm terminates and returns `false`: $\mathcal{L}(\mathcal{B}) \not\subseteq \mathcal{L}(\mathcal{B}')$.

2.5.2 Checking the language inclusion between regular and context-free languages

In this Section we describe the framework put forward in [GRV19] to check whether one context-free language L_1 is contained in a regular language L_2. We remark that our main interest is in the language inclusion between ω-regular languages, but it turns out that the qo that we define in Definition 3.1.1 can be plugged in this framework to solve whether the language of a context-free grammar is included in a regular one.

Let $\mathcal{G} = \langle \mathcal{V}, \Sigma, P \rangle$ be a CGF in CNF where $\mathcal{V} = \{X_0, \ldots, X_n\}$. First, observe that $\mathcal{G}$ induces the following set of equations:

$$Eqn(\mathcal{G}) \triangleq \{X_i = \cup_{X_1 \to \beta_j \in P} \beta_j \mid i \in [0, n]\}$$

Given a subset of variables $S \subseteq \mathcal{V}$ of a grammar, the set of words generated from some variable in S is defined as $W_S^{\mathcal{G}} \triangleq \{w \in \Sigma^* \mid \exists X \in S, X \rightsquigarrow w\}$. When $S = \{X\}$ we slightly abuse the notation and write $W_X^{\mathcal{G}}$. Observe that $\mathcal{L}(\mathcal{G}) = W_{X_0}^{\mathcal{G}}$.

In order to give a *least fixpoint characterization* of the language of $\mathcal{G}$ we define the vector $\vec{b} \in \wp(\Sigma^*)^{|\mathcal{V}|}$ and the function $Fn_{\mathcal{G}} : \wp(\Sigma^*)^{|\mathcal{V}|} \to \wp(\Sigma^*)^{|\mathcal{V}|}$:

$$\vec{b} \triangleq \langle b_i \rangle_{i \in [0,n]} \text{ where } b_i \triangleq \{\beta \mid X_i \to \beta \in P, \beta \in \Sigma \cup \{\epsilon\}\}$$

$$Fn_{\mathcal{G}}(\langle X_i \rangle_{i \in [0,n]}) \triangleq \langle \beta_1^{(i)} \cup \cdots \cup \beta_n^{(i)} \rangle_{i \in [0,n]} \text{ where } \beta_j^{(i)} \in \mathcal{V}^2 \text{ and } X_i \to \beta_j^{(i)} \in P$$

We remark that $\lambda \vec{X}.\ \vec{b} \cup Fn_{\mathcal{G}}(\vec{X})$ is a well-defined monotonic function in $\wp(\Sigma^*)^{|\mathcal{V}|} \to \wp(\Sigma^*)^{|\mathcal{V}|}$, which therefore has the least fixpoint $\langle Y_i \rangle_{i \in [0,n]} = \text{lfp}\ (\lambda \vec{X}.\ \vec{b} \cup Fn_{\mathcal{G}}(\vec{X}))$. It is well-known [GR62] that the language $\mathcal{L}(\mathcal{G})$ accepted by $\mathcal{G}$ is such that $\mathcal{L}(\mathcal{G}) = Y_0$. Let L_2 be a language, we define $\langle \vec{L}_2^{X_0} \rangle_0 \triangleq L_2$ and $\langle \vec{L}_2^{X_0} \rangle_{i \in [1,n]} \triangleq \Sigma^*$. It turns out that:

$$\mathcal{L}(\mathcal{G}) \subseteq L_2 \iff \text{lfp}\ (\lambda \vec{X}.\ \vec{b} \cup Fn_{\mathcal{G}}(\vec{X})) \subseteq \vec{L}_2^{X_0}$$

Similarly to Section 2.5.1, backward completeness is again linked to effectively checking the language inclusion. Let ρ be a upper closure operator on Σ^*. In [GRV19] they show that if ρ is backward complete for both $\lambda X.Xa$ and $\lambda X.aX$ for all $a \in \Sigma$, then ρ is backward complete for $Fn_{\mathcal{G}}$ and $\lambda \vec{X}.(\vec{b} \cup Fn_{\mathcal{G}}(\vec{X}))$. It turns out that if ρ is backward complete for $\lambda \vec{X}.(\vec{b} \cup Fn_{\mathcal{G}}(\vec{X}))$, then:

$$\rho(\text{lfp}\ (\lambda \vec{X}.(\vec{b} \cup Fn_{\mathcal{G}}(\vec{X})))) = \text{lfp}\ (\lambda \vec{X}.\rho(\vec{b} \cup Fn_{\mathcal{G}}(\vec{X})))$$

Furthermore, if ρ is backward complete for left and right concatenation and $\rho(L_2) = L_2$, then we have that:

$$\mathcal{L}(\mathcal{G}) \subseteq L_2 \iff \text{lfp}\ (\lambda \vec{X}.\rho(\vec{b} \cup Fn_{\mathcal{G}}(\vec{X}))) \subseteq \vec{L}_2^{X_0}$$

We recall that a *Galois Connection* (GC) between two posets $\langle C, \leq_C \rangle$ (called *concrete domain*) and $\langle A, \leq_A \rangle$ (called *abstract domain*) consists of two functions $\alpha : C \to A$ and $\gamma : A \to C$ such that $\alpha(c) \leq_A a \iff c \leq_C \gamma(a)$ holds for all $c \in C$ and $a \in A$. A GC is denoted by $\langle C, \leq_C \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \leq_A \rangle$. If C or A is a qoset, $\langle C, \leq_C \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \leq_A \rangle$ is a *quasiorder Galois Connection* (QGC).

The following Theorem relies on the equivalence $\mathcal{L}(\mathcal{G}) \subseteq L_2 \iff \text{lfp}\ (\lambda \vec{X}.\rho(\vec{b} \cup Fn_{\mathcal{G}}(\vec{X}))) \subseteq \vec{L}_2^{X_0}$, but formulated on GCs rather than closures. In particular, it shows

how to design an algorithm that solves $\mathcal{L}(\mathcal{G}) \subseteq L_2$ on an abstraction D of the concrete domain $\langle \wp(\Sigma^*), \subseteq \rangle$ whenever D satisfies a list of requirements related to backward completeness and computability.

Theorem 2.5.6. *Let $\mathcal{G}$ be a CFG in CNF and let L_2 be a language over Σ. Let $\langle \wp(\Sigma^*), \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle D, \leq_D \rangle$ be a GC where $\langle D, \leq_D \rangle$ is a poset. Assume that the following properties hold:*

1. *$L_2 \in \gamma(D)$ and for every $a \in \Sigma, x \in \wp(\Sigma^*)$, $\gamma(\alpha(aX)) = \gamma(\alpha(a\gamma(\alpha(X))))$ and $\gamma(\alpha(Xa)) = \gamma(\alpha(\gamma(\alpha(X))a))$;*
2. *$\langle D, \leq_D, \sqcup, \bot_D \rangle$ is an* effective domain*, meaning that $\langle D, \leq_D, \sqcup, \bot_D \rangle$ is an ACC join-semilattice with bottom $\bot_D$, every element of D has a finite representation, the binary relation $\leq_D$ is decidable and the binary lub $\sqcup$ is computable;*
3. *There is an algorithm, say $Fn^\sharp(\vec{X}^\sharp)$, which computes $\alpha(Fn_{\mathcal{G}}(\gamma(\vec{X})))$, for all $\vec{X}^\sharp \in \alpha(\wp(\Sigma^*)^{|\mathcal{V}|})$;*
4. *There is an algorithm, say $\vec{b}^\sharp$, which computes $\alpha(\vec{b})$;*
5. *There is an algorithm, say $Incl^\sharp(\vec{X}^\sharp)$, which decides the abstract inclusion $\vec{X}^\sharp \leq_D \alpha(\vec{L}_2^{X_0})$, for all $\vec{X}^\sharp \in \alpha(\wp(\Sigma^*)^{|\mathcal{V}|})$.*

Then, the following is an algorithm which decides whether $\mathcal{L}(\mathcal{G}) \subseteq L_2$:

$$\begin{cases} \langle Y_i^\sharp \rangle_{i \in [0,n]} \leftarrow \texttt{Kleene}(\lambda \vec{X}^\sharp.(\vec{b}^\sharp \cup Fn_{\mathcal{G}}^\sharp(\vec{X})), \vec{\bot}_D) \\ \textbf{return } Incl^\sharp(\langle Y_i^\sharp \rangle_{i \in [0,n]}) \end{cases}$$

Theorem 2.5.6 is parametric on the effective domain D that can be used to instantiate it. Let $\leq$ be a quasiorder on Σ^*. In [GRV19] they observe that the qoset of antichains $\langle AC_{\langle \Sigma^*, \leq \rangle}, \sqsubseteq \rangle$ can be viewed as an abstraction of the concrete domain of all languages $\wp(\Sigma^*)$ as follows. The maps $\alpha_\leq : \wp(\Sigma^*) \to AC_{\langle \Sigma^*, \leq \rangle}$ and $\gamma_\leq : AC_{\langle \Sigma^*, \leq \rangle} \to \wp(\Sigma^*)$ are defined by:

$$\alpha_\leq(X) \triangleq \lfloor X \rfloor$$

$$\gamma_\leq(X) \triangleq \rho_\leq(X)$$

Where $\alpha_\leq(X)$ is meant to return *any* minor set of the language X since minors are not unique. It turns out that $\langle \wp(\Sigma^*), \subseteq \rangle \underset{\alpha_\leq}{\overset{\gamma_\leq}{\leftrightarrows}} \langle AC_{\langle \Sigma^*, \leq \rangle}, \sqsubseteq \rangle$ is a QGC and $\gamma_\leq \circ \alpha_\leq = \rho_\leq$. Since by definition $\alpha_\leq(X) = \lfloor X \rfloor$, and $\lfloor X \rfloor$ is *finite* (see Section 2.3), the QGC $\langle \wp(\Sigma^*), \subseteq \rangle \underset{\alpha_\leq}{\overset{\gamma_\leq}{\leftrightarrows}} \langle AC_{\langle \Sigma^*, \leq \rangle}, \sqsubseteq \rangle$ allows us to represent and manipulate $\leq$-upward closed sets in $\wp(\Sigma^*)$ using finite subsets. Furthermore, they prove that if $\leq$ is a L_2-consistent well-quasiorder (see Section 2.3.2), Algorithm `CFGInc` solves whether $\mathcal{L}(\mathcal{G}) \subseteq L_2$ holds.

Let $\mathcal{A}$ be a FA. In [GRV19] they observe that $\leq^1_{\mathcal{A}}$ and $\leq_{\mathcal{L}(\mathcal{A})}$ (see Section 2.3.2) are monotonic $\mathcal{L}(\mathcal{A})$-consistent computable wqos that can be used to instantiate Algorithm `CFGInc` in order to solve the language inclusion problem between context-free and regular languages.

Even if we are mainly interested in the language inclusion problem between ω-regular languages, we show that the quasiorder defined in Definition 3.1.1 is a L_2-consistent computable wqo when L_2 is regular, and then it can be used to instantiate Algorithm `CFGInc`.

CFGInc: Algorithm that computes whether $\mathcal{L}(\mathcal{G}) \subseteq L_2$ holds

Data: CGF $\mathcal{G} = \langle \mathcal{V}, \Sigma, P \rangle$
Data: Procedure deciding $u \in L_2$, given $u \in \Sigma^*$
Data: Decidable L_2-consistent wqo $\leq$
Result: Whether $\mathcal{L}(\mathcal{G}) \subseteq L_2$ holds

$\langle Y_i^\sharp \rangle_{i \in [0,n]} \leftarrow \texttt{Kleene}(\lambda \vec{X}.(\lfloor \vec{b} \rfloor \cup \lfloor Fn_{\mathcal{G}}(\vec{X}) \rfloor), \langle \emptyset \rangle_{i \in [0,n]})$;
forall $u \in Y_0$ **do**
  if $u \notin L_2$ **then**
    return false;
  end
end
return true

Example 2.5.7. (Taken from [GRV19]) Let $\mathcal{G} = \langle \{X_0, X_1\}, \{a, b\}, \{X_0 \to X_0X_1 | X_1X_0 | b, X_1 \to a\} \rangle$ a CFG in CNF and $\mathcal{A}$ the automaton in Figure 2.17. For this example, we instantiate the algorithm CFGInc with the state-based quasiorder $\leq^1_{\mathcal{A}}$. First, we observe some relations between words induced by $\leq^1_{\mathcal{A}}$. Observe $ctx_{\mathcal{A}}(\epsilon) = ctx_{\mathcal{A}}(a) = ctx_{\mathcal{A}}(b) = ctx_{\mathcal{A}}(aa) = \{(q_0, q_0), (q_1, q_1), (q_2, q_2)\}$, moreover $ctx_{\mathcal{A}}(ab) = ctx_{\mathcal{A}}(a)$ and $ctx_{\mathcal{A}}(ba) = ctx_{\mathcal{A}}(aa) = ctx_{\mathcal{A}}(baa) = ctx_{\mathcal{A}}(aab) = ctx_{\mathcal{A}}(aba) = \{(q_0, q_2), (q_1, q_2), (q_2, q_2)\}$. We show the first three Kleene iterates computed by Algorithm CFGInc using the qo $\leq^1_{\mathcal{A}}$:

$$\vec{Y}^{(0)} = \langle \emptyset, \emptyset \rangle$$
$$\vec{Y}^{(1)} = \lfloor \vec{b} \rfloor = \langle \{b\}, \{a\} \rangle$$
$$\vec{Y}^{(2)} = \lfloor \vec{b} \rfloor \sqcup \lfloor Fn_{\mathcal{G}}(\vec{Y}^{(1)}) \rfloor = \langle \lfloor \{ba, ab, b\} \rfloor, \lfloor \{a\} \rfloor \rangle = \langle \{ba, ab, b\}, \{a\} \rangle$$
$$\vec{Y}^{(3)} = \lfloor \vec{b} \rfloor \sqcup \lfloor Fn_{\mathcal{G}}(\vec{Y}^{(2)}) \rfloor = \langle \lfloor \{ba, ab, b\} \rfloor, \lfloor \{a\} \rfloor \rangle = \langle \{ba, ab, b\}, \{a\} \rangle$$

So that the least fixpoint is $\vec{Y} = \langle \{ba, ab, b\}, \{a\} \rangle$. Since $ab \in (\vec{Y})_0$ but $ab \notin \mathcal{L}(\mathcal{A})$, Algorithm CFGInc derives $\mathcal{L}(\mathcal{G}) \not\subseteq \mathcal{L}(\mathcal{A})$.

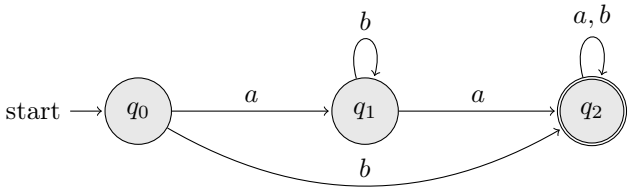

Figure 2.17: Automaton that accepts the language $(b + ab^*a)(a + b)^*$

Chapter 3

Quasiorders on words

In this chapter we define various simulation-based qos and we point out how they can be used to solve the language inclusion problem. In particular, in Section 3.1 we introduce a number of different simulation-based quasiorders on words and for each one of them we prove some properties; in Section 3.2 we show how the defined qos can be used to solve the language inclusion problem and, finally, in Section 3.3 we discuss the relations between the considered quasiorders.

3.1 Simulation-based quasiorders on Σ^*

In this section we define a number of qos on Σ^* that are simulation-based. For each one of them we provide one example, the proof of being a computable well-quasiorder, we discuss its monotonicity properties and we make some meaningful comparisons with other qos.

Definition 3.1.1. Let $u, v \in \Sigma^*$.

$$u \sqsubseteq^1_{\mathcal{A}} v \overset{\triangle}{\iff} \forall (q_1, q_2) \in ctx_{\mathcal{A}}(u)\ \exists (q_3, q_4) \in ctx_{\mathcal{A}}(v)$$
$$such\ that\ q_1 \preceq^r q_3\ \wedge\ q_2 \preceq^{di} q_4$$

Example 3.1.2. Let $\mathcal{A}$ be the automaton in Figure 3.1. We observe that $b \sqsubseteq^1_{\mathcal{A}} d$. Consider $ctx_{\mathcal{A}}(b) = \{(q_2, q_3)\}$ and $ctx_{\mathcal{A}}(d) = \{(q_9, q_{10})\}$: if from q_2 Spoiler plays $q_2 \xrightarrow{a}_R q_1 \xrightarrow{a}_R q_0$, Duplicator can play $q_9 \xrightarrow{a}_R q_8 \xrightarrow{a}_R q_0$ and since $q_0 \in I$ Duplicator wins, so that $q_2 \preceq^r q_9$. Observe that doesn't matter if $q_1 \in F$, since $\preceq^r$ doesn't impose any condition on final states. It also holds that $q_3 \preceq^{di} q_{10}$: if Spoiler plays $q_3 \xrightarrow{e} q_4$, Duplicator can answer $q_{10} \xrightarrow{e} q_{11}$ and $q_4 \preceq^{di} q_{11}$. Furthermore, $q_4 \in F$ and $q_{11} \in F$. The other cases are similar.

Proposition 3.1.3.

$$\sqsubseteq^1_{\mathcal{A}} \text{ is a decidable wqo.}$$

Proof. For every $u \in \Sigma^*$, $ctx_{\mathcal{A}}(u)$ is a finite and computable set, $\preceq^{di}$ and $\preceq^r$ are computable so that $\sqsubseteq^1_{\mathcal{A}}$ is a decidable wqo. □

Proposition 3.1.4 (Monotonicity)**.** Let $u, v, x, y \in \Sigma^*$.

$$u \sqsubseteq^1_{\mathcal{A}} v \implies xuy \sqsubseteq^1_{\mathcal{A}} xvy$$

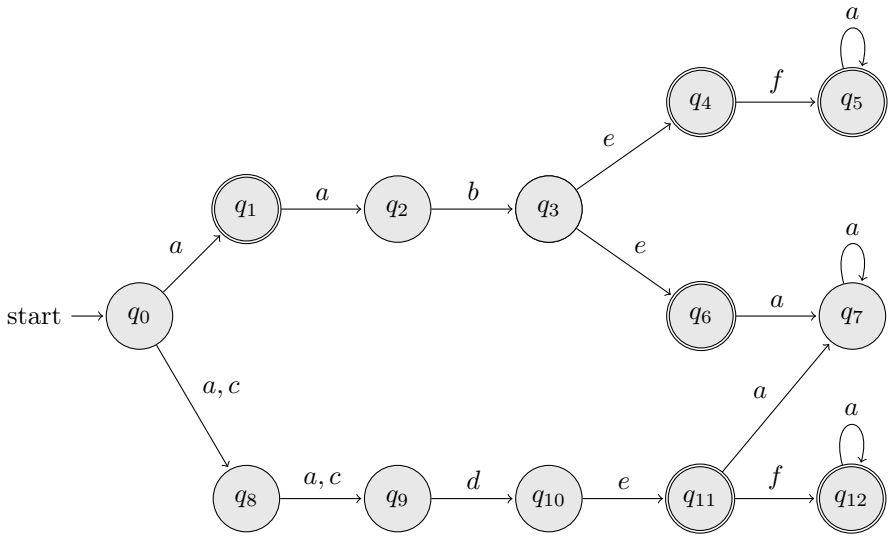

Figure 3.1: Example of $\sqsubseteq^1_{\mathcal{A}}$

Proof. Let $(q'_1, q'_2) \in ctx_{\mathcal{A}}(xuy)$, then $\exists (q_1, q_2) \in ctx_{\mathcal{A}}(u)$ such that $q'_1 \overset{x}{\rightsquigarrow} q_1$ and $q_2 \overset{y}{\rightsquigarrow} q'_2$. Furthermore, since $u \sqsubseteq^1_{\mathcal{A}} v$, $\exists (q_3, q_4) \in ctx_{\mathcal{A}}(v)$ such that $q_1 \preceq^r q_3 \wedge q_2 \preceq^{di} q_4$. For Lemma 2.2.1, $q'_1 \overset{x}{\rightsquigarrow} q_1$ implies that $q_1 \overset{x^R}{\rightsquigarrow}_R q'_1$, and since $q_1 \preceq^r q_3$, $\exists q'_3 \in Q$ such that $q_3 \overset{x^R}{\rightsquigarrow}_R q'_3$. Furthermore, for the definition of reverse simulation $q'_1 \preceq^r q'_3$ holds. Observe that $q_2 \overset{y}{\rightsquigarrow} q'_2$ and $q_2 \preceq^{di} q_4$ imply that $\exists q'_4 \in Q$ such that $q_4 \overset{y}{\rightsquigarrow} q'_4$ and, due to the definition of direct simulation, $q'_2 \preceq^{di} q'_4$. $(q'_3, q'_4) \in ctx_{\mathcal{A}}(xvy)$ holds because $(q_3, q_4) \in ctx_{\mathcal{A}}(v)$, $q'_3 \overset{x}{\rightsquigarrow} q_3$ and $q_4 \overset{y}{\rightsquigarrow} q'_4$. □

We now discuss the relations between $\sqsubseteq^1_{\mathcal{A}}$ and the qos on Σ^* presented in Section 2.3.2.

Proposition 3.1.5.

$$\leq^1_{\mathcal{A}} \subseteq \sqsubseteq^1_{\mathcal{A}}$$

Proof. Let $u, v \in \Sigma^*$ such that $u \leq^1_{\mathcal{A}} v$. Then $ctx_{\mathcal{A}}(u) \subseteq ctx_{\mathcal{A}}(v)$ so that $\forall (q_1, q_2) \in ctx_{\mathcal{A}}(u)$ $\exists (q_3, q_4) \in ctx_{\mathcal{A}}(v)$ such that $q_1 \preceq^r q_3 \wedge q_2 \preceq^{di} q_4$ because $\preceq^{di}$ and $\preceq^r$ are reflexive. Then, $\leq^1_{\mathcal{A}} \subseteq \sqsubseteq^1_{\mathcal{A}}$. □

Proposition 3.1.6. Let $\mathcal{A}$ be an FA.

$$\sqsubseteq^1_{\mathcal{A}} \subseteq \leqq_{\mathcal{L}(\mathcal{A})}$$

Proof. Let $u, v \in \Sigma^*$ such that $u \sqsubseteq^1_{\mathcal{A}} v$, $(x, y) \in ctx_{\mathcal{L}(\mathcal{A})}(u)$, $(q_1, q_2) \in ctx_{\mathcal{A}}(u)$ such that $x \in W^{\mathcal{A}}_{I,q_1}$ and $y \in W^{\mathcal{A}}_{q_2,F}$. Since $u \sqsubseteq^1_{\mathcal{A}} v$, $\exists (q_3, q_4) \in ctx_{\mathcal{A}}(v)$ such that $q_1 \preceq^r q_3 \wedge q_2 \preceq^{di} q_4$. Since $x \in W^{\mathcal{A}}_{I,q_1}$ and $q_1 \preceq^r q_3$, $x \in W^{\mathcal{A}}_{I,q_3}$. Similarly, since $y \in W^{\mathcal{A}}_{q_2,F}$ and $q_2 \preceq^{di} q_4$, $y \in W^{\mathcal{A}}_{q_4,F}$ so that $(x, y) \in ctx_{\mathcal{L}(\mathcal{A})}(v)$. Then, $\sqsubseteq^1_{\mathcal{A}} \subseteq \leqq_{\mathcal{L}(\mathcal{A})}$. □

Remark 3.1.7. Let $\mathcal{A}$ be the FA represented in Figure 3.2. In this case $ctx_{\mathcal{A}}(c) = \{(q_1, q_4)\}$ and $ctx_{\mathcal{A}}(d) = \{(q_2, q_5), (q_3, q_6)\}$ so that $d \not\leq^1_{\mathcal{A}} c$. Furthermore we observe that $q_2 \preceq^r q_1$, $q_5 \preceq^{di} q_4$, $q_3 \preceq^r q_1$ and $q_6 \preceq^{di} q_4$ so that $\forall (p_1, p_2) \in ctx_{\mathcal{A}}(d)$ $\exists (p_3, p_4) \in$

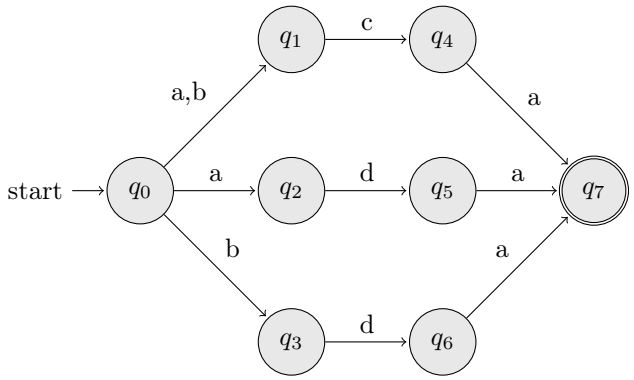

Figure 3.2: FA accepting the language $(a+b)ca + ada + bda$.

$ctx_{\mathcal{A}}(c)$ such that $p_1 \preceq^r p_3 \wedge p_2 \preceq^{di} p_4$ and then $d \sqsubseteq^1_{\mathcal{A}} c$. This shows that $\boxed{\leq_{\mathcal{A}} \subsetneq \sqsubseteq^1_{\mathcal{A}}}$. Let us observe that $ctx_{\mathcal{L}(\mathcal{A})}(c) = \{(a,a),(b,a)\}$ and $ctx_{\mathcal{L}(\mathcal{A})}(d) = \{(a,a),(b,a)\}$ so that $c \leqq_{\mathcal{L}(\mathcal{A})} d$. Furthermore $c \sqsubseteq^1_{\mathcal{A}} d$ does not hold because $\nexists (p_3,p_4) \in ctx_{\mathcal{A}}(d)$ such that $q_1 \preceq^r p_3$: $q_2 \overset{b}{\nrightarrow}_R$ and $q_3 \overset{a}{\nrightarrow}_R$. This shows that $\boxed{\sqsubseteq^1_{\mathcal{A}} \subsetneq \leqq_{\mathcal{L}(\mathcal{A})}}$.

Proposition 3.1.8.

$$\sqsubseteq^1_{\mathcal{A}} \subseteq \sqsubseteq^r_{\mathcal{A}} \qquad \sqsubseteq^1_{\mathcal{A}} \subseteq \sqsubseteq^l_{\mathcal{A}}$$

Proof. Let $u, v \in \Sigma^*$ such that $u \sqsubseteq^1_{\mathcal{A}} v$. Consider $(q_1,q_2) \in ctx_{\mathcal{A}}(u)$ such that $q_1 \in I$ so that $q_2 \in post^{\mathcal{A}}_u(I)$. Then $\exists (q_3,q_4) \in ctx_{\mathcal{A}}(v)$ such that $q_1 \preceq^r q_3 \wedge q_2 \preceq^{di} q_4$. Observe that $q_1 \in I$ and $q_1 \preceq^r q_3$ imply that $q_3 \in I$ and then $q_4 \in post^{\mathcal{A}}_v(I)$. Since $q_2 \preceq^{di} q_4$, $u \sqsubseteq^r_{\mathcal{A}} v$. The proof that $\sqsubseteq^1_{\mathcal{A}} \subseteq \sqsubseteq^l_{\mathcal{A}}$ is symmetric. □

Remark 3.1.9.

$$\sqsubseteq^1_{\mathcal{A}} \subseteq (\sqsubseteq^r_{\mathcal{A}} \cap \sqsubseteq^l_{\mathcal{A}})$$

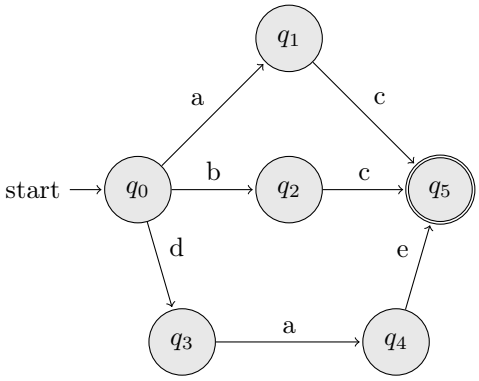

Figure 3.3: FA accepting the language $ac + bc + dae$.

Remark 3.1.10. Let $\mathcal{A}$ be the FA in Figure 3.3. Observe that $a \sqsubseteq^r_{\mathcal{A}} b$: $post^{\mathcal{A}}_a(I) = \{q_1\}$, and $q_2 \in post^{\mathcal{A}}_b(I)$ simulates q_1. Furthermore, $a \sqsubseteq^l_{\mathcal{A}} b$ because $pre^{\mathcal{A}}_a(F) = \varnothing$. Observe that $(q_3,q_4) \in ctx_{\mathcal{A}}(a)$, but $\nexists (p_3,p_4) \in ctx_{\mathcal{A}}(b)$ such that $p_4 \overset{e}{\rightarrow}$ and $p_3 \overset{d}{\rightarrow}_R$,

and then $a \sqsubseteq^1_{\mathcal{A}} b$ does not hold. This shows that $(a,b) \in \sqsubseteq^r_{\mathcal{A}} \cap \sqsubseteq^l_{\mathcal{A}}$ and $(a,b) \notin \sqsubseteq^1_{\mathcal{A}}$ so that $\boxed{\sqsubseteq^1_{\mathcal{A}} \subsetneq \sqsubseteq^r_{\mathcal{A}} \cap \sqsubseteq^l_{\mathcal{A}}}$.

We now give a definition of a quasiorder that is a strengthened version of $\sqsubseteq^1_{\mathcal{A}}$.

Definition 3.1.11. Let $u, v \in \Sigma^*$.

$$u \sqsubseteq^2_{\mathcal{A}} v \overset{\triangle}{\iff} \forall (q_1,q_2) \in ctx_{\mathcal{A}}(u)\ \exists (q_3,q_4) \in ctx_{\mathcal{A}}(v) \text{ such that}$$
$$(i)\ q_1 \preceq^b q_3 \ \wedge\ q_2 \preceq^{di} q_4$$
$$(ii)\ q_1 \overset{u}{\rightarrowtail} q_2 \implies q_3 \overset{v}{\rightarrowtail} q_4$$

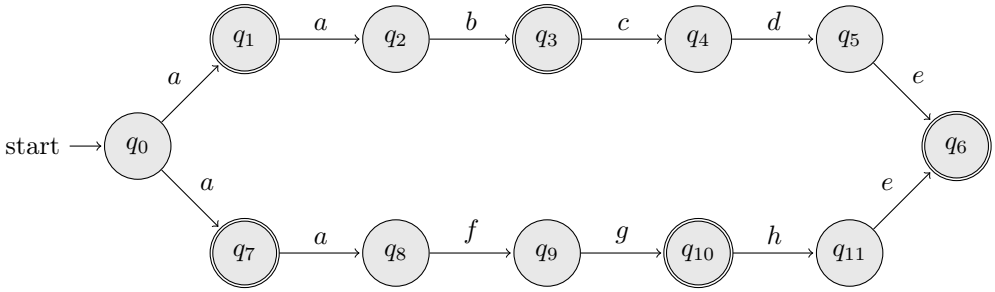

Figure 3.4: Example of $\sqsubseteq^2_{\mathcal{A}}$

Example 3.1.12. Let $\mathcal{A}$ be the automaton in Figure 3.4. We observe that $bcd \sqsubseteq^2_{\mathcal{A}} fgh$. Consider $ctx_{\mathcal{A}}(bcd) = \{(q_2,q_5)\}$, $ctx_{\mathcal{A}}(fgh) = \{(q_8,q_{11})\}$: if from q_2 Spoiler plays $q_2 \xrightarrow{a}_R q_1 \xrightarrow{a}_R q_0$ Duplicator can play $q_8 \xrightarrow{a}_R q_7 \xrightarrow{a}_R q_0$. Furthermore, $q_1 \in F$, $q_7 \in F$ and $q_0 \in I$ so that *both initial and final states* are matched. This implies $q_2 \preceq^b q_8$. It is also easy to see that $q_5 \preceq^{di} q_{11}$. Furthermore observe that, since $q_2 \overset{bdc}{\rightarrowtail} q_5$, it holds that $q_8 \overset{fgh}{\rightarrowtail} q_{11}$.

Proposition 3.1.13.

$$\sqsubseteq^2_{\mathcal{A}} \text{ is a decidable wqo.}$$

Proof. For every $u \in \Sigma^*$, $ctx_{\mathcal{A}}(u)$ is a finite and computable set, $\preceq^{di}$ and $\preceq^b$ are computable, $\overset{u}{\rightarrowtail}$ is computable so that $\sqsubseteq^2_{\mathcal{A}}$ is a decidable wqo. □

Proposition 3.1.14 (Monotonicity)**.** Let $u, v, x, y \in \Sigma^*$.

$$u \sqsubseteq^2_{\mathcal{A}} v \implies xuy \sqsubseteq^2_{\mathcal{A}} xvy$$

Proof. Let $(q'_1,q'_2) \in ctx_{\mathcal{A}}(xuy)$, so that there exists $\exists (q_1,q_2) \in ctx_{\mathcal{A}}(u)$ such that $q'_1 \overset{x}{\rightsquigarrow} q_1$ and $q_2 \overset{y}{\rightsquigarrow} q'_2$. Then, by $u \sqsubseteq^2_{\mathcal{A}} v$, $\exists (q_3,q_4) \in ctx_{\mathcal{A}}(v)$ such that $q_1 \preceq^b q_3 \ \wedge\ q_2 \preceq^{di} q_4$. Since $q_1 \overset{x^R}{\rightsquigarrow} q'_1$ and $q_1 \preceq^b q_3$, $\exists q'_3 \in Q$ such that $q_3 \overset{x^R}{\rightsquigarrow} q'_3$. By definition of backward simulation, $q'_1 \preceq^b q'_3$. Since $q_2 \overset{y}{\rightsquigarrow} q'_2$ and $q_2 \preceq^{di} q_4$, $\exists q'_4 \in Q$ such that $q_4 \overset{y}{\rightsquigarrow} q'_4$. By definition of direct simulation, $q'_2 \preceq^{di} q'_4$. Observe that $(q'_3,q'_4) \in ctx_{\mathcal{A}}(xvy)$ because $q'_3 \overset{x}{\rightsquigarrow} q_3$, $(q_3,q_4) \in ctv_{\mathcal{A}}(v)$ and $q_4 \overset{y}{\rightsquigarrow} q'_4$. Assume now that $q'_1 \overset{xuy}{\rightarrowtail} q'_2$ holds. At least one of the following conditions holds: (i) $q'_1 \overset{x}{\rightarrowtail} q_1 \overset{u}{\rightsquigarrow} q_2 \overset{y}{\rightsquigarrow} q'_2$; (ii) $q'_1 \overset{x}{\rightsquigarrow} q_1 \overset{u}{\rightarrowtail} q_2 \overset{y}{\rightsquigarrow}$

q_2'; (*iii*) $q_1' \overset{x}{\rightsquigarrow} q_1 \overset{u}{\rightsquigarrow} q_2 \overset{y}{\rightarrowtail} q_2'$. If $q_1' \overset{x}{\rightarrowtail} q_1 \overset{u}{\rightsquigarrow} q_2 \overset{y}{\rightsquigarrow} q_2'$, by $u \sqsubseteq^2_{\mathcal{A}} v$, $\exists (q_3, q_4) \in ctx_{\mathcal{A}}(v)$ such that $q_1 \preceq^b q_3 \ \wedge \ q_2 \preceq^{di} q_4$. Since $q_1 \overset{x^R}{\rightarrowtail} q_1'$ and $q_1 \preceq^b q_3$, $\exists q_3' \in Q$ such that $q_3 \overset{x^R}{\rightarrowtail} q_3'$. By definition of backward simulation, $q_1' \preceq^b q_3'$. Since $q_2 \overset{y}{\rightsquigarrow} q_2'$ and $q_2 \preceq^{di} q_4$, $\exists q_4' \in Q$ such that $q_4 \overset{y}{\rightsquigarrow} q_4'$. By definition of direct simulation, $q_2' \preceq^{di} q_4'$. Observe that $(q_3', q_4') \in ctx_{\mathcal{A}}(xvy)$ because $q_3' \overset{x}{\rightarrowtail} q_3$, $(q_3, q_4) \in ctv_{\mathcal{A}}(v)$ and $q_4 \overset{y}{\rightsquigarrow} q_4'$. The subcase $q_1' \overset{x}{\rightsquigarrow} q_1 \overset{u}{\rightsquigarrow} q_2 \overset{y}{\rightarrowtail} q_2'$ is similar. If $q_1' \overset{x}{\rightsquigarrow} q_1 \overset{u}{\rightarrowtail} q_2 \overset{y}{\rightsquigarrow} q_2'$, by $u \sqsubseteq^2_{\mathcal{A}} v$, $\exists (q_3, q_4) \in ctx_{\mathcal{A}}(v)$ such that $q_1 \preceq^b q_3 \ \wedge \ q_2 \preceq^{di} q_4$. Furthermore, $q_3 \overset{v}{\rightarrowtail} q_4$. Since $q_1 \overset{x^R}{\rightsquigarrow} q_1'$ and $q_1 \preceq^b q_3$, $\exists q_3' \in Q$ such that $q_3 \overset{x^R}{\rightsquigarrow} q_3'$. By definition of backward simulation, $q_1' \preceq^b q_3'$. Since $q_2 \overset{y}{\rightsquigarrow} q_2'$ and $q_2 \preceq^{di} q_4$, $\exists q_4' \in Q$ such that $q_4 \overset{y}{\rightsquigarrow} q_4'$. By definition of direct simulation, $q_2' \preceq^{di} q_4'$. Observe that $(q_3', q_4') \in ctx_{\mathcal{A}}(xvy)$ because $q_3' \overset{x}{\rightsquigarrow} q_3$, $(q_3, q_4) \in ctv_{\mathcal{A}}(v)$ and $q_4 \overset{y}{\rightsquigarrow} q_4'$. In each possible subcase $q_3' \overset{xvy}{\rightarrowtail} q_4'$. □

Proposition 3.1.15.

$$\sqsubseteq^2_{\mathcal{A}} \subseteq \sqsubseteq^1_{\mathcal{A}}$$

Proof. It is immediate to see that $\sqsubseteq^2_{\mathcal{A}} \subseteq \sqsubseteq^1_{\mathcal{A}}$, since $\preceq^b \ \subseteq \preceq^r$ (see Section 2.4). □

The following quasiorder is a generalization of $\sqsubseteq^r_{\mathcal{A}}$, that exploits the *delayed simulation* in order to define a coarser relation.

Definition 3.1.16. Let $u, v \in \Sigma^*$.

$$u \sqsubseteq^{de,r}_{\mathcal{A}} v \overset{\triangle}{\iff} \forall p \in post^{\mathcal{A}}_u(I) \ \exists q \in post^{\mathcal{A}}_v(I) \text{ such that } p \preceq^{de} q$$

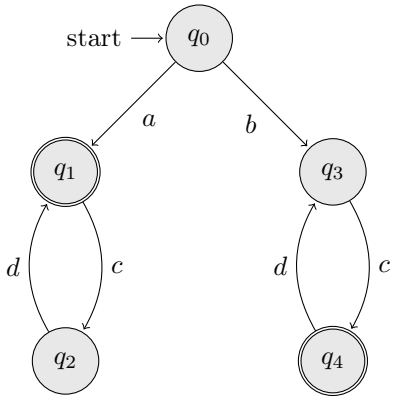

Figure 3.5: Example of $\sqsubseteq^{de,r}_{\mathcal{A}}$

Example 3.1.17. Let $\mathcal{A}$ be the automaton in Figure 3.5. We observe that $a \sqsubseteq^{de,r}_{\mathcal{A}} b$. Consider $post^{\mathcal{A}}_a(I) = \{q_1\}$, $post^{\mathcal{A}}_b(I) = \{q_3\}$: from q_1 starts the infinite trace $\pi_0 = q_1 \overset{c}{\rightarrow} q_2 \overset{d}{\rightarrow} q_1 \overset{c}{\rightarrow} \cdots$, which can be matched with the trace $\pi_1 = q_3 \overset{c}{\rightarrow} q_4 \overset{d}{\rightarrow} q_3 \overset{c}{\rightarrow} \cdots$. Furthermore, each final state in π_0 will be matched after exactly one move. This implies $q_1 \preceq^{de} q_3$, so that $a \sqsubseteq^{de,r}_{\mathcal{A}} b$.

Proposition 3.1.18.

$$\sqsubseteq^{de,r}_{\mathcal{A}} \text{ is a decidable wqo.}$$

Proof. For every $u \in \Sigma^*$, $post_u^{\mathcal{A}}(I)$ is a finite and computable set, $\preceq^{de}$ is computable so that $\sqsubseteq_{\mathcal{A}}^{de,r}$ is a decidable wqo. □

Proposition 3.1.19 (Right monotonicity)**.** Let $u, v, x \in \Sigma^*$.

$$u \sqsubseteq_{\mathcal{A}}^{de,r} v \implies ux \sqsubseteq_{\mathcal{A}}^{de,r} vx$$

Proof. Let $q \in post_{ux}^{\mathcal{A}}(I)$, then $\exists i \in I, p \in Q$ such that $i \overset{u}{\rightsquigarrow} p \overset{x}{\rightsquigarrow} q$. Since $u \sqsubseteq_{\mathcal{A}}^{de,r} v$, $\exists i' \in I, p' \in Q$ such that $i' \overset{v}{\rightsquigarrow} p'$ and $p \preceq^{de} p'$. We observe that $p \overset{x}{\rightsquigarrow} q$ and $p \preceq^{de} p'$ implies that $\exists q' \in Q$ such that $p' \overset{x}{\rightsquigarrow} q'$. Furthermore, by definition of delayed simulation, $q \preceq^{de} q'$. We conclude by observing that $q' \in post_{vx}^{\mathcal{A}}(I)$ implies that $ux \sqsubseteq_{\mathcal{A}}^{de,r} vx$. □

Example 3.1.20. Let $\mathcal{A}$ be automaton in Figure 3.6. Observe that $u \sqsubseteq_{\mathcal{A}}^{de,r} v$, because $post_u^{\mathcal{A}}(I) = \{q_1\}$, $q_1 \preceq^{de} q_4$ and $q_4 \in post_v^{\mathcal{A}}(I)$. Consider now $post_{wu}^{\mathcal{A}}(I) = \{q_8\}$: since $post_{wv}^{\mathcal{A}}(I) = \{q_{11}\}$, $\nexists q \in post_{wv}^{\mathcal{A}}(I)$ such that $q_8 \preceq^{de} q$. This is due to the fact that $q_8 \overset{a}{\rightarrow} q_9$, but $q_{11} \overset{a}{\not\rightarrow}$. This implies that $u \sqsubseteq_{\mathcal{A}}^{de,r} v \not\Longrightarrow wu \sqsubseteq_{\mathcal{A}}^{de,r} wv$, so that $\sqsubseteq_{\mathcal{A}}^{de,r}$ *is not a left-monotonic quasiorder.*

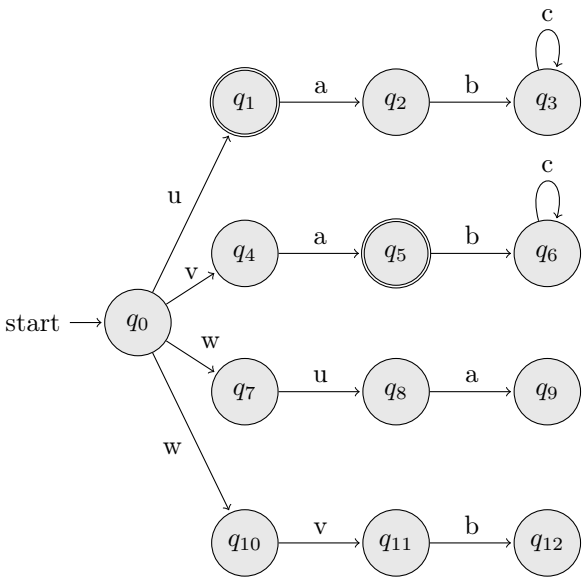

Figure 3.6: Automaton that shows that $\sqsubseteq_{\mathcal{A}}^{de,r}$ is not left-monotonic

Proposition 3.1.21.

$$\sqsubseteq_{\mathcal{A}}^{r} \subseteq \sqsubseteq_{\mathcal{A}}^{de,r}$$

Proof. Let $u, v \in \Sigma^*$ such that $u \sqsubseteq_{\mathcal{A}}^{r} v$. This implies that $\forall u, v \in \Sigma^*$, if $\forall p \in post_u^{\mathcal{A}}(I), \exists q \in post_v^{\mathcal{A}}(I)$ such that $p \preceq^{di} q$, then it also holds that $p \preceq^{de} q$. This is implied by $\preceq^{di} \subseteq \preceq^{de}$ (see Section 2.4). □

The following quasiorder is the last one that we define and is another generalization of $\sqsubseteq_{\mathcal{A}}^{r}$ that relies on the *fair* simulation.

Definition 3.1.22. Let $u, v \in \Sigma^*$.

$$u \sqsubseteq_{\mathcal{A}}^{fair,r} v \overset{\triangle}{\iff} \forall p \in post_u^{\mathcal{A}}(I)\ \exists q \in post_v^{\mathcal{A}}(I) \text{ such that } p \preceq^{f} q$$

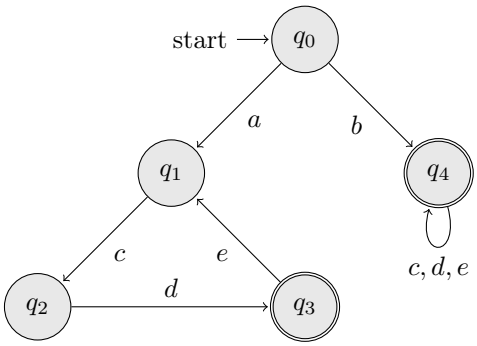

Figure 3.7: Example of $\sqsubseteq_{\mathcal{A}}^{fair,r}$

Example 3.1.23. Let $\mathcal{A}$ be the automaton in Figure 3.7. We observe that $a \sqsubseteq_{\mathcal{A}}^{fair,r} b$. Consider $post_a^{\mathcal{A}}(I) = \{q_1\}$, $post_b^{\mathcal{A}}(I) = \{q_4\}$: from q_1 starts the fair trace $\pi_0 = q_1 \xrightarrow{c} q_2 \xrightarrow{d} q_3 \xrightarrow{e} q_1 \xrightarrow{c} \cdots$, which can be matched with the trace $\pi_1 = q_4 \xrightarrow{c} q_4 \xrightarrow{d} q_4 \xrightarrow{e} q_4 \xrightarrow{c} q_4 \cdots$. Furthermore, also π_1 is fair. This implies $q_1 \preceq^f q_3$, so that $a \sqsubseteq_{\mathcal{A}}^{fair,r} b$.

Proposition 3.1.24.

$$\sqsubseteq_{\mathcal{A}}^{fair,r} \text{ is a decidable wqo.}$$

Proof. For every $u \in \Sigma^*$, $post_u^{\mathcal{A}}(I)$ is a finite and computable set, $\preceq^f$ is computable so that $\sqsubseteq_{\mathcal{A}}^{fair,r}$ is a decidable wqo. □

Remark 3.1.25. Let us observe that if $p \preceq^f q$ and π is a fair trace starting from p then for all $p' \in Q$ such that, for some $w \in \Sigma^*$, $p \overset{w}{\rightsquigarrow} p'$ is a prefix of π there exists $q' \in Q$ such that $q \overset{w}{\rightsquigarrow} q'$ and $p' \preceq^f q'$. In fact, there exists a fair trace π' starting from q which matches π, so that there exists $q' \in Q$ such that $q \overset{w}{\rightsquigarrow} q'$. Moreover, the suffix $\pi_{p' \to}$ is a fair trace matched by the fair trace $\pi'_{q' \to}$, so that $p' \preceq^f q'$ holds. □

Remark 3.1.26. Let us also remark that for all $q \in Q$, if there is no fair trace starting from q, then for all $q' \in Q$, $q \preceq^f q'$ holds, because, by forward completeness of $\mathcal{A}$, any infinite (but not fair) trace starting from q can be matched by an infinite trace starting from any $q' \in Q$.

Proposition 3.1.27 (Right monotonicity)**.** Let $u, v, x \in \Sigma^*$.

$$u \sqsubseteq_{\mathcal{A}}^{fair,r} v \implies ux \sqsubseteq_{\mathcal{A}}^{fair,r} vx$$

Proof. Consider $i \in I, p, q \in Q$ such that $i \overset{u}{\rightsquigarrow} p \overset{x}{\rightsquigarrow} q$. Since $p \in post_u^{\mathcal{A}}(I)$ and $u \sqsubseteq_{\mathcal{A}}^{fair,r} v$, $\exists p' \in Q$ such that $p' \in post_v^{\mathcal{A}}(I)$, $p \preceq^f p'$. If $p \overset{x}{\rightsquigarrow} q$ can be prolonged to a fair trace π_0 then, by Remark 3.1.25, there exists $q' \in Q$ such that $p' \overset{x}{\rightsquigarrow} q'$ and $q \preceq^f q'$ holds. If $p \overset{x}{\rightsquigarrow} q$ cannot be prolonged to a fair trace, by forward completeness of $\mathcal{A}$, the finite trace $p \overset{x}{\rightsquigarrow} q$ can still be matched so that there exists $q' \in Q$ such that $p' \overset{x}{\rightsquigarrow} q'$. Moreover, since there exists no fair trace starting from q, otherwise $p \overset{x}{\rightsquigarrow} q$ could be prolonged to a fair trace, by Remark 3.1.26, it turns out that $q \preceq^f q'$. □

Example 3.1.28. Let $\mathcal{A}$ be automaton in Figure 3.8. Observe that $u \sqsubseteq_{\mathcal{A}}^{fair,r} v$, because $post_u^{\mathcal{A}}(I) = \{q_1\}$, $q_1 \preceq^f q_4$ and $q_4 \in post_v^{\mathcal{A}}(I)$. Consider now $post_{wu}^{\mathcal{A}}(I) = \{q_9\}$: since $post_{wv}^{\mathcal{A}}(I) = \{q_{12}\}$, $\nexists q \in post_{wv}^{\mathcal{A}}(I)$ such that $q_8 \preceq^f q$. This is due to the fact that from q_9 starts one fair trance, while from q_{12} starts an *infinite* but not *fair* trace. This

implies that $u \sqsubseteq_{\mathcal{A}}^{fair,r} v \implies wu \sqsubseteq_{\mathcal{A}}^{fair,r} wv$, so that $\sqsubseteq_{\mathcal{A}}^{fair,r}$ *is not a left-monotonic quasiorder.*

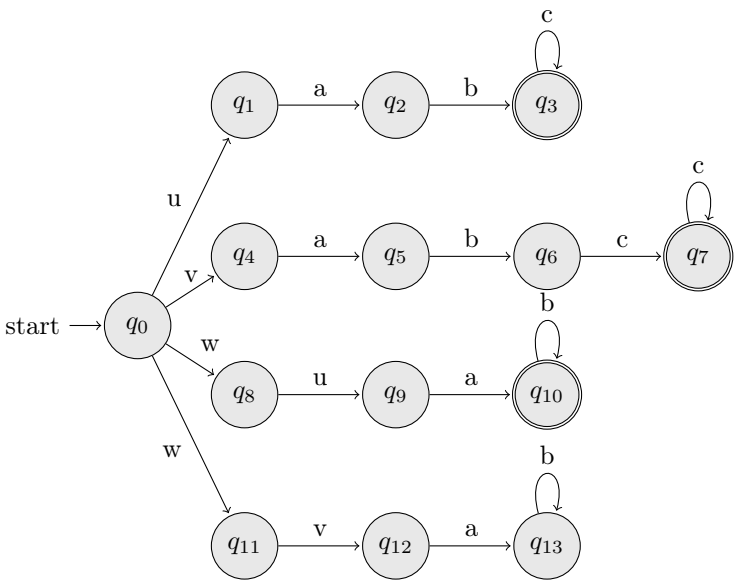

Figure 3.8: Automaton that shows that $\sqsubseteq_{\mathcal{A}}^{fair,r}$ is not left-monotonic

Proposition 3.1.29.

$$\sqsubseteq_{\mathcal{A}}^{de,r} \subseteq \sqsubseteq_{\mathcal{A}}^{fair,r}$$

Proof. Let $u, v \in \Sigma^*$ such that $u \sqsubseteq_{\mathcal{A}}^{de,r} v$. Since $\forall p \in post_u^{\mathcal{A}}(I), \exists q \in post_v^{\mathcal{A}}(I)$ such that $p \preceq^{de} q$, then it also holds that $p \preceq^{f} q$. This is due to the fact that $\preceq^{de} \subseteq \preceq^{f}$ (see Section 2.4). □

3.1.1 Other simulation-based quasiorders on Σ^*

We define here two families of qos on words that are not suitable for being used in Algorithm BAInc to check the language inclusion, since it turns out that they are not right-monotonic qos. The first relies on the *k-lookahead simulation*, while the second on *trace inclusions*.

Definition 3.1.30. Let $u, v \in \Sigma^*$, $x \in \{di, de, f\}$ and $k \geq 1$.

$$u \sqsubseteq_{\mathcal{A}}^{k-x} v \overset{\triangle}{\iff} \forall (q_1, q_2) \in ctx_{\mathcal{A}}(u)\ \exists (q_3, q_4) \in ctx_{\mathcal{A}}(v) \text{ such that}$$
$$q_1 \preceq^{b} q_3 \ \wedge\ q_2 \preceq^{k-x} q_4$$

Example 3.1.31. Let $\mathcal{A}$ be the automaton in Figure 3.9. Consider $k = 2$: $q_2 \preceq^{2-x} q_7$. This is due to the fact that if Spoiler plays $q_2 \xrightarrow{w_1} q_3 \xrightarrow{w_2} q_4$ Duplicator can play $q_7 \xrightarrow{w_1} q_8 \xrightarrow{w_2} q_{10}$. Similarly, if Spoiler plays $q_2 \xrightarrow{w_1} q_3 \xrightarrow{w_3} q_5$, Duplicator can answer $q_7 \xrightarrow{w_1} q_9 \xrightarrow{w_3} q_{11}$. It is obvious that $q_4 \preceq^{2-x} q_{10}$ and $q_5 \preceq^{2-x} q_{11}$. Observe that $ctx_{\mathcal{A}}(u) = \{(q_1, q_2)\}$ and $ctx_{\mathcal{A}}(v) = \{(q_6, q_7)\}$. Since $q_2 \preceq^{2-x} q_7$ and $q_1 \preceq^{b} q_6$,

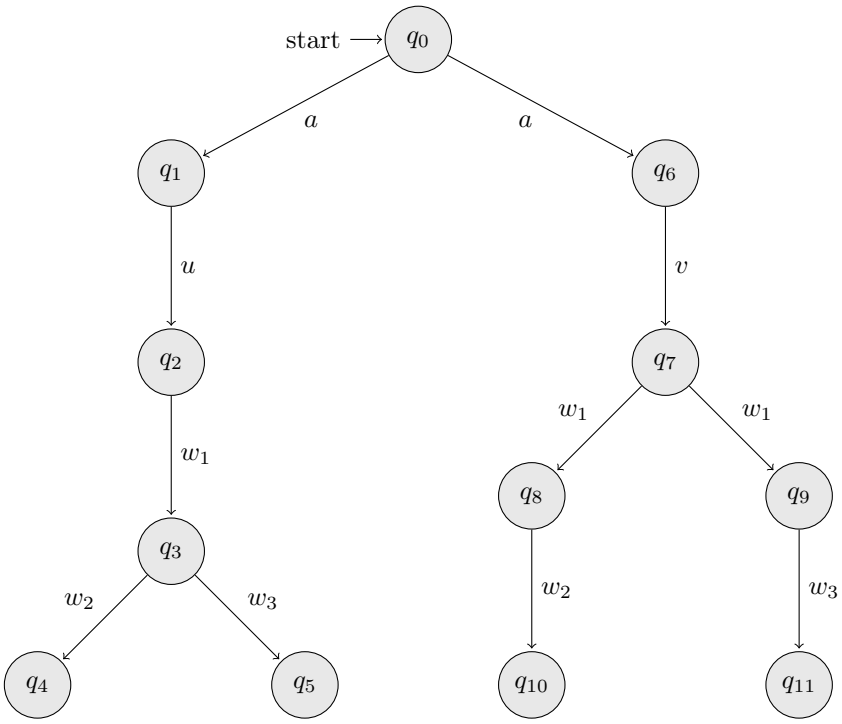

Figure 3.9: Example of use of $\sqsubseteq_{\mathcal{A}}^{k-x}$ and $\sqsubseteq_{\mathcal{A}}^{t-x}$

$\forall (p_1, p_2) \in ctx_{\mathcal{A}}(u)\ \exists (p_3, p_4) \in ctx_{\mathcal{A}}(v)$ such that $p_1 \preceq^b p_3 \ \wedge\ p_2 \preceq^{2-x} p_4$, and then $u \sqsubseteq_{\mathcal{A}}^{2-x} v$.

Consider now uw_1 and vw_1. Observe that $ctx_{\mathcal{A}}(uw_1) = \{(q_1, q_3)\}$ and $ctx_{\mathcal{A}}(vw_1) = \{(q_6, q_8), (q_6, q_9)\}$. Due to the fact that $q_8 \overset{w_3}{\not\to}$ and $q_9 \overset{w_2}{\not\to}$, $q_3 \preceq^{2-x} q_8$ and $q_3 \preceq^{2-x} q_9$ don't hold. This implies that also $uw_1 \sqsubseteq_{\mathcal{A}}^{2-x} vw_1$ doesn't hold. We showed that $\exists u, v, w_1 \in \Sigma^*$ such that $u \sqsubseteq_{\mathcal{A}}^{2-x} v \implies uw_1 \sqsubseteq_{\mathcal{A}}^{2-x} vw_1$, so that $\sqsubseteq_{\mathcal{A}}^{k-x}$ *is not right-monotonic*. This implies that for our purposes $\sqsubseteq_{\mathcal{A}}^{k-x}$ is not interesting.

Definition 3.1.32. Let $u, v \in \Sigma^*$ and $x \in \{di, de, f\}$.

$$u \sqsubseteq_{\mathcal{A}}^{t-x} v \overset{\triangle}{\iff} \forall (q_1, q_2) \in ctx_{\mathcal{A}}(u)\ \exists (q_3, q_4) \in ctx_{\mathcal{A}}(v) \textit{ such that } q_1 \preceq^b q_3 \ \wedge\ q_2 \preceq^{t-x} q_4$$

Example 3.1.33. Let $\mathcal{A}$ be the automaton in Figure 3.9. Observe that $q_2 \preceq^{t-x} q_7$. This is due to the fact that no matter what Spoiler plays from q_2, Duplicator will always be able to match that play. If Spoiler plays $q_2 \overset{w_1}{\to} q_3 \overset{w_2}{\to} q_4$ Duplicator can play $q_7 \overset{w_1}{\to} q_8 \overset{w_2}{\to} q_{10}$. Similarly, if Spoiler plays $q_2 \overset{w_1}{\to} q_3 \overset{w_3}{\to} q_5$, Duplicator can answer $q_7 \overset{w_1}{\to} q_9 \overset{w_3}{\to} q_{11}$. Observe that $ctx_{\mathcal{A}}(u) = \{(q_1, q_2)\}$ and $ctx_{\mathcal{A}}(v) = \{(q_6, q_7)\}$. Since $q_2 \preceq^{t-x} q_7$ and $q_1 \preceq^b q_6$, $\forall (p_1, p_2) \in ctx_{\mathcal{A}}(u)\ \exists (p_3, p_4) \in ctx_{\mathcal{A}}(v)$ such that $p_1 \preceq^b p_3 \ \wedge\ p_2 \preceq^{t-x} p_4$, and then $u \sqsubseteq_{\mathcal{A}}^{t-x} v$.

Observe that $q_3 \preceq^{t-x} q_8$ and $q_3 \preceq^{t-x} q_9$ don't hold due to the fact that $q_8 \overset{w_3}{\not\to}$ and $q_9 \overset{w_2}{\not\to}$. This implies that also $uw_1 \sqsubseteq_{\mathcal{A}}^{t-x} vw_1$ doesn't hold. We showed that $\exists u, v, w_1 \in \Sigma^*$ such that $u \sqsubseteq_{\mathcal{A}}^{t-x} v \implies uw_1 \sqsubseteq_{\mathcal{A}}^{t-x} vw_1$, so that $\sqsubseteq_{\mathcal{A}}^{t-k}$ x *is not right-monotonic*. This implies that for our purposes $\sqsubseteq_{\mathcal{A}}^{t-k} x$ is not interesting.

3.2 Using simulation-based quasiorders to solve the language inclusion problem

In this section we show how to to use the simulation-based quasiorders to solve the language inclusion problem. In particular, in Section 3.2.1 we show which pairs of qos can be used in Algorithm `BAInc` for checking the inclusion between the languages of two Büchi automata, while in Section 3.2.2 we point out that `CFGInc` can be instantiated with $\sqsubseteq^1_{\mathcal{A}}$ to solve the language inclusion problem between CFGs and FAs.

3.2.1 Languages recognized by Büchi automata

We remark that the algorithm that decides the language inclusion between two ω-regular languages is parametrized by two qos $\leq_1, \leq_2$ that must be:

1. computable well-quasiorders;
2. right-monotonic;
3. such that $\rho_{\leq_1 \times \leq_2}(I_{L_2}) = I_{L_2}$.

We observe that the quasiorders defined in Section 3.1 meet the first two requirements. We remark that Proposition 2.5.1 offers an alternative characterization of the third one. Observe that in order to show that $\forall u, s \in \Sigma^*, v, t \in \Sigma^+$ such that $uv^\omega \in L_2$, $u \leq_1 s$ and $v \leq_2 t$, it holds that $st^\omega \in L_2$, we can proceed as follows:

1. We prove $\forall u, s \in \Sigma^*, v \in \Sigma^+$, if $uv^\omega \in L_2$ and $u \leq_1 s$, then $sv^\omega \in L_2$;
2. We prove $\forall u \in \Sigma^*, v, t \in \Sigma^+$, if $uv^\omega \in L_2$ and $v \leq_2 t$, then $ut^\omega \in L_2$.

And then $st^\omega \in L_2$ immediately follows. This is exactly what we show: first we prove that 1 holds for $\sqsubseteq^1_{\mathcal{A}}$, $\sqsubseteq^r_{\mathcal{A}}$, $\sqsubseteq^{de,r}_{\mathcal{A}}$ and $\sqsubseteq^{fair,r}_{\mathcal{A}}$; then, we show that 2 holds for $\sqsubseteq^2_{\mathcal{A}}$. This implies that the pairs $(\sqsubseteq^1_{\mathcal{A}}, \sqsubseteq^2_{\mathcal{A}})$, $(\sqsubseteq^r_{\mathcal{A}}, \sqsubseteq^2_{\mathcal{A}})$, $(\sqsubseteq^{de,r}_{\mathcal{A}}, \sqsubseteq^2_{\mathcal{A}})$ and $(\sqsubseteq^{fair,r}_{\mathcal{A}}, \sqsubseteq^2_{\mathcal{A}})$ can be used in Algorithm `BAInc` to solve the language inclusion problem between ω-regular languages.

Proposition 3.2.1. Let $\mathcal{B}$ be a BA, $u, s \in \Sigma^*, v \in \Sigma^+$ such that $uv^\omega \in \mathcal{L}(\mathcal{B})$ and $u \sqsubseteq^1_{\mathcal{B}} s$. Then, $sv^\omega \in \mathcal{L}(\mathcal{B})$.

Proof. If $uv^\omega \in \mathcal{L}(\mathcal{B})$, then $\exists i \in I, p, q \in Q$ such that $i \overset{u}{\rightsquigarrow} p \overset{v^n}{\rightsquigarrow} q \overset{v^m}{\rightarrowtail} q$ for $n \geq 0, m \geq 1$. Since $(i, p) \in ctx_{\mathcal{B}}(u)$, by $u \sqsubseteq^1_{\mathcal{B}} s$, $\exists (i', p') \in ctx_{\mathcal{B}}(s)$ such that $i \preceq^r i'$ and $p \preceq^{di} p'$. Since $i \in I$ and $i \preceq^r i'$, then $i' \in I$. Furthermore, since from p starts one fair trace that matches the infinite word v^ω, by $p \preceq^{di} p'$, also from p' starts one fair trace that matches the same infinite word. We recall that one trace is fair if $q_f \in F$ is present in the trace infinitely many times. This implies that $sv^\omega \in \mathcal{L}(\mathcal{B})$. □

Proposition 3.2.2. Let $\mathcal{B}$ be a BA, $u, s \in \Sigma^*, v \in \Sigma^+$ such that $uv^\omega \in \mathcal{L}(\mathcal{B})$ and $u \sqsubseteq^r_{\mathcal{B}} s$. Then, $sv^\omega \in \mathcal{L}(\mathcal{B})$.

Proof. Since $uv^\omega \in \mathcal{L}(\mathcal{B})$, $\exists i \in I, p, q \in Q$ such that $i \overset{u}{\rightsquigarrow} p \overset{v^n}{\rightsquigarrow} q \overset{v^m}{\rightarrowtail} q$ for some $n \geq 0, m \geq 1$. By $u \sqsubseteq^r_{\mathcal{B}} s$, $\exists i' \in I, p' \in Q$ such that $i' \overset{s}{\rightsquigarrow} p'$ and $p \preceq^{di} p'$. We observe that from p starts a fair trace $\pi_0 = p \overset{a_1}{\rightarrow} p_1 \overset{a_2}{\rightarrow} \cdots \overset{a_n}{\rightarrow} p_n \overset{a_1}{\rightarrow} p_{n+1} \overset{a_2}{\rightarrow} p_{n+2} \cdots$ where $a_1 \dots a_n = v$. For this reason and by $p \preceq^{di} p'$, also from p' starts a fair trace $\pi_1 = p' \overset{a_1}{\rightarrow} p'_1 \overset{a_2}{\rightarrow} \cdots \overset{a_n}{\rightarrow} p'_n \overset{a_1}{\rightarrow} p'_{n+1} \overset{a_2}{\rightarrow} p'_{n+2} \cdots$, and then $sv^\omega \in \mathcal{L}(\mathcal{B})$. □

Proposition 3.2.3. Let $\mathcal{B}$ be a BA, $u, s \in \Sigma^*, v \in \Sigma^+$ such that $uv^\omega \in \mathcal{L}(\mathcal{B})$ and $u \sqsubseteq^{de,r}_{\mathcal{B}} s$. Then, $sv^\omega \in \mathcal{L}(\mathcal{B})$.

Proof. Since $uv^\omega \in \mathcal{L}(\mathcal{B})$, $\exists i \in I, p, q \in Q$ such that $i \overset{u}{\rightsquigarrow} p \overset{v^n}{\rightsquigarrow} q \overset{v^m}{\rightarrowtail} q$ for some $n \geq 0, m \geq 1$. By $u \sqsubseteq^{de,r}_{\mathcal{B}} s$, $\exists i' \in I, p' \in Q$ such that $i' \overset{s}{\rightsquigarrow} p'$ and $p \preceq^{de} p'$. We observe that from p starts a fair trace $\pi_0 = p \overset{a_1}{\rightarrow} p_1 \overset{a_2}{\rightarrow} \cdots \overset{a_n}{\rightarrow} p_n \overset{a_1}{\rightarrow} p_{n+1} \overset{a_2}{\rightarrow} p_{n+2} \cdots$ where $a_1 \dots a_n = v$. For this reason and by $p \preceq^{de} p'$, also from p' starts a fair trace $\pi_1 = p' \overset{a_1}{\rightarrow} p'_1 \overset{a_2}{\rightarrow} \cdots \overset{a_n}{\rightarrow} p'_n \overset{a_1}{\rightarrow} p'_{n+1} \overset{a_2}{\rightarrow} p'_{n+2} \cdots$, and then $sv^\omega \in \mathcal{L}(\mathcal{B})$. □

Proposition 3.2.4. Let $\mathcal{B}$ be a BA, $u, s \in \Sigma^*, v \in \Sigma^+$ such that $uv^\omega \in \mathcal{L}(\mathcal{B})$ and $u \sqsubseteq^{fair,r}_{\mathcal{B}} s$. Then, $sv^\omega \in \mathcal{L}(\mathcal{B})$.

Proof. Since $uv^\omega \in \mathcal{L}(\mathcal{B})$, $\exists i \in I, p, q \in Q$ such that $i \overset{u}{\rightsquigarrow} p \overset{v^n}{\rightsquigarrow} q \overset{v^m}{\rightarrowtail} q$ for some $n \geq 0, m \geq 1$. By $u \sqsubseteq^{fair,r}_{\mathcal{B}} s$, $\exists i' \in I, p' \in Q$ such that $i' \overset{s}{\rightsquigarrow} p'$ and $p \preceq^{f} p'$. We observe that from p starts a fair trace $\pi_0 = p \overset{a_1}{\rightarrow} p_1 \overset{a_2}{\rightarrow} \cdots \overset{a_n}{\rightarrow} p_n \overset{a_1}{\rightarrow} p_{n+1} \overset{a_2}{\rightarrow} p_{n+2} \cdots$ where $a_1 \dots a_n = v$. For this reason and by $p \preceq^{f} p'$, also from p' starts a fair trace $\pi_1 = p' \overset{a_1}{\rightarrow} p'_1 \overset{a_2}{\rightarrow} \cdots \overset{a_n}{\rightarrow} p'_n \overset{a_1}{\rightarrow} p'_{n+1} \overset{a_2}{\rightarrow} p'_{n+2} \cdots$, and then $sv^\omega \in \mathcal{L}(\mathcal{B})$. □

In order to prove that $\forall u \in \Sigma^*, v, t \in \Sigma^+$ such that $uv^\omega \in \mathcal{L}(\mathcal{B})$ and $v \sqsubseteq^2_{\mathcal{B}} t$, it holds that $ut^\omega \in \mathcal{L}(\mathcal{B})$, we need two preliminary results.

Lemma 3.2.5. Let $\mathcal{A}$ be an automaton. Let $u, v, w \in \Sigma^*$ such that $u \sqsubseteq^2_{\mathcal{A}} v$, $i \in I, p, q \in Q$, such that $i \overset{w}{\rightsquigarrow} p \overset{u^n}{\rightsquigarrow} q$ for $n \geq 1$. Then $\forall n', n'' : n' + n'' = n, n', n'' \geq 0, \exists i' \in I, p'_1, p'_2, q' \in Q$ such that $i' \overset{w}{\rightsquigarrow} p'_1 \overset{v^{n'}}{\rightsquigarrow} p'_2 \overset{u^{n''}}{\rightsquigarrow} q' \ \wedge \ p \preceq^b p'_1 \ \wedge \ q \preceq^{di} q'$.

Proof. It follows from one induction on n'. If $n' = 0$, then $n'' = n$. We observe that $i \overset{w}{\rightsquigarrow} p \overset{\epsilon}{\rightsquigarrow} p \overset{u^{n''}}{\rightsquigarrow} q$, and since $\preceq^b$ and $\preceq^{di}$ are reflexive the thesis holds. If $0 < n' \leq n$, by induction hypothesis $\exists i'' \in I, p''_1, p''_2, p''_3, q'' \in Q$ such that $i'' \overset{w}{\rightsquigarrow} p''_1 \overset{v^{n'-1}}{\rightsquigarrow} p''_2 \overset{u}{\rightsquigarrow} p''_3 \overset{u^{n''}}{\rightsquigarrow} q'' \ \wedge \ p \preceq^b p''_1 \ \wedge \ q \preceq^{di} q''$. Since $(p''_2, p''_3) \in ctx_{\mathcal{A}}(u)$ and $u \sqsubseteq^2_{\mathcal{A}} v$, $\exists (p'_2, p'_3) \in ctx_{\mathcal{A}}(v)$ such that $p''_2 \preceq^b p'_2 \ \wedge \ p''_3 \preceq^{di} p'_3$. $p''_2 \preceq^b p'_2$ implies that $\exists i' \in I, p'_1, \in Q$ such that $i' \overset{w}{\rightsquigarrow} p'_1 \overset{v^{n'-1}}{\rightsquigarrow} p'_2$ and $p''_1 \preceq^b p'_1$. Furthermore $p''_3 \overset{u^{n''}}{\rightsquigarrow} q''$ implies that $\exists q' \in Q, p'_3 \overset{u^{n''}}{\rightsquigarrow} q'$ and $q'' \preceq^{di} q'$. We conclude by observing that by transitivity $p \preceq^b p'_1$ and $q \preceq^{di} q'$. □

Lemma 3.2.6. Let $\mathcal{A}$ be an automaton. Let $u, v, w \in \Sigma^*$ such that $u \sqsubseteq^2_{\mathcal{A}} v$, $i \in I, p_1, p_2, q \in Q$. If $i \overset{w}{\rightsquigarrow} p_1 \overset{u}{\rightarrowtail} p_2 \overset{u^n}{\rightsquigarrow} q$, then $\exists i' \in I, p'_1, p'_2, q' \in Q$ such that $i' \overset{w}{\rightsquigarrow} p'_1 \overset{v}{\rightarrowtail} p'_2 \overset{v^n}{\rightsquigarrow} q' \ \wedge \ p_1 \preceq^b p'_1 \ \wedge \ q \preceq^{di} q'$.

Proof. By $u \sqsubseteq^2_{\mathcal{A}} v$, from $p_1 \overset{u}{\rightarrowtail} p_2$ we obtain that $\exists (p''_1, p''_2) \in ctx_{\mathcal{A}}(v)$ such that $p_1 \preceq^b p''_1 \ \wedge \ p_2 \preceq^{di} p''_2 \ \wedge \ p''_1 \overset{v}{\rightarrowtail} p''_2$. Since $i \overset{w}{\rightsquigarrow} p_1$ and $p_2 \overset{u^n}{\rightsquigarrow} q$ this implies that $\exists i'' \in I, q'' \in Q$ such that $i'' \overset{w}{\rightsquigarrow} p''_1 \overset{v}{\rightarrowtail} p''_2 \overset{u^n}{\rightsquigarrow} q'' \ \wedge \ q \preceq^{di} q''$. By Lemma 3.2.5 $\exists i'_1 \in I, p'_2, q' \in Q$ such that $i'_1 \overset{wv}{\rightsquigarrow} p'_2 \overset{v^n}{\rightsquigarrow} q' \ \wedge \ p''_2 \preceq^b p'_2 \ \wedge \ q'' \preceq^{di} q'$. $p''_2 \preceq^b p'_2$ and $p''_2 \overset{v^R}{\rightarrowtail}_R p''_1$ imply that $\exists p'_1 \in Q$, $p'_2 \overset{v^R}{\rightarrowtail}_R p'_1$. By definition of backward simulation, $p''_1 \preceq^b p''_1$ so that $\exists i'_2 \in I$ such that $i'_2 \overset{w}{\rightsquigarrow} p'_1 \overset{v}{\rightarrowtail} p'_2 \overset{v^n}{\rightsquigarrow} q'$. We conclude by observing that, by transitivity, $p_1 \preceq^b p'_1 \ \wedge \ p_2 \preceq^{di} p'_2$. □

Proposition 3.2.7. Let $\mathcal{B}$ be a BA, $u, \in \Sigma^*, v, t \in \Sigma^+$ such that $uv^\omega \in \mathcal{L}(\mathcal{B})$ and $v \sqsubseteq^2_{\mathcal{B}} t$. Then, $ut^\omega \in \mathcal{L}(\mathcal{B})$.

Proof. The idea is to show that we can keep substituting pairs of states in $ctx_{\mathcal{B}}(v)$ with pairs in $ctx_{\mathcal{B}}(t)$. It turns out that if we do this enough times, by the *pigeonhole principle* we find a loop. Since, $uv^\omega \in \mathcal{L}(\mathcal{B})$ holds, $\exists i \in I, p, q_0 \in Q$ such that $i \overset{u}{\rightsquigarrow} p \overset{v^n}{\rightsquigarrow} q_0 \overset{v^m}{\rightarrowtail} q_0$ for some $n, m \geq 1$ Then $\exists \pi = q_0 \overset{a_0}{\to} q_1 \overset{a_1}{\to} \cdots \overset{a_{\hat{n}}}{\to} q_0$ where $a_0 a_1 \dots a_{\hat{n}} = v^m$. At least one of the states in π is final, say q_i with $i \in [0..\hat{n}]$. Let j, k such that $q_j \overset{x}{\rightarrowtail} q_i \overset{y}{\rightarrowtail} q_k$, where $xy = v$. Observe that $(q_j, q_k) \in ctx^F_{\mathcal{B}}(v)$. Let m' be the least value such that $q_0 \overset{v^{m'}}{\rightsquigarrow} q_j$. It holds that $i \overset{u}{\rightsquigarrow} p \overset{v^n v^{m'}}{\rightsquigarrow} q_j \overset{v}{\rightarrowtail} q_k$. By Lemma 3.2.5 and since $v \sqsubseteq^2_{\mathcal{B}} t$, $\exists i' \in I, p', q'_0 \in Q$ such that $i' \overset{u}{\rightsquigarrow} p' \overset{t^n t^{m'}}{\rightsquigarrow} q'_0 \ \wedge \ q_j \preceq^{di} q'_0$, and then $\exists \hat{q}'_0$ such that $q'_0 \overset{v}{\rightarrowtail} \hat{q}'_0 \ \wedge \ q_k \preceq^{di} \hat{q}'_0$. This implies that $\exists q'_1, \hat{q}'_1, q'_2, \hat{q}'_2, \dots, q'_{|Q|+1}, \hat{q}'_{|Q|+1} \in Q$ such that:

$$i' \overset{u}{\rightsquigarrow} p' \overset{t^n t^{m'}}{\rightsquigarrow} q'_0 \overset{v}{\rightarrowtail} \hat{q}'_0 \overset{v^{m-1}}{\rightsquigarrow} q'_1 \overset{v}{\rightarrowtail} \hat{q}'_1 \overset{v^{m-1}}{\rightsquigarrow} \cdots \overset{v^{m-1}}{\rightsquigarrow} q'_{|Q|} \overset{v}{\rightarrowtail} \hat{q}'_{|Q|} \overset{v^{m-1}}{\rightsquigarrow} q'_{|Q|+1} \overset{v}{\rightarrowtail} \hat{q}'_{|Q|+1}$$

and $q_k \preceq^{di} \hat{q}'_{|Q|+1}$. Applying Lemma 3.2.6 we observe that $\exists i'' \in I, q''_0, \hat{q}''_0, q''_1, \hat{q}''_1, \dots, q''_{|Q|+1}, \hat{q}''_{|Q|+1} \in Q$ such that:

$$i'' \overset{ut^n t^{m'}}{\rightsquigarrow} q''_0 \overset{t}{\rightarrowtail} \hat{q}''_0 \overset{t^{m-1}}{\rightsquigarrow} q''_1 \overset{v}{\rightarrowtail} \hat{q}''_1 \overset{v^{m-1}}{\rightsquigarrow} \cdots \overset{v^{m-1}}{\rightsquigarrow} q''_{|Q|} \overset{v}{\rightarrowtail} \hat{q}''_{|Q|} \overset{v^{m-1}}{\rightsquigarrow} q''_{|Q|+1} \overset{v}{\rightarrowtail} \hat{q}''_{|Q|+1}$$

and $q_k \preceq^{di} \hat{q}''_{|Q|+1}$. Applying Lemma 3.2.6 $|Q|$ more times, we observe that: $\exists i''' \in I, q'''_0, \hat{q}'''_0, q'''_1, \hat{q}'''_1, \dots, q'''_{|Q|+1}, \hat{q}'''_{|Q|+1} \in Q$ such that:

$$i''' \overset{ut^n t^{m'}}{\rightsquigarrow} q'''_0 \overset{t}{\rightarrowtail} \hat{q}'''_0 \overset{t^{m-1}}{\rightsquigarrow} q'''_1 \overset{t}{\rightarrowtail} \hat{q}'''_1 \overset{t^{m-1}}{\rightsquigarrow} \cdots \overset{t^{m-1}}{\rightsquigarrow} q'''_{|Q|} \overset{t}{\rightarrowtail} \hat{q}'''_{|Q|} \overset{t^{m-1}}{\rightsquigarrow} q'''_{|Q|+1} \overset{v}{\rightarrowtail} \hat{q}'''_{|Q|+1}$$

By the *pingeonhole principle*, we can find $\hat{i}, \hat{j} \in [0..|Q|]$ such that $\hat{i} < \hat{j} \ \wedge \ q'''_{\hat{i}} = q'''_{\hat{j}}$. Therefore, $\exists \hat{n}, \hat{m}, i''' \overset{ut^{\hat{n}}}{\rightsquigarrow} q'''_{\hat{i}} \overset{t^{\hat{m}}}{\rightarrowtail} q'''_{\hat{i}}$ so that $ut^\omega \in \mathcal{L}(\mathcal{B})$. □

The following list of propositions summarizes which pairs of simulation-based quasiorders meet the requirement $\rho_{\leq_1 \times \leq_2}(I_{L_2}) = I_{L_2}$.

Proposition 3.2.8. Let $\mathcal{B}$ be a BA.

$$\rho_{\sqsubseteq^1_{\mathcal{B}} \times \sqsubseteq^2_{\mathcal{B}}}(I_{\mathcal{L}(\mathcal{B})}) = I_{\mathcal{L}(\mathcal{B})}$$

Proof. It follows immediately from Proposition 2.5.1, Proposition 3.2.1 and Proposition 3.2.7. □

Proposition 3.2.9. Let $\mathcal{B}$ be a BA.

$$\rho_{\sqsubseteq^r_{\mathcal{B}} \times \sqsubseteq^2_{\mathcal{B}}}(I_{\mathcal{L}(\mathcal{B})}) = I_{\mathcal{L}(\mathcal{B})}$$

Proof. It follows immediately from Proposition 2.5.1, Proposition 3.2.2 and Proposition 3.2.7. □

Proposition 3.2.10. Let $\mathcal{B}$ be a BA.

$$\rho_{\sqsubseteq^{de,r}_{\mathcal{B}} \times \sqsubseteq^2_{\mathcal{B}}}(I_{\mathcal{L}(\mathcal{B})}) = I_{\mathcal{L}(\mathcal{B})}$$

Proof. It follows immediately from Proposition 2.5.1, Proposition 3.2.3 and Proposition 3.2.7. □

Proposition 3.2.11. Let $\mathcal{B}$ be a BA.

$$\rho_{\sqsubseteq_{\mathcal{B}}^{fair,r} \times \sqsubseteq_{\mathcal{B}}^{2}}(I_{\mathcal{L}(\mathcal{B})}) = I_{\mathcal{L}(\mathcal{B})}$$

Proof. It follows immediately from Proposition 2.5.1, Proposition 3.2.4 and Proposition 3.2.7. □

3.2.2 Languages recognized by CFGs and FAs

In what follows, we show that $\sqsubseteq_{\mathcal{A}}^{1}$ meets the requirements of the framework described in [GRV19], and then can be used to instantiate Algorithm `CFGInc` to check le language inclusion between context-free and regular languages.

Proposition 3.2.12. Let $\mathcal{A}$ be an FA.

$$\sqsubseteq_{\mathcal{A}}^{1} \cap (\mathcal{L}(\mathcal{A}) \times \neg\mathcal{L}(\mathcal{A})) = \varnothing$$

Proof. Let $u \in \mathcal{L}(\mathcal{A})$, then $\exists (q_1, q_2) \in ctx_{\mathcal{A}}(u)$ such that $q_1 \in I$ and $q_2 \in F$. Let $v \notin \mathcal{L}(\mathcal{A})$, then $\forall (q_3, q_4) \in ctx_{\mathcal{A}}(v)$, $q_3 \notin I \ \vee q_4 \notin F$ holds. If $q_3 \notin I$, since $q_1 \in I$, then $q_1 \not\preceq^{r} q_3$. If $q_4 \notin F$, since $q_2 \in F$, then $q_2 \not\preceq^{di} q_4$. In both cases $u \sqsubseteq_{\mathcal{A}}^{1} v$ does not hold. □

Proposition 3.2.13. Let $\mathcal{A}$ be a FA.

$$\sqsubseteq_{\mathcal{A}}^{1} \text{ is a } \mathcal{L}(\mathcal{A})\text{-consistent decidable wqo.}$$

Proof. We remark a quasiorder $\leq$ is $\mathcal{L}(\mathcal{A})$-*consistent* iff it is a computable right-monotonic wqo such that $\leq \cap (\mathcal{L}(\mathcal{A}) \times \neg\mathcal{L}(\mathcal{A})) = \varnothing$. Then, thesis follows immediately from Proposition 3.1.3, 3.1.4 and 3.2.12 □

We remark that the only requirement of Algorithm `CFGInc` for the quasiorder $\leq$ is to be L_2-consistent. Since for a FA $\mathcal{A}$, $\sqsubseteq_{\mathcal{A}}^{1}$ is $\mathcal{L}(\mathcal{A})$-consistent, it can be used to instantiate Algorithm `CFGInc` to check the language inclusion between context-free and regular languages.

3.3 Overview of the considered quasiorders

In this section we summarize the properties of the considered quasiorders for the [DG20]'s framework: the newly defined simulation-based qos, the state-based and the syntactic ones. We also discuss some of the relations between them. In what follows let $\mathcal{B}$ be a BA. We recall from Proposition 3.1.15 that:

$$\sqsubseteq_{\mathcal{B}}^{2} \subseteq \sqsubseteq_{\mathcal{B}}^{1}$$

Furthermore, Proposition 3.1.8 states that:

$$\sqsubseteq_{\mathcal{B}}^{1} \subseteq \sqsubseteq_{\mathcal{B}}^{r}$$

We remark that Proposition 3.1.21 states:

$$\sqsubseteq_{\mathcal{B}}^{r} \subseteq \sqsubseteq_{\mathcal{B}}^{de,r}$$

Additionally, Proposition 3.1.29 states:

$$\sqsubseteq_{\mathcal{B}}^{de,r} \subseteq \sqsubseteq_{\mathcal{B}}^{fair,r}$$

Type	Quasiorder	Monotonicity	
Simulation-based	$\sqsubseteq^1_{\mathcal{B}}$	Monotonic	Proposition 3.1.4
	$\sqsubseteq^2_{\mathcal{B}}$	Monotonic	Proposition 3.1.14
	$\sqsubseteq^r_{\mathcal{B}}$	Right-monotonic	[GRV19]
	$\sqsubseteq^{de,r}_{\mathcal{B}}$	Right-monotonic	Proposition 3.1.19
	$\sqsubseteq^{fair,r}_{\mathcal{B}}$	Right-monotonic	Proposition 3.1.27
State-based	$\leq^1_{\mathcal{B}}$	Monotonic	[DG20]
	$\leq^2_{\mathcal{B}}$	Monotonic	[DG20]
	$\leq^r_{\mathcal{B}}$	Right-monotonic	[DG20]
Syntactic	$\leq^1_{\mathcal{L}(\mathcal{B})}$	Monotonic	[DG20]
	$\leq^2_{\mathcal{L}(\mathcal{B})}$	Monotonic	[DG20]
	$\leq^r_{\mathcal{L}(\mathcal{B})}$	Right-monotonic	[DG20]

Table 3.1: Monotonicity properties of the considered qos

Type	Quasiorder	Proof of being a computable well-quasiorder
Simulation-based	$\sqsubseteq^1_{\mathcal{B}}$	Proposition 3.1.3
	$\sqsubseteq^2_{\mathcal{B}}$	Proposition 3.1.13
	$\sqsubseteq^r_{\mathcal{B}}$	[GRV19]
	$\sqsubseteq^{de,r}_{\mathcal{B}}$	Proposition 3.1.18
	$\sqsubseteq^{fair,r}_{\mathcal{B}}$	Proposition 3.1.24
State-based	$\leq^1_{\mathcal{B}}$	[DG20]
	$\leq^2_{\mathcal{B}}$	[DG20]
	$\leq^r_{\mathcal{B}}$	[DG20]
Syntactic	$\leq^1_{\mathcal{L}(\mathcal{B})}$	[DG20]
	$\leq^2_{\mathcal{L}(\mathcal{B})}$	[DG20]
	$\leq^r_{\mathcal{L}(\mathcal{B})}$	[DG20]

Table 3.2: Proofs of being a computable well-quasiorder for the considered qos

Table 3.1 summarizes the monotonicity properties of the considered quasiorders. Table 3.2 summarizes where the reader can find the proofs that the considered quasiorders are computable well-quasiorders.

We recall that two qos $\leq_1, \leq_2$, in order to be used in the framework for checking the language inclusion between ω-regular languages, must meet the following requirements:

1. $\leq_1$ and $\leq_2$ must be computable well-quasiorders;
2. $\leq_1$ and $\leq_2$ must be right-monotonic;
3. It must hold that $\rho_{\leq_1 \times \leq_2}(I_{L_2}) = I_{L_2}$.

While Table 3.1 and Table 3.2 show that the considered quasiorders meet the first two requirements, Table 3.3 summarizes which pairs of qos meet the last one.

We now discuss the relations between the simulation-based and the state-based qos. Observe that $\leq^2_{\mathcal{B}} \subseteq \leq^1_{\mathcal{B}} \subseteq \leq^r_{\mathcal{B}}$ [DG20]. Since $\forall u, v \in \Sigma^*$, if $ctx_{\mathcal{B}}(u) \subseteq ctx_{\mathcal{B}}(v)$, then $\forall (q_1, q_2) \in ctx_{\mathcal{B}}(u)\ \exists (q_3, q_4) \in ctx_{\mathcal{B}}(v)$ such that $q_1 \preceq^r q_3$ and $q_2 \preceq^{di} q_4$, because $\preceq^r$ and $\preceq^{di}$ are reflexive. This implies:

$$\leq^1_{\mathcal{B}} \subseteq \sqsubseteq^1_{\mathcal{B}}$$

Type	$\leq_1$	$\leq_2$	
Simulation-based	$\sqsubseteq^1_{\mathcal{B}}$	$\sqsubseteq^2_{\mathcal{B}}$	Proposition 3.2.8
	$\sqsubseteq^r_{\mathcal{B}}$	$\sqsubseteq^2_{\mathcal{B}}$	Proposition 3.2.9
	$\sqsubseteq^{de,r}_{\mathcal{B}}$	$\sqsubseteq^2_{\mathcal{B}}$	Proposition 3.2.10
	$\sqsubseteq^{fair,r}_{\mathcal{B}}$	$\sqsubseteq^2_{\mathcal{B}}$	Proposition 3.2.11
State-based	$\leq^1_{\mathcal{B}}$	$\leq^2_{\mathcal{B}}$	[DG20]
	$\leq^r_{\mathcal{B}}$	$\leq^2_{\mathcal{B}}$	[DG20]
Syntactic	$\leq^1_{\mathcal{L}(\mathcal{B})}$	$\leq^2_{\mathcal{L}(\mathcal{B})}$	[DG20]
	$\leq^r_{\mathcal{L}(\mathcal{B})}$	$\leq^2_{\mathcal{L}(\mathcal{B})}$	[DG20]

Table 3.3: Pairs of quasiorders that meet the requirement $\rho_{\leq_1 \times \leq_2}(I_{L_2}) = I_{L_2}$

Similarly, $\forall u, v \in \Sigma^*$, if $ctx_{\mathcal{B}}(u) \subseteq ctx_{\mathcal{B}}(v)$ and $ctx^F_{\mathcal{B}}(u) \subseteq ctx^F_{\mathcal{B}}(v)$ then $\forall (q_1, q_2) \in ctx_{\mathcal{B}}(u)$ $\exists (q_3, q_4) \in ctx_{\mathcal{B}}(v)$ such that $q_1 \preceq^b q_3$ and $q_2 \preceq^{di} q_4$ again because $\preceq^b$ and $\preceq^{di}$ are reflexive. Furthermore, $q_1 \overset{u}{\rightarrowtail} q_2$ implies $(q_1, q_2) \in ctx^F_{\mathcal{B}}(u)$, so that $(q_3, q_4) \in ctx^F_{\mathcal{B}}(v)$. This implies:

$$\leq^2_{\mathcal{B}} \subseteq \sqsubseteq^2_{\mathcal{B}}$$

Lastly, $\forall u, v \in \Sigma^*$, if $post^{\mathcal{B}}_u(I) \subseteq post^{\mathcal{B}}_v(I)$ we observe that $\forall p \in post^{\mathcal{B}}_u(I), \exists q \in post^{\mathcal{B}}_v(I)$ such that $p \preceq^{di} q$ again by reflexivity of $\preceq^{di}$, so that:

$$\leq^r_{\mathcal{B}} \subseteq \sqsubseteq^r_{\mathcal{B}}$$

For analogous arguments it holds that $\leq^r_{\mathcal{B}} \subseteq \sqsubseteq^{de,r}_{\mathcal{B}}$ and $\leq^r_{\mathcal{B}} \subseteq \sqsubseteq^{fair,r}_{\mathcal{B}}$.

We now discuss the relations between the simulation-based qos and the syntactic qos. Observe that $\leq^2_{\mathcal{L}(\mathcal{B})} \subseteq \leq^1_{\mathcal{L}(\mathcal{B})} \subseteq \leq^r_{\mathcal{L}(\mathcal{B})}$ [DG20]. First, Table 3.4 summarizes the fact that all the proposed pairs of simulation-based qos *cover* the language of a BA.

$\leq_1$	$\leq_2$	**Coverage of $\mathcal{L}(\mathcal{B})$**
$\sqsubseteq^1_{\mathcal{B}}$	$\sqsubseteq^2_{\mathcal{B}}$	Proposition 2.5.4, Proposition 3.2.1 and Proposition 3.2.7
$\sqsubseteq^r_{\mathcal{B}}$	$\sqsubseteq^2_{\mathcal{B}}$	Proposition 2.5.4, Proposition 3.2.2 and Proposition 3.2.7
$\sqsubseteq^{de,r}_{\mathcal{B}}$	$\sqsubseteq^2_{\mathcal{B}}$	Proposition 2.5.4, Proposition 3.2.3 and Proposition 3.2.7
$\sqsubseteq^{fair,r}_{\mathcal{B}}$	$\sqsubseteq^2_{\mathcal{B}}$	Proposition 2.5.4, Proposition 3.2.4 and Proposition 3.2.7

Table 3.4: Coverage properties of the pairs of simulation-based qos

By the monotonicity properties of the simulation-based qos, their relations and the fact that they cover the language of a BA, by Proposition 2.5.3, we can infer the following relations:

$$\sqsubseteq^1_{\mathcal{B}} \subseteq \leq^1_{\mathcal{L}(\mathcal{B})}$$
$$\sqsubseteq^2_{\mathcal{B}} \subseteq \leq^2_{\mathcal{L}(\mathcal{B})}$$
$$\sqsubseteq^r_{\mathcal{B}} \subseteq \leq^r_{\mathcal{L}(\mathcal{B})}$$
$$\sqsubseteq^{de,r}_{\mathcal{B}} \subseteq \leq^r_{\mathcal{L}(\mathcal{B})}$$
$$\sqsubseteq^{fair,r}_{\mathcal{B}} \subseteq \leq^r_{\mathcal{L}(\mathcal{B})}$$

Finally, Figure 3.10 summarizes the relations between the considered quasiorders. One arrow from one qo to another means that the former is a subset of the latter. Observe that some arrows are not strictly necessary, but we included them to be more clear.

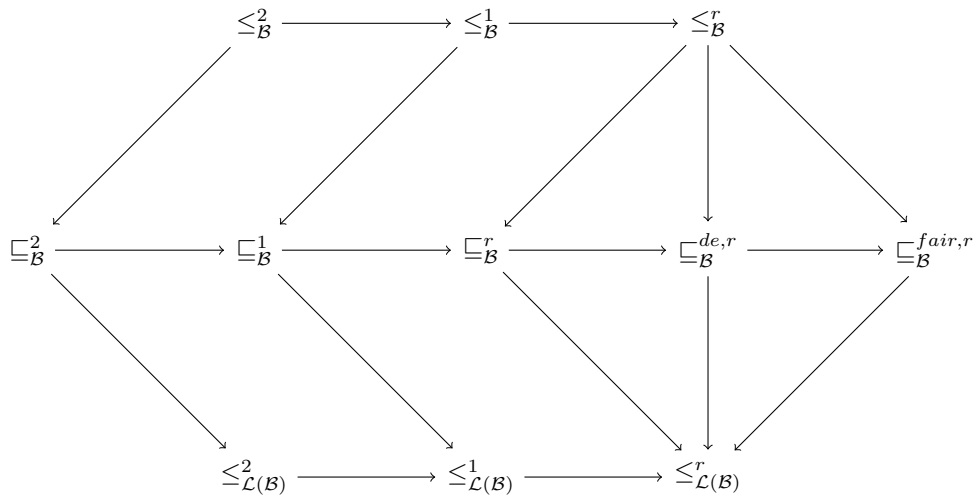

Figure 3.10: Relations between the considered quasiorders

Chapter 4

Illustrative examples

In this chapter we compare, by means of some examples, a number of quasiorders. Since we considered a large number of preorders on Σ^*, here we present only the most interesting comparisons. Observe that the contribution given by the qos to the algorithm presented in [DG20] depends if they are used in the subprocedure `BAPrefixes` or in `BAPeriods`. In the former case, one qo $\leq_1$ is used only during the computation of $D^{N_1}_{1,i}(\emptyset)$, where $N_1 \in \mathbb{N}$ is the least value such that $(D^{N_1+1}_{1,i}(\emptyset))_q \leq_1^{\forall\exists} (D^{N_1}_{1,i}(\emptyset))_q$ for all $q \in Q$ (see Section 2.5.1). Similarly, one qo $\leq_2$ is used only during the computation of $D^{N_2}_{2,p}(\emptyset)$, where $N_2 \in \mathbb{N}$ the least value such that $(D^{N_1+1}_{2,p}(\emptyset))_q \leq_2^{\forall\exists} (D^{N_1}_{2,p}(\emptyset))_q$ for all $q \in Q$ and $p \in F$. For this reason, in the examples we omit the execution of the whole algorithm and we focus either on the computation of $D^{N_1}_{1,i}(\emptyset)$ or $D^{N_2}_{2,p}(\emptyset)$, depending if we are considering qos for prefixes or for periods.

Let $\leq$ be a right-monotonic qo on words. We denote by $\mathscr{A}_{1,i}$ the function that associates $\leq$ with the least $N_1 \in \mathbb{N}$ such that $(D^{N_1+1}_{1,i}(\emptyset))_q \leq^{\forall\exists} (D^{N_1}_{1,i}(\emptyset))_q$ for all $q \in Q$, i.e. $\mathscr{A}_{1,i}(\leq) = N_1$. Similarly, we denote by $\mathscr{A}_{2,p}$ the function that associates $\leq$ with the least $N_2 \in \mathbb{N}$ such that $(D^{N_2+1}_{2,p}(\emptyset))_q \leq^{\forall\exists} (D^{N_2}_{2,p}(\emptyset))_q$ for all $q \in Q$ and for all $p \in F$, i.e. $\mathscr{A}_{2,p}(\leq) = N_2$. Observe that $\mathscr{A}_{1,i}$ and $\mathscr{A}_{2,p}$ return the number of iterations of respectively `BAPrefixes` and `BAPeriods`, and hence we will use them as a metric to compare the qos.

Let $\leq_1$ and $\leq_2$ two right-monotonic qos used for prefixes (periods). We recall that if $\leq_1 \subseteq \leq_2$, then $\mathscr{A}_{1,i}(\leq_2) \leq \mathscr{A}_{1,i}(\leq_1)$ ($\mathscr{A}_{2,p}(\leq_2) \leq \mathscr{A}_{2,p}(\leq_1)$). We now give practical examples that show that coarser preorders allow the subprocedures `BAPrefixes` and `BAPeriods` to converge in less iterations.

4.1 Prefixes

Example 4.1.1. Let $\mathcal{B}$ be the automaton in Figure 4.1. We compare $\leq^1_{\mathcal{B}}$ with $\sqsubseteq^1_{\mathcal{B}}$. We remark that that $\leq^1_{\mathcal{B}} \subseteq \sqsubseteq^1_{\mathcal{B}}$, because $\forall u, v \in \Sigma^*$, $ctx_{\mathcal{B}}(u) \subseteq ctx_{\mathcal{B}}(v) \implies \forall (q_1, q_2) \in ctx_{\mathcal{B}}(u), \exists (q_3, q_4) \in ctx_{\mathcal{B}}(v)$ such that $q_1 \preceq^r q_3$ and $q_2 \preceq^{di} q_4$. This is due to the fact that $\preceq^r$ and $\preceq^{di}$ are reflexive. Table 4.1 illustrates the first four *Kleene's iterates* of the function D_{1,q_0}. Consider $(D^3_{1,q_0}(\emptyset))_{q_1}$. We observe that $a \leq^1_{\mathcal{B}} bd$ doesn't hold: $ctx_{\mathcal{B}}(a) = \{(q_0, q_1), (q_0, q_3)\}$ and $ctx_{\mathcal{B}}(bc) = \{(q_0, q_1), (q_0, q_5)\}$ so that $ctx_{\mathcal{B}}(a) \subsetneq ctx_{\mathcal{B}}(bc)$. Since $\exists u \in (D^3_{1,q_0}(\emptyset))_{q_1}$ such that $\nexists v \in (D^2_{1,q_0}(\emptyset))_{q_1}$ such that $v \leq^1_{\mathcal{B}} u$, $\mathscr{A}_{1,q_0}(\leq^1_{\mathcal{B}}) > 1$.

We observe that $a \sqsubseteq^1_{\mathcal{B}} bc$: in fact $q_0 \preceq^r q_0$, $q_1 \preceq^{di} q_1$ and $q_3 \preceq^{di} q_5$. For this reason,

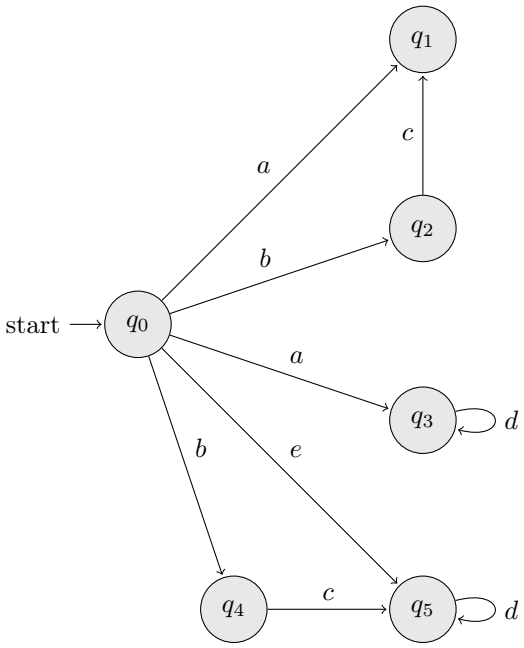

Figure 4.1: Automaton used to compare $\preceq^1_{\mathcal{B}}$ and $\sqsubseteq^1_{\mathcal{B}}$.

$\forall u \in (D^3_{1,q_0}(\emptyset))_{q_2}$, $\exists v \in (D^2_{1,q_0}(\emptyset))_{q_2}$ such that $v \sqsubseteq^{fair,r}_{\mathcal{B}} u$. The reader can also verify that $a \sqsubseteq^1_{\mathcal{B}} ad$, hence $\forall u \in (D^3_{1,q_0}(\emptyset))_{q_3}$, $\exists v \in (D^2_{1,q_0}(\emptyset))_{q_3}$ such that $v \sqsubseteq^1_{\mathcal{B}} u$. Similarly, $e \sqsubseteq^1_{\mathcal{B}} bc$ and $e \sqsubseteq^1_{\mathcal{B}} ed$, so that $\forall u \in (D^3_{1,q_0}(\emptyset))_{q_5}$, $\exists v \in (D^2_{1,q_0}(\emptyset))_{q_5}$ such that $v \sqsubseteq^1_{\mathcal{B}} u$. This implies that $\mathscr{A}_{1,q_0}(\sqsubseteq^1_{\mathcal{B}}) = 2$. It is also possible to show that $\mathscr{A}_{1,q_0}(\preceq^1_{\mathcal{B}}) = 3$. This example shows that for the automaton $\mathcal{B}$, `BAPrefixes(`$\mathcal{B}$`,`$\sqsubseteq^1_{\mathcal{B}}$`)` converges in strictly less iterations than `BAPrefixes(`$\mathcal{B}$`,`$\preceq^1_{\mathcal{B}}$`)`.

	q_0	q_1	q_2	q_3	q_4	q_5
$D^0_{1,q_0}(\emptyset)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$D^1_{1,q_0}(\emptyset)$	$\{\epsilon\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$D^2_{1,q_0}(\emptyset)$	$\{\epsilon\}$	$\{a\}$	$\{b\}$	$\{a\}$	$\{b\}$	$\{e\}$
$D^3_{1,q_0}(\emptyset)$	$\{\epsilon\}$	$\{a, bc\}$	$\{b\}$	$\{a, ad\}$	$\{b\}$	$\{e, ed, bc\}$
$D^4_{1,q_0}(\emptyset)$	$\{\epsilon\}$	$\{a, bc\}$	$\{b\}$	$\{a, ad, add\}$	$\{b\}$	$\{e, ed, edd, bc, bcd\}$

Table 4.1: The first four Kleene's iterates of D_{1,q_0} on the automaton in Figure 4.1

Example 4.1.2. Let $\mathcal{B}$ be the automaton in Figure 4.2. We compare $\preceq^r_{\mathcal{B}}$ with $\sqsubseteq^r_{\mathcal{B}}$. We remark that $\preceq^r_{\mathcal{B}} \subseteq \sqsubseteq^r_{\mathcal{B}}$, because $\forall u, v \in \Sigma^*$, $post^{\mathcal{B}}_u(I) \subseteq post^{\mathcal{B}}_v(I) \implies \forall q \in post^{\mathcal{B}}_u(I)\ \exists p \in post^{\mathcal{B}}_v(I)$ such that $q \preceq^{di} p$. This is due to the fact that $\preceq^{di}$ is reflexive. Table 4.2 illustrates the first five *Kleene's iterates* of the function D_{1,q_0}. Consider $(D^4_{1,q_0}(\emptyset))_{q_2}$. Observe that $post^{\mathcal{B}}_{def}(I) = \{q_2, q_5\}$, while $post^{\mathcal{B}}_{ab}(I) = \{q_2, q_7\}$, so that $ab \preceq^r_{\mathcal{B}} def$ doesn't hold. For this reason, $\nexists u \in (D^3_{1,q_0}(\emptyset))_{q_2}$ such that $u \preceq^r_{\mathcal{B}} def$. Observing that $def \in (D^4_{1,q_0}(\emptyset))_{q_2}$, we conclude that $\mathscr{A}_{1,q_0}(\preceq^r_{\mathcal{B}}) > 3$.

On the other hand, $\forall p \in post^{\mathcal{B}}_{ab}(I)$, $\exists q \in post^{\mathcal{B}}_{def}(I)$ such that $p \preceq^{di} q$: in fact $q_2 \preceq^{di} q_2$ and $q_7 \preceq^{di} q_2$. This implies $ab \preceq^{di} def$, and then $\forall u \in (D^4_{1,q_0}(\emptyset))_{q_2}$,

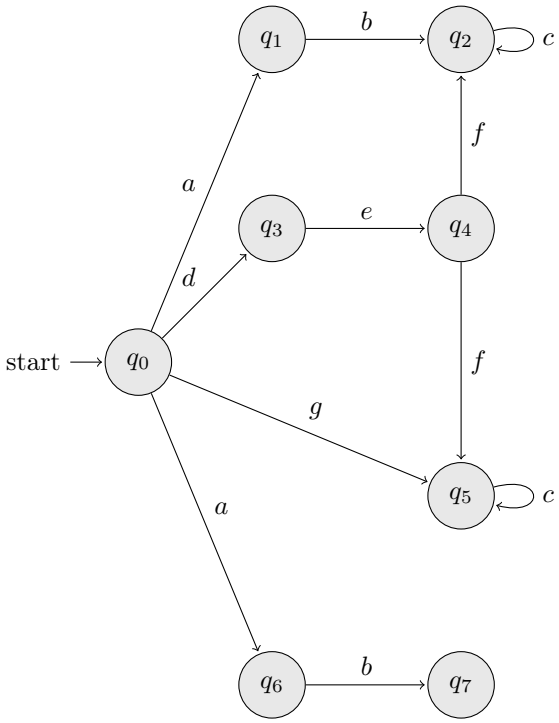

Figure 4.2: Automaton used to compare $\leq^r_{\mathcal{B}}$ and $\sqsubseteq^r_{\mathcal{B}}$.

$\exists v \in (D^3_{1,q_0}(\emptyset))_{q_2}$ such that $v \sqsubseteq^r_{\mathcal{B}} u$. The reader can also verify that $\forall u \in (D^4_{1,q_0}(\emptyset))_{q_5}$, $\exists v \in (D^3_{1,q_0}(\emptyset))_{q_5}$ such that $v \sqsubseteq^r_{\mathcal{B}} u$. This implies that $\mathscr{A}_{1,q_0}(\sqsubseteq^r_{\mathcal{B}}) = 3$. It is also possible to show that $\mathscr{A}_{1,q_0}(\leq^r_{\mathcal{B}}) = 4$. This example shows that `BAPrefixes(`$\mathcal{B}$`,`$\sqsubseteq^r_{\mathcal{B}}$`)` converges in strictly less iterations than `BAPrefixes(`$\mathcal{B}$`,`$\leq^r_{\mathcal{B}}$`)`.

	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7
$D^0_{1,q_0}(\emptyset)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$D^1_{1,q_0}(\emptyset)$	$\{\epsilon\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$D^2_{1,q_0}(\emptyset)$	$\{\epsilon\}$	$\{a\}$	$\emptyset$	$\{d\}$	$\emptyset$	$\{g\}$	$\{a\}$	$\emptyset$
$D^3_{1,q_0}(\emptyset)$	$\{\epsilon\}$	$\{a\}$	$\{ab\}$	$\{d\}$	$\{de\}$	$\{g, gc\}$	$\{a\}$	$\{ab\}$
$D^4_{1,q_0}(\emptyset)$	$\{\epsilon\}$	$\{a\}$	$\{ab, abc, def\}$	$\{d\}$	$\{de\}$	$\{g, gc, gcc, def\}$	$\{a\}$	$\{ab\}$
$D^5_{1,q_0}(\emptyset)$	$\{\epsilon\}$	$\{a\}$	$\{ab, abc, abcc, def\}$	$\{d\}$	$\{de\}$	$\{g, gc, gcc, gccc, def\}$	$\{a\}$	$\{ab\}$

Table 4.2: The first five Kleene's iterates of D_{1,q_0} on the automaton in Figure 4.2

Example 4.1.3. Let $\mathcal{B}$ be the automaton in Figure 4.3. We compare $\sqsubseteq^r_{\mathcal{B}}$ with $\sqsubseteq^{de,r}_{\mathcal{B}}$. We remark that $\sqsubseteq^r_{\mathcal{B}} \subseteq \sqsubseteq^{de,r}_{\mathcal{B}}$, because $\forall u, v \in \Sigma^*$, if $\forall p \in post^{\mathcal{B}}_u(I), \exists q \in post^{\mathcal{B}}_v(I)$ such that $p \preceq^{di} q$, then it also holds that $p \preceq^{de} q$. This is due to the fact that $\preceq^{di} \subseteq \preceq^{de}$ (see Section 2.4). Table 4.3 illustrates the first four *Kleene's iterates* of the function D_{1,q_0}. Consider $(D^3_{1,q_0}(\emptyset))_{q_2}$. We observe that $c \sqsubseteq^r_{\mathcal{B}} ab$ doesn't hold: $post^{\mathcal{B}}_c(I) = \{q_2, q_3\}$, $post^{\mathcal{B}}_{ab}(I) = \{q_2\}$ and since q_2 is not a final state $q_3 \preceq^{di} q_2$ doesn't hold. This implies that $\mathscr{A}_{1,q_0}(\sqsubseteq^r_{\mathcal{B}}) > 2$.

We observe that $c \sqsubseteq^{de,r}_{\mathcal{B}} ab$: in fact $q_3 \preceq^{de} q_2$. From q_3 starts the infinite (and fair)

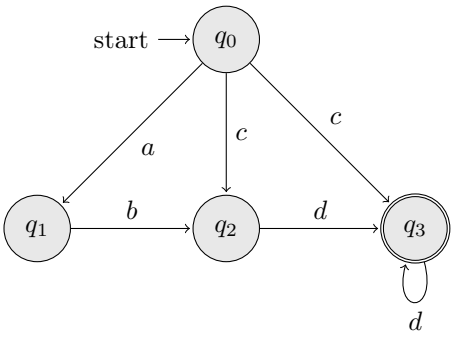

Figure 4.3: Automaton used to compare $\sqsubseteq^{r}_{\mathcal{B}}$ and $\sqsubseteq^{de,r}_{\mathcal{B}}$.

trace $\pi_0 = q_3 \xrightarrow{d} q_3 \xrightarrow{d} \cdots$ in which each state is final. From q_2 starts the infinite (and fair) trace $\pi_1 = q_2 \xrightarrow{d} q_3 \xrightarrow{d} \cdots$ that matches π_0. In particular, for each final state in π_0 there exists a corresponding final state in π_1.

It also holds that $c \sqsubseteq^{de,r}_{\mathcal{B}} cd$: $post^{\mathcal{B}}_{c}(I) = \{q_2, q_3\}$, $post^{\mathcal{B}}_{cd}(I) = \{q_3\}$, $q_2 \preceq^{de} q_3$ and $q_3 \preceq^{de} q_3$. This implies that $\mathscr{A}_{1,q_0}(\sqsubseteq^{de,r}_{\mathcal{B}}) = 2$. It is also possible to show that $\mathscr{A}_{1,q_0}(\sqsubseteq^{r}_{\mathcal{B}}) = 3$. This example shows that for the automaton $\mathcal{B}$, `BAPrefixes`$(\mathcal{B}, \sqsubseteq^{de,r}_{\mathcal{B}})$ converges in strictly less iterations than `BAPrefixes`$(\mathcal{B}, \sqsubseteq^{r}_{\mathcal{B}})$.

	q_0	q_1	q_2	q_3
$D^0_{1,q_0}(\emptyset)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$D^1_{1,q_0}(\emptyset)$	$\{\epsilon\}$	$\emptyset$	$\emptyset$	$\emptyset$
$D^2_{1,q_0}(\emptyset)$	$\{\epsilon\}$	$\{a\}$	$\{c\}$	$\{c\}$
$D^3_{1,q_0}(\emptyset)$	$\{\epsilon\}$	$\{a\}$	$\{c, ab\}$	$\{c, cd\}$
$D^4_{1,q_0}(\emptyset)$	$\{\epsilon\}$	$\{a\}$	$\{c, ab\}$	$\{c, cd, cdd\}$

Table 4.3: The first five Kleene's iterates of D_{1,q_0} on the automaton in Figure 4.3

Example 4.1.4. Let $\mathcal{B}$ be the automaton in Figure 4.4. We compare $\sqsubseteq^{de,r}_{\mathcal{B}}$ with $\sqsubseteq^{fair,r}_{\mathcal{B}}$. We remark that $\sqsubseteq^{de,r}_{\mathcal{B}} \subseteq \sqsubseteq^{fair,r}_{\mathcal{B}}$, because $\forall u, v \in \Sigma^*$, if $\forall p \in post^{\mathcal{B}}_{u}(I), \exists q \in post^{\mathcal{B}}_{v}(I)$ such that $p \preceq^{de} q$, then it also holds that $p \preceq^{f} q$. This is due to the fact that $\preceq^{de} \subseteq \preceq^{f}$ (see Section 2.4). Table 4.4 illustrates the first four *Kleene's iterates* of the function D_{1,q_0}. Consider $(D^3_{1,q_0}(\emptyset))_{q_2}$. We observe that $ab \sqsubseteq^{de,r}_{\mathcal{B}} def$ doesn't hold: $post^{\mathcal{B}}_{ab}(I) = \{q_2, q_6\}$, $post^{\mathcal{B}}_{def}(I) = \{q_2, q_7\}$ and while $q_2 \preceq^{de} q_2$, it is not true that $q_6 \preceq^{de} q_2$ or $q_6 \preceq^{de} q_7$. This is due to the fact that the trace $\pi_0 = q_6 \xrightarrow{a} q_7 \xrightarrow{a} \cdots$ contains a final state, and the only trace $\pi_1 = q_7 \xrightarrow{a} q_7 \xrightarrow{a} \cdots$ that starts from q_7 that can match the a's contains no final state. This implies that $\mathscr{A}_{1,q_0}(\sqsubseteq^{de,r}_{\mathcal{B}}) > 2$.

We observe that $ab \sqsubseteq^{fair,r}_{\mathcal{B}} def$: in fact $q_6 \preceq^{f} q_7$. From q_6 starts the trace π_0, but since it doesn't contain one infinite number of final states it is not fair. This implies that since q_7 can match π_0 with π_1, $q_6 \preceq^{f} q_7$. For this reason, $ab \sqsubseteq^{fair,r}_{\mathcal{B}} def$ so that $\forall u \in (D^3_{1,q_0}(\emptyset))_{q_2}$, $\exists v \in (D^2_{1,q_0}(\emptyset))_{q_2}$ such that $v \sqsubseteq^{fair,r}_{\mathcal{B}} u$. The reader can also verify that $g \sqsubseteq^{fair,r}_{\mathcal{B}} gaa$, $g \sqsubseteq^{fair,r}_{\mathcal{B}} aba$ and $g \sqsubseteq^{fair,r}_{\mathcal{B}} def$, hence $\forall u \in (D^3_{1,q_0}(\emptyset))_{q_7}$, $\exists v \in (D^2_{1,q_0}(\emptyset))_{q_7}$ such that $v \sqsubseteq^{fair,r}_{\mathcal{B}} u$. This implies that $\mathscr{A}_{1,q_0}(\sqsubseteq^{fair,r}_{\mathcal{B}}) = 2$. It is also possible to show that $\mathscr{A}_{1,q_0}(\sqsubseteq^{de,r}_{\mathcal{B}}) = 3$. This example shows that for

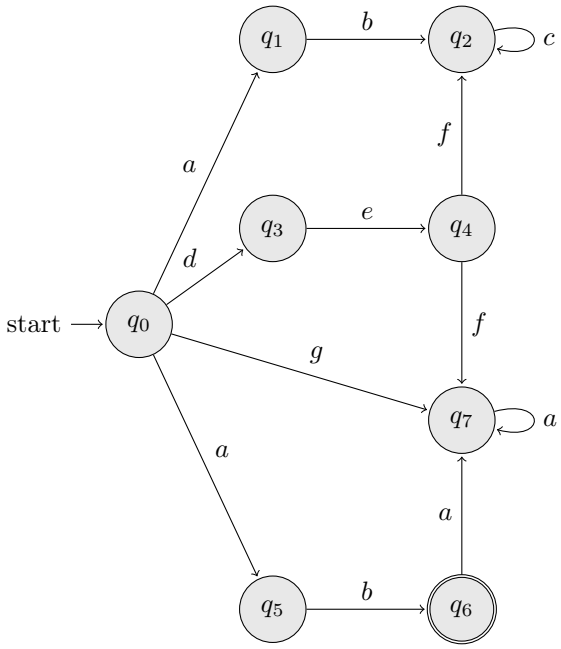

Figure 4.4: Automaton used to compare $\sqsubseteq^{de,r}_{\mathcal{B}}$ and $\sqsubseteq^{fair,r}_{\mathcal{B}}$.

the automaton $\mathcal{B}$, `BAPrefixes`$(\mathcal{B}, \sqsubseteq^{fair,r}_{\mathcal{B}})$ converges in strictly less iterations than `BAPrefixes`$(\mathcal{B}, \sqsubseteq^{de,r}_{\mathcal{B}})$.

	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7
$D^0_{1,q_0}(\emptyset)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$D^1_{1,q_0}(\emptyset)$	$\{\epsilon\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$D^2_{1,q_0}(\emptyset)$	$\{\epsilon\}$	$\{a\}$	$\emptyset$	$\{d\}$	$\emptyset$	$\{a\}$	$\emptyset$	$\{g\}$
$D^2_{1,q_0}(\emptyset)$	$\{\epsilon\}$	$\{a\}$	$\{ab\}$	$\{d\}$	$\{de\}$	$\{a\}$	$\{ab\}$	$\{g, ga\}$
$D^3_{1,q_0}(\emptyset)$	$\{\epsilon\}$	$\{a\}$	$\{ab, abc, def\}$	$\{d\}$	$\{de\}$	$\{a\}$	$\{ab\}$	$\{g, ga, gaa, aba, def\}$
$D^4_{1,q_0}(\emptyset)$	$\{\epsilon\}$	$\{a\}$	$\{ab, abc, abcc, def, defc\}$	$\{d\}$	$\{de\}$	$\{a\}$	$\{ab\}$	$\{g, ga, gaa, gaaa, aba, abaa, def, defa\}$

Table 4.4: The first four Kleene's iterates of D_{1,q_0} on the automaton in Figure 4.4

4.2 Periods

Example 4.2.1. Let $\mathcal{B}$ be the automaton in Figure 4.5. We compare $\leq^2_{\mathcal{B}}$ with $\sqsubseteq^2_{\mathcal{B}}$. We remark that $\leq^2_{\mathcal{B}} \subseteq \sqsubseteq^2_{\mathcal{B}}$, because $\forall u, v \in \Sigma^*$, if $ctx_{\mathcal{B}}(u) \subseteq ctx_{\mathcal{B}}(v)$ and $ctx^F_{\mathcal{B}}(u) \subseteq ctx^F_{\mathcal{B}}(v)$ then $\forall (q_1, q_2) \in ctx_{\mathcal{B}}(u)\ \exists (q_3, q_4) \in ctx_{\mathcal{B}}(v)$ such that $q_1 \preceq^b q_3$ and $q_2 \preceq^{di} q_4$. In particular, consider $q_3 = q_1$ and $q_4 = q_2$. Furthermore, $q_1 \overset{u}{\rightarrowtail} q_2$ implies $(q_1, q_2) \in ctx^F_{\mathcal{B}}(u)$, so that $(q_3, q_4) \in ctx^F_{\mathcal{B}}(v)$. Table 4.5 illustrates the first three *Kleene's iterates* of the function D_{2,q_6}. Consider $(D^2_{2,q_6}(\emptyset))_{q_7}$. We observe that $b \leq^2_{\mathcal{B}} de$ doesn't hold: $ctx_{\mathcal{B}}(b) = \{(q_1, q_2), (q_6, q_7)\}$ and $ctx_{\mathcal{B}}(de) = \{(q_3, q_5), (q_6, q_7)\}$. For this reason, $\exists u \in (D^2_{2,q_6}(\emptyset))_{q_7}$ such that $\nexists v \in (D^1_{2,q_6}(\emptyset))_{q_7}$ such that $v \leq^2_{\mathcal{B}} u$. This

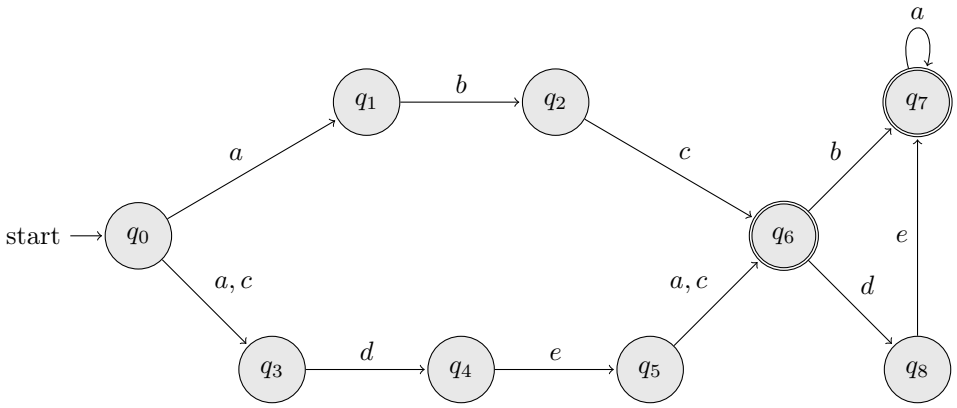

Figure 4.5: Automaton used to compare $\leq^2_{\mathcal{B}}$ and $\sqsubseteq^2_{\mathcal{B}}$.

implies that $\mathscr{A}_{2,q_6}(\leq^2_{\mathcal{B}}) > 1$.

We observe that $b \sqsubseteq^2_{\mathcal{B}} de$. This is due to the fact that $q_1 \preceq^b q_3$, $q_2 \preceq^{di} q_5$, $q_6 \preceq^b q_6$ and $q_7 \preceq^{di} q_7$. For this reason, $\forall u \in (D^2_{2,q_6}(\emptyset))_{q_7}$, $\exists v \in (D^1_{2,q_6}(\emptyset))_{q_7}$ such that $v \sqsubseteq^2_{\mathcal{B}} u$. This implies that $\mathscr{A}_{2,q_6}(\sqsubseteq^2_{\mathcal{B}}) = 1$. It is also possible to show that $\mathscr{A}_{2,q_6}(\leq^2_{\mathcal{B}}) = 2$. This example shows that for the automaton $\mathcal{B}$, `BAPeriods(`$\mathcal{B}$`,` $\leq^2_{\mathcal{B}}$`)` converges in strictly less iterations than `BAPeriods(`$\mathcal{B}$`,`$\sqsubseteq^2_{\mathcal{B}}$`)`.

	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
$D^0_{2,q_6}(\emptyset)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$D^1_{2,q_6}(\emptyset)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{b\}$	$\{d\}$
$D^2_{2,q_6}(\emptyset)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{b, ba, de\}$	$\{d\}$
$D^3_{2,q_6}(\emptyset)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{b, ba, baa, de, dea\}$	$\{d\}$

Table 4.5: The first three Kleene's iterates of D_{2,q_6} on the automaton in Figure 4.5

Chapter 5

Conclusion

We have defined four suitable different quasiorders on words that are based on a number of simulation relations. We also proved that the pairs $(\sqsubseteq^1_{\mathcal{A}}, \sqsubseteq^2_{\mathcal{A}})$, $(\sqsubseteq^r_{\mathcal{A}}, \sqsubseteq^2_{\mathcal{A}})$, $(\sqsubseteq^{de,r}_{\mathcal{A}}, \sqsubseteq^2_{\mathcal{A}})$ and $(\sqsubseteq^{fair,r}_{\mathcal{A}}, \sqsubseteq^2_{\mathcal{A}})$ can be plugged in the framework described in [DG20] in order to solve the language inclusion problem for ω-regular languages. It turns out that the newly defined quasiorders are *coarser* than the state-based, while being *finer* than the syntactic ones. By means of some examples we have shown the advantage of using coarser relations in the algorithm to check the language inclusion. In particular, there is evidence that coarser relations lead to a smaller number of iterations in the procedures `BAPrefixes` and `BAPeriods`.

We believe that, even if we considered numerous different simulation relations on states, many more could be used to build new quasiorders on words. The most promising continuation is to embed delayed and fair simulations in qos used in the procedure `BAPeriods`. We also remark that even though the k-lookahead simulations and the trace inclusions did not lead to suitable qos with the straightforward approach, this does not mean that they cannot be used in some brighter ways to define right-monotonic relations. We mention that there are a lot of well-known simulations, and possibly each one can be used to define suitable coarser qos on words. For example, the *multipebble* and the *fixed-words* simulations [CM17] are coarser than the simulations that we used, but finer than the trace inclusions.

The natural extension of this work would be to provide an implementation for the framework. This would allow us to compare the practical performance of the algorithm using different pairs of qos. In fact, we used as primary metric for measuring the performance the number of iterations of the procedures `BAPrefixes` and `BAPeriods`, while in practice several factors impact on the efficiency of the algorithm. In the past, algorithms based on simulations achieved excellent results, for example in [BP13] they use the *bisimulation up to conguence* in order to check the inclusion between the languages of two FAs. They devise an optimisation of the classical algorithm by Hopcroft and Karp [HK73], and their approach exponentially improves the performance of the *antichain algorithm* [DW+06]. We believe that in the same way they have been able to achieve such a valuable result using simulations, the [DG20]'s framework combined with the simulation-based quasiorders, if properly implemented, can lead to a tool that may even compete with the existing state of the art implementations to solve the language inclusion problem for ω-regular languages.

Bibliography

[Abd+10] Parosh Aziz Abdulla et al. "When simulation meets antichains". In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer. 2010, pp. 158–174 (cit. on p. 8).

[Abd+11] Parosh Aziz Abdulla et al. "Advanced Ramsey-based Büchi automata inclusion testing". In: *International Conference on Concurrency Theory*. Springer. 2011, pp. 187–202 (cit. on pp. 1, 20, 22).

[BG03] Doron Bustan and Orna Grumberg. "Simulation-based minimization". In: *ACM Transactions on Computational Logic (TOCL)* 4.2 (2003), pp. 181–206 (cit. on p. 16).

[BP13] Filippo Bonchi and Damien Pous. "Checking NFA equivalence with bisimulations up to congruence". In: *ACM SIGPLAN Notices* 48.1 (2013), pp. 457–468 (cit. on p. 55).

[CC77] Patrick Cousot and Radhia Cousot. "Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints". In: *Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*. 1977, pp. 238–252 (cit. on pp. 1, 10).

[CC79] Patrick Cousot and Radhia Cousot. "Systematic design of program analysis frameworks". In: *Proceedings of the 6th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*. 1979, pp. 269–282 (cit. on p. 23).

[Céc17] Gérard Cécé. "Foundation for a series of efficient simulation algorithms". In: *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE. 2017, pp. 1–12 (cit. on pp. 16, 17).

[CG77] Rina S Cohen and Arie Y Gold. "Theory of ω-languages I: Characterizations of ω-context-free languages". In: *Journal of Computer and SysteGGm Sciences* 15.2 (1977), pp. 169–184 (cit. on p. 1).

[Cho59] Noam Chomsky. "On certain formal properties of grammars". In: *Information and control* 2.2 (1959), pp. 137–167 (cit. on p. 7).

[CM] Yu-Fang Chen and Richard Mayr. *RABIT/Reduce: Tools for language inclusion testing and reduction of nondeterministic Büchi automata and NFA*. URL: http://www.languageinclusion.org/doku.php?id=tools (cit. on p. 20).

[CM17] Lorenzo Clemente and Richard Mayr. "Efficient reduction of nondeterministic automata with application to language inclusion testing". In: *arXiv preprint arXiv:1711.09946* (2017) (cit. on pp. 2, 13, 14, 17, 22, 55).

[CM18] Lorenzo Clemente and Richard Mayr. "Revisiting Call-by-Value Boehm Trees". In: 15.1 (2018), pp. 1–24. DOI: 10.2168/LMCS. arXiv: arXiv:1809.00134v1. URL: https://arxiv.org/pdf/1809.00134.pdf (cit. on p. 15).

[CNP93] Hugues Calbrix, Maurice Nivat, and Andreas Podelski. "Ultimately periodic words of rational ω-languages". In: *International Conference on Mathematical Foundations of Programming Semantics*. Springer. 1993, pp. 554–566 (cit. on p. 22).

[Cou+05] Patrick Cousot et al. "The ASTRÉE analyzer". In: *European Symposium on Programming*. Springer. 2005, pp. 21–30 (cit. on p. 2).

[CRT11] Silvia Crafa, Francesco Ranzato, and Francesco Tapparo. "Saving space in a time efficient simulation algorithm". In: *Fundamenta Informaticae* 108.1-2 (2011), pp. 23–42 (cit. on p. 16).

[DG20] Kyveli Doveri and Pierre Ganty. "Inclusion Checking Algorithms for ω-Languages". submitted to the 27th Static Analysis Symposium. 2020 (cit. on pp. 1, 2, 8, 22, 23, 25, 26, 45–47, 49, 55).

[DHWT91] David L Dill, Alan J Hu, and Howard Wong-Toi. "Checking for language inclusion using simulation preorders". In: *International Conference on Computer Aided Verification*. Springer. 1991, pp. 255–265 (cit. on pp. 2, 13).

[Dij18] Tom van Dijk. "Oink: An Implementation and Evaluation of Modern Parity Game Solvers". In: *Tools and Algorithms for the Construction and Analysis of Systems*. Ed. by Dirk Beyer and Marieke Huisman. Cham: Springer International Publishing, 2018, pp. 291–308. ISBN: 978-3-319-89960-2 (cit. on p. 20).

[DW+06] Martin De Wulf et al. "Antichains: A new algorithm for checking universality of finite automata". In: *International Conference on Computer Aided Verification*. Springer. 2006, pp. 17–30 (cit. on pp. 8, 55).

[Esp17] Javier Esparza. *Automata theory-An algorithmic approach*. Lecture notes. 2017. URL: https://www7.in.tum.de/~esparza/autoskript.pdf (cit. on pp. 1, 8).

[Ete02] Kousha Etessami. "A hierarchy of polynomial-time computable simulations for automata". In: *International Conference on Concurrency Theory*. Springer. 2002, pp. 131–144 (cit. on p. 15).

[EWS05] Kousha Etessami, Thomas Wilke, and Rebecca A Schuller. "Fair simulation relations, parity games, and state space reduction for Büchi automata". In: *SIAM Journal on Computing* 34.5 (2005), pp. 1159–1175 (cit. on pp. 13, 17, 18, 20).

[GR62] Seymour Ginsburg and H Gordon Rice. "Two families of languages related to ALGOL". In: *Journal of the ACM (JACM)* 9.3 (1962), pp. 350–371 (cit. on p. 29).

[GRV19] Pierre Ganty, Francesco Ranzato, and Pedro Valero. "Language Inclusion Algorithms as Complete Abstract Interpretations". In: *International Static Analysis Symposium*. Springer. 2019, pp. 140–161 (cit. on pp. 1, 8, 11, 20, 22, 29–31, 45, 46).

[HC14] Martin Hofmann and Wei Chen. "Abstract interpretation from Büchi automata". In: *Proceedings 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)*. 2014, pp. 1–10 (cit. on pp. 1, 8).

[HHK95] Monika Rauch Henzinger, Thomas A Henzinger, and Peter W Kopke. "Computing simulations on finite and infinite graphs". In: *Proceedings of IEEE 36th Annual Foundations of Computer Science*. IEEE. 1995, pp. 453–462 (cit. on pp. 13, 16).

[HK73] John E Hopcroft and Richard M Karp. "An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs". In: *SIAM Journal on computing* 2.4 (1973), pp. 225–231 (cit. on p. 55).

[HKR02] Thomas A Henzinger, Orna Kupferman, and Sriram K Rajamani. "Fair simulation". In: *Information and Computation* 173.1 (2002), pp. 64–81 (cit. on pp. 13, 17).

[HMU13] John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. *Introduction to Automata Theory, Languages, and Computation: Pearson New International Edition*. Pearson Higher Ed, 2013 (cit. on p. 4).

[Jur00] Marcin Jurdziński. "Small progress measures for solving parity games". In: *Annual Symposium on Theoretical Aspects of Computer Science*. Springer. 2000, pp. 290–301 (cit. on pp. 18, 19).

[Kla94] Nils Klarlund. "Progress measures, immediate determinacy, and a subset construction for tree automata". In: *Annals of Pure and Applied Logic* 69.2-3 (1994), pp. 243–268 (cit. on p. 18).

[Kle+52] Stephen Cole Kleene et al. *Introduction to metamathematics*. Vol. 483. from Nostrand New York, 1952 (cit. on p. 10).

[Kup18] Orna Kupferman. "Automata theory and model checking". In: *Handbook of Model Checking*. Springer, 2018, pp. 107–151 (cit. on pp. 1, 8).

[KV96] Orna Kupferman and Moshe Y Vardi. "Verification of fair transition systems". In: *International conference on computer aided verification*. Springer. 1996, pp. 372–382 (cit. on p. 8).

[Min17] Antoine Miné. "Tutorial on static inference of numeric invariants by abstract interpretation". In: *Foundations and Trends in Programming Languages* 4.3-4 (2017), pp. 120–372 (cit. on p. 2).

[MMN17] Roland Meyer, Sebastian Muskalla, and Elisabeth Neumann. "Liveness verification and synthesis: New algorithms for recursive programs". In: *arXiv preprint arXiv:1701.02947* (2017) (cit. on p. 8).

[OW04] Joël Ouaknine and James 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. IEEE. 2004, pp. 54–63 (cit. on p. 1).

[RT07] Francesco Ranzato and Francesco Tapparo. "A new efficient simulation equivalence algorithm". In: *22nd Annual IEEE Symposium on Logic in Computer Science (LICS 2007).* IEEE. 2007, pp. 171–180 (cit. on pp. 2, 16).

[RZ19] Francesco Ranzato and Marco Zanella. "Robustness verification of support vector machines". In: *International Static Analysis Symposium.* Springer. 2019, pp. 271–295 (cit. on p. 2).

[SB00] Fabio Somenzi and Roderick Bloem. "Efficient Büchi automata from LTL formulae". In: *International Conference on Computer Aided Verification.* Springer. 2000, pp. 248–263 (cit. on p. 13).

[Tar+55] Alfred Tarski et al. "A lattice-theoretical fixpoint theorem and its applications." In: *Pacific journal of Mathematics* 5.2 (1955), pp. 285–309 (cit. on p. 10).

[Wal00] Igor Walukiewicz. "Completeness of Kozen's axiomatisation of the propositional μ-calculus". In: *Information and Computation* 157.1-2 (2000), pp. 142–182 (cit. on p. 18).