Complete Abstractions for Checking Language Inclusion

PIERRE GANTY, IMDEA Software Institute, Spain
FRANCESCO RANZATO, Dipartimento di Matematica, University of Padova, Italy
PEDRO VALERO, IMDEA Software Institute, Spain

We study the language inclusion problem $L_1 \subseteq L_2$ where L_1 is regular or context-free. Our approach relies on abstract interpretation and checks whether an overapproximating abstraction of L_1, obtained by approximating the Kleene iterates of its least fixpoint characterization, is included in L_2. We show that a language inclusion problem is decidable whenever this overapproximating abstraction satisfies a completeness condition (i.e., its loss of precision causes no false alarm) and prevents infinite ascending chains (i.e., it guarantees termination of least fixpoint computations). This overapproximating abstraction of languages can be defined using quasiorder relations on words, where the abstraction gives the language of all the words "greater than or equal to" a given input word for that quasiorder. We put forward a range of such quasiorders that allow us to systematically design decision procedures for different language inclusion problems such as regular languages into regular languages or into trace sets of one-counter nets, and context-free languages into regular languages. In the case of inclusion between regular languages, some of the induced inclusion checking procedures correspond to well-known state-of-the-art algorithms like the so-called antichain algorithms. Finally, we provide an equivalent language inclusion checking algorithm based on a greatest fixpoint computation that relies on quotients of languages and, to the best of our knowledge, was not previously known.

CCS Concepts: • **Theory of computation → Regular languages**; **Grammars and context-free languages**; **Abstraction**; **Program reasoning**; • **Software and its engineering → Formal language definitions**.

Additional Key Words and Phrases: Abstract interpretation, completeness, language inclusion, regular language, context-free language, one-counter net, automaton, grammar.

ACM Reference Format:
Pierre Ganty, Francesco Ranzato, and Pedro Valero. 2021. Complete Abstractions for Checking Language Inclusion. *ACM Trans. Comput. Logic* 1, 1, Article 1 (January 2021), 40 pages. https://doi.org/10.1145/3462673

1 INTRODUCTION

Language inclusion is a fundamental and classical problem [Hopcroft and Ullman 1979, Chapter 11] which consists in deciding, given two languages L_1 and L_2, whether $L_1 \subseteq L_2$ holds. Language inclusion problems are found in diverse fields ranging from compiler construction [Bauer and Eickel 1976; Waite and Goos 1984] to model checking [Baier and Katoen 2008; Clarke et al. 2018]. We consider languages of finite words over a finite alphabet Σ. For regular and context-free languages, the inclusion problem is well known to be PSPACE-complete (see [Hunt et al. 1976]).

Authors' addresses: Pierre Ganty, pierre.ganty@imdea.org, IMDEA Software Institute, Madrid, Spain; Francesco Ranzato, francesco.ranzato@unipd.it, Dipartimento di Matematica, University of Padova, Padova, Italy; Pedro Valero, pedro.valero@imdea.org, IMDEA Software Institute, Madrid, Spain.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2021 Association for Computing Machinery.
1529-3785/2021/1-ART1 $15.00
https://doi.org/10.1145/3462673

The basic idea of our approach for solving a language inclusion problem $L_1 \subseteq L_2$ is to leverage Cousot and Cousot's abstract interpretation [Cousot and Cousot 1977, 1979] for checking the inclusion of an overapproximation (i.e., a superset) of L_1 into L_2. This idea draws inspiration from the work of Hofmann and Chen [2014], who used abstract interpretation to decide language inclusion between languages of infinite words.

Let us assume that L_1 is specified as least fixpoint of an equation system $X = F_{L_1}(X)$ on sets of words in $\wp(\Sigma^*)$, that is, $L_1 = \mathrm{lfp}(F_{L_1})$ is viewed as limit of the possibly infinite sequence of Kleene iterates $\{F^n_{L_1}(\varnothing)\}_{n\in\mathbb{N}}$ of the transformer F_{L_1}. An approximation of L_1 is obtained by applying an overapproximation for sets of words as modeled by a closure operator $\rho : \wp(\Sigma^*) \to \wp(\Sigma^*)$. In abstract interpretation one such closure ρ logically defines an *abstract domain*, which is here used for overapproximating a language by adding words to it, possibly none in case of no approximation. The language abstraction ρ is then used for defining an abstract check of convergence for the Kleene iterates of F_{L_1} whose limit is L_1, i.e., the convergence of the sequence $\{F^n_{L_1}(\varnothing)\}_{n\in\mathbb{N}}$ is checked on the abstraction ρ by the condition $\rho(F^{n+1}_{L_1}(\varnothing)) \subseteq \rho(F^n_{L_1}(\varnothing))$. If the abstraction ρ does not contain infinite ascending chains then we obtain finite convergence w.r.t. this abstract check for some $F^N_{L_1}(\varnothing)$.

This abstract interpretation-based approach finitely computes an abstraction $L_1^\rho = \rho(F^N_{L_1}(\varnothing))$ such that the abstract language inclusion check $L_1^\rho \subseteq L_2$ is *sound* because $L_1 \subseteq L_1^\rho$ always holds. We then give conditions on the closure ρ which ensure a *complete* abstract inclusion check, namely, the answer to $L_1^\rho \subseteq L_2$ is always exact (no "false alarm" in abstract interpretation terminology):

(i) L_2 is *exactly represented* by the abstraction ρ, i.e., $\rho(L_2) = L_2$;
(ii) ρ is a *complete abstraction* for symbol concatenation $\lambda X \in \wp(\Sigma^*).\, aX$, for all $a \in \Sigma$, according to the standard notion of completeness in abstract interpretation [Cousot and Cousot 1977]; this entails that $\rho(L_1) = L_1^\rho$ holds, so that $L_1^\rho \nsubseteq L_2$ implies $L_1 \nsubseteq L_2$.

This approach leads us to design a general algorithmic framework for language inclusion problems which is parameterized by an underlying language abstraction ρ.

We then focus on language abstractions which are induced by a quasiorder relation on words $\leqslant \subseteq \Sigma^* \times \Sigma^*$. Here, a language L is overapproximated by adding all the words which are "greater than or equal to" some word of L for $\leqslant$. This allows us to instantiate the above conditions (i) and (ii) for achieving a complete abstract inclusion check in terms of the quasiorder relation $\leqslant$. Termination, which corresponds to having finitely many Kleene iterates, is guaranteed by requiring that the relation $\leqslant$ is a *well-quasiorder*.

We define well-quasiorders satisfying the conditions (i) and (ii) which are directly derived from the standard Nerode equivalence relations on words. These quasiorders have been first investigated by Ehrenfeucht et al. [1983] and have been later generalized and extended by de Luca and Varricchio [1994; 2011]. In particular, drawing from a result by de Luca and Varricchio [1994], we show that the language abstractions induced by the Nerode quasiorders are the most general ones (intuitively, optimal) which fit in our algorithmic framework for checking language inclusion. While these quasiorder abstractions do not depend on some finite representation of languages (e.g., some class of automata), we provide quasiorders which instead exploit an underlying language representation given by a finite automaton. In particular, by selecting suitable well-quasiorders for the class of language inclusion problems at hand we are able to systematically derive decision procedures of the inclusion problem $L_1 \subseteq L_2$ for the following cases:

(1) both L_1 and L_2 are regular;
(2) L_1 is regular and L_2 is the trace language of a one-counter net;
(3) L_1 is context-free and L_2 is regular.

These decision procedures, here systematically designed by instantiating our framework, are then related to existing language inclusion checking algorithms. We study in detail the case where both languages L_1 and L_2 are regular and represented by finite state automata. When our decision procedure for $L_1 \subseteq L_2$ is derived from a well-quasiorder on Σ^* by exploiting an automaton-based representation of L_2, it turns out that we obtain the well-known "antichain algorithm" by De Wulf et al. [2006]. Moreover, by including a simulation relation in the definition of the well-quasiorder we derive a decision procedure that partially matches the language inclusion algorithm by Abdulla et al. [2010], and, in turn, also that by Bonchi and Pous [2013]. It is worth pointing out that for the case in which L_1 is regular and L_2 is the set of traces of a one-counter net, our systematic instantiation provides an alternative proof for the decidability of the corresponding language inclusion problem [Jančar et al. 1999].

Finally, we leverage a standard duality result between abstract least and greatest fixpoint checking [Cousot 2000] and put forward a *greatest* fixpoint approach (instead of the above *least* fixpoint-based procedures) for the case where both L_1 and L_2 are regular languages. Here, we exploit the properties of the overapproximating abstraction induced by the quasiorder relation in order to show that the Kleene iterates converging to the greatest fixpoint are finitely many. Interestingly, the Kleene iterates of the greatest fixpoint are finitely many whether you apply the overapproximating abstraction or not, and this is shown by relying on a second type of completeness in abstract interpretation called forward completeness [Giacobazzi and Quintarelli 2001].

Structure of the Article. In Section 2 we recall the needed basic notions and background on order theory, abstract interpretation and formal languages. Section 3 defines a general method for checking the convergence of Kleene iterates on an abstract domain, which provides the basis for designing in Section 4 an abstract interpretation-based framework for checking language inclusion, in particular by relying on abstractions that are complete for concatenation of languages. This general framework is instantiated in Section 5 to the class of abstractions induced by well-quasiorders on words, thus yielding effective inclusion checking algorithms for regular languages and traces of one-counter nets. Section 6 shows that one specific instance of our algorithmic framework turns out to be equivalent to the well-known antichain algorithm for language inclusion by De Wulf et al. [2006]. The instantiation of the framework for checking the inclusion of context-free languages into regular languages is described in Section 7. Section 8 shows how to derive a new language inclusion algorithm which relies on the computation of a greatest fixpoint rather than a least fixpoint. Finally, Section 9 outlines some directions for future work.

This article is an extended and revised version of the conference paper [Ganty et al. 2019], that includes full proofs, additional detailed examples, a simplification of some technical notions, and a new application for checking the inclusion of context-free languages into regular languages.

2 BACKGROUND

2.1 Order Theory

If X is any set then $\wp(X)$ denotes its powerset. 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. If $f : X \to Y$ is a function between sets and $S \in \wp(X)$ then $f(S) \triangleq \{f(x) \in Y \mid x \in S\}$ denotes its image on a subset S. A composition of two functions f and g is denoted both by fg and $f \circ g$.

$\langle D, \leqslant\rangle$ is a *quasiordered set* (qoset) when $\leqslant$ is a quasiorder (qo) relation on D, i.e. a reflexive and transitive binary relation $\leqslant \subseteq D \times D$. In a qoset $\langle D, \leqslant\rangle$ we will use the following induced equivalence relation $\sim_D$: for all $d, d' \in D$, $d \sim_D d' \overset{\triangle}{\Leftrightarrow} d \leqslant d' \wedge d' \leqslant d$. 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 $i \in \mathbb{N}$, $x_i \leqslant x_{i+1}$ (resp. $x_{i+1} \leqslant x_i$). A qoset is called ACC (DCC) when it satisfies the ACC (DCC).

A qoset $\langle D, \leqslant \rangle$ is a *partially ordered set* (poset) when $\leqslant$ is antisymmetric. A subset $X \subseteq D$ of a poset is *directed* if X is nonempty and every pair of elements in X has an upper bound in X. A poset $\langle D, \leqslant \rangle$ is a *directed-complete partial order* (CPO) if it has the least upper bound (lub) of all its directed subsets. A poset is a *join-semilattice* if it has the lub of all its nonempty finite subsets (therefore binary lubs are enough). A poset is a *complete lattice* if it has the lub of all its arbitrary (possibly empty) subsets; in this case, let us recall that it also has the greatest lower bound (glb) of all its arbitrary subsets.

An *antichain* in a qoset $\langle D, \leqslant \rangle$ is a subset $X \subseteq D$ such that any two distinct elements in X are incomparable for $\leqslant$. We denote the set of antichains of a qoset $\langle D, \leqslant \rangle$ by $\mathrm{AC}_{\langle D, \leqslant \rangle} \triangleq \{X \subseteq D \mid X \text{ is an antichain}\}$. A qoset $\langle D, \leqslant \rangle$ is a *well-quasiordered set* (wqoset), and $\leqslant$ is called well-quasiorder (wqo) on D, 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 $x_i \leqslant x_j$. Equivalently, $\langle D, \leqslant \rangle$ is a wqoset iff D is DCC and D has no infinite antichain. For every qoset $\langle D, \leqslant \rangle$, let us define the following binary relation $\sqsubseteq$ on the powerset $\wp(D)$: for all $X, Y \in \wp(D)$,

$$X \sqsubseteq Y \overset{\triangle}{\Leftrightarrow} \forall x \in X, \exists y \in Y,\ y \leqslant x \ . \tag{1}$$

Sometimes, we use the notation $\sqsubseteq_{\leqslant}$ to highlight the underlying qo $\leqslant$. A *minor* of a subset $X \subseteq D$, denoted by $\lfloor X \rfloor$, is a subset of the minimal elements of X w.r.t. $\leqslant$, i.e. $\lfloor X \rfloor \subseteq \min_{\leqslant}(X) \triangleq \{x \in X \mid \forall y \in X, y \leqslant x \Rightarrow y = x\}$, such that $X \sqsubseteq \lfloor X \rfloor$ holds. Therefore, a minor $\lfloor X \rfloor$ of $X \subseteq D$ is always an antichain in D. Let us recall that every subset X of a wqoset $\langle D, \leqslant \rangle$ has at least one minor set, all minor sets of X are finite, $\lfloor \{x\} \rfloor = \{x\}$, $\lfloor \varnothing \rfloor = \varnothing$, and if $\langle D, \leqslant \rangle$ is additionally a poset then there exists exactly one minor set of X. It turns out that $\langle \mathrm{AC}_{\langle D, \leqslant \rangle}, \sqsubseteq \rangle$ is a qoset, which is ACC if $\langle D, \leqslant \rangle$ is a wqoset and is a poset if $\langle D, \leqslant \rangle$ is a poset.

For the sake of clarity, we overload the notation and use the same symbol for a function/relation and its componentwise (i.e., pointwise) extension on product domains, e.g., if $f : X \to Y$ then f also denotes the standard product function $f : X^n \to Y^n$ which is componentwise defined by $\lambda \langle x_1, \ldots, x_n \rangle \in X^n . \langle f(x_1), \ldots, f(x_n) \rangle$. A vector $\vec{x}$ in some product domain $D^{|S|}$ indexed by a finite set S is also denoted by $\langle x_i \rangle_{i \in S}$ and, for some $i \in S$, $\vec{x}_i$ denotes its component x_i.

Let $\langle X, \leqslant \rangle$ be a qoset and $f : X \to X$ be a function. f is *monotonic* when $x \leqslant y$ implies $f(x) \leqslant f(y)$. For all $n \in \mathbb{N}$, the n-th power $f^n : X \to X$ of f is inductively defined by: $f^0 \triangleq \lambda x.x$; $f^{n+1} \triangleq f \circ f^n$ (or, equivalently, $f^{n+1} \triangleq f^n \circ f$). The denumerable sequence of *Kleene iterates* of f starting from an initial value $a \in X$ is given by $\langle f^n(a) \rangle_{n \in \mathbb{N}}$. If $\langle X, \leqslant \rangle$ is a poset and $a \in X$ then $\mathrm{lfp}_a(f)$ (resp. $\mathrm{gfp}_a(f)$) denotes the least (resp. greatest) fixpoint of f which is greater (resp. less) than or equal to a, when this exists; in particular, $\mathrm{lfp}(f)$ (resp. $\mathrm{gfp}(f)$) denotes the least (resp. greatest) fixpoint of f, when this exists. If $\langle X, \leqslant \rangle$ is an ACC (resp. DCC) CPO, $a \leqslant f(a)$ (resp. $f(a) \leqslant a$) holds and f is monotonic then the Kleene iterates $\langle f^n(a) \rangle_{n \in \mathbb{N}}$ *finitely converge* to $\mathrm{lfp}_a(f)$ (resp. $\mathrm{gfp}_a(f)$), i.e., there exists $k \in \mathbb{N}$ such that for all $n \geq k$, $f^n(a) = f^k(a) = \mathrm{lfp}_a(f)$ (resp. $\mathrm{gfp}_a(f)$). In particular, if $\bot$ (resp. $\top$) is the least (greatest) element of $\langle X, \leqslant \rangle$ then $\langle f^n(\bot) \rangle_{n \in \mathbb{N}}$ (resp. $\langle f^n(\top) \rangle_{n \in \mathbb{N}}$) finitely converges to $\mathrm{lfp}(f)$ (resp. $\mathrm{gfp}(f)$).

2.2 Abstract Interpretation

Let us recall some basic notions on closure operators and Galois Connections commonly used in abstract interpretation (see, e.g., [Cousot and Cousot 1979; Miné 2017; Rival and Yi 2020]). Closure operators and Galois Connections are equivalent notions and, therefore, they are both used for defining the notion of approximation in abstract interpretation. Closure operators allow us to define and reason on abstract domains independently of a specific representation for abstract values which is instead required by Galois Connections.

Let $\langle C, \leq_C, \vee, \wedge\rangle$ be a complete lattice, where $\vee$ and $\wedge$ denote, resp., lub and glb. An *upper closure operator*, or simply *closure*, on $\langle C, \leq_C\rangle$ is a function $\rho : C \to C$ which is: (i) monotonic, (ii) idempotent: $\rho(\rho(x)) = \rho(x)$ for all $x \in C$, and (iii) extensive: $x \leq_C \rho(x)$ for all $x \in C$. The set of all upper closed operators on C is denoted by $\mathrm{uco}(C)$. We often write $c \in \rho(C)$, or simply $c \in \rho$, to denote that there exists $c' \in C$ such that $c = \rho(c')$, and we recall that this happens iff $\rho(c) = c$. If $\rho \in \mathrm{uco}(C)$ then for all $c_1 \in C$, $c_2 \in \rho$ and $X \subseteq C$, it turns out that:

$$c_1 \leq_C c_2 \Leftrightarrow \rho(c_1) \leq_C \rho(c_2) \Leftrightarrow \rho(c_1) \leq_C c_2 \tag{2}$$

$$\rho(\vee X) = \rho(\vee \rho(X)) \quad \text{and} \quad \wedge\rho(X) = \rho(\wedge\rho(X)) \ . \tag{3}$$

In abstract interpretation, a closure operator $\rho \in \mathrm{uco}(C)$ on a concrete domain C plays the role of abstraction function for objects of C. Given two closures $\rho, \rho' \in \mathrm{uco}(C)$, ρ is a *coarser* abstraction than ρ' (or, equivalently, ρ' is a more precise abstraction than ρ) iff the image of ρ is a subset of the image of ρ', i.e. $\rho(C) \subseteq \rho'(C)$, and this happens iff for any $x \in C$, $\rho'(x) \leq_C \rho(x)$.

Let us recall that a *Galois Connection* (GC) or *adjunction* between two posets $\langle C, \leq_C\rangle$, called concrete domain, and $\langle A, \leq_A\rangle$, called abstract domain, consists of two functions $\alpha\colon C \to A$ and $\gamma\colon A \to C$ such that $\alpha(c) \leq_A a \;\Leftrightarrow\; c \leq_C \gamma(a)$ always holds. A Galois Connection is denoted by $\langle C, \leq_C\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \leq_A\rangle$. The function α is called the left-adjoint of γ, and, dually, γ is called the right-adjoint of α. This terminology is justified by the fact that if some function $\alpha : C \to A$ admits a right-adjoint $\gamma : A \to C$ then this is unique, and this dually holds for left-adjoints. It turns out that in a GC between complete lattices, γ is always co-additive (i.e., γ preserves arbitrary glb's) while α is always additive (i.e., α preserves arbitrary lub's). Moreover, an additive function $\alpha : C \to A$ uniquely determines its right-adjoint by $\gamma \triangleq \lambda a. \vee_C\{c \in C \mid \alpha(c) \leq_A a\}$ and, dually, a co-additive function $\gamma : A \to C$ uniquely determines its left-adjoint by $\alpha \triangleq \lambda c. \wedge_A\{a \in A \mid c \leq_C \gamma(a)\}$.

The following remark is folklore in abstract interpretation and a proof is here provided for the sake of completeness.

Lemma 2.1. *Let $\langle C, \leq_C\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \leq_A\rangle$ be a GC between complete lattices and $f\colon C \to C$ be a monotonic function. Then, $\gamma(\mathrm{lfp}(\alpha f \gamma)) = \mathrm{lfp}(\gamma\alpha f)$.*

Proof. Let us first show that $\mathrm{lfp}(\gamma\alpha f) \leq_C \gamma(\mathrm{lfp}(\alpha f\gamma))$:

$$\begin{aligned}
\gamma(\mathrm{lfp}(\alpha f\gamma)) \leq_C \gamma(\mathrm{lfp}(\alpha f\gamma)) \Leftrightarrow \quad & [\text{Since } g(\mathrm{lfp}(g)) = \mathrm{lfp}(g)]\\
\gamma\alpha f(\gamma(\mathrm{lfp}(\alpha f\gamma))) \leq_C \gamma(\mathrm{lfp}(\alpha f\gamma)) \Rightarrow \quad & [\text{Since } g(x) \leq x \Rightarrow \mathrm{lfp}(g) \leq x]\\
\mathrm{lfp}(\gamma\alpha f) \leq_C \gamma(\mathrm{lfp}(\alpha f\gamma)) \quad &
\end{aligned}$$

Then, let us prove that $\gamma(\mathrm{lfp}(\alpha f\gamma)) \leq_C \mathrm{lfp}(\gamma\alpha f)$:

$$\begin{aligned}
\mathrm{lfp}(\gamma\alpha f) \leq_C \mathrm{lfp}(\gamma\alpha f) \Leftrightarrow \quad & [\text{Since } g(\mathrm{lfp}(g)) = \mathrm{lfp}(g)]\\
\gamma\alpha f(\mathrm{lfp}(\gamma\alpha f)) \leq_C \mathrm{lfp}(\gamma\alpha f) \Rightarrow \quad & [\text{Since } \alpha \text{ is monotone}]\\
\alpha\gamma\alpha f(\mathrm{lfp}(\gamma\alpha f)) \leq_A \alpha(\mathrm{lfp}(\gamma\alpha f)) \Leftrightarrow \quad & [\text{Since } \alpha\gamma\alpha = \alpha \text{ in GCs}]\\
\alpha f(\mathrm{lfp}(\gamma\alpha f)) \leq_A \alpha(\mathrm{lfp}(\gamma\alpha f)) \Leftrightarrow \quad & [\text{Since } \gamma\alpha(\mathrm{lfp}(\gamma\alpha f)) = \mathrm{lfp}(\gamma\alpha f)]\\
\alpha f\gamma(\alpha(\mathrm{lfp}(\gamma\alpha f))) \leq_A \alpha(\mathrm{lfp}(\gamma\alpha f)) \Rightarrow \quad & [\text{Since } g(x) \leq x \Rightarrow \mathrm{lfp}(g) \leq x]\\
\mathrm{lfp}(\alpha f\gamma) \leq_A \alpha(\mathrm{lfp}(\gamma\alpha f)) \Rightarrow \quad & [\text{Since } \gamma \text{ is monotone}]\\
\gamma(\mathrm{lfp}(\alpha f\gamma)) \leq_C \gamma\alpha(\mathrm{lfp}(\gamma\alpha f)) \Leftrightarrow \quad & [\text{Since } \gamma\alpha(\mathrm{lfp}(\gamma\alpha f)) = \mathrm{lfp}(\gamma\alpha f)]\\
\gamma(\mathrm{lfp}(\alpha f\gamma)) \leq_C \mathrm{lfp}(\gamma\alpha f) \quad &
\end{aligned}$$

□

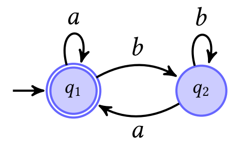


Fig. 1. A finite automaton $\mathcal{A}$ with $\mathcal{L}(\mathcal{A}) = (b^*a)^*$.

2.3 Languages

Let Σ be an alphabet, i.e., a finite nonempty set of symbols. A word (or string) on Σ is a finite (possibly empty) sequence of symbols in Σ, where ϵ denotes the empty sequence. Σ^* denotes the set of finite words on Σ. A language on Σ is a subset $L \subseteq \Sigma^*$. Concatenation of words and languages is denoted by simple juxtaposition, that is, the concatenation of words $u, v \in \Sigma^*$ is denoted by $uv \in \Sigma^*$, while the concatenation of languages $L, L' \subseteq \Sigma^*$ is denoted by $LL' \triangleq \{uv \mid u \in L,\ v \in L'\}$. By considering a word as a singleton language, we also concatenate words with languages, for example uL and uLv.

A *finite automaton* (FA) is a tuple $\mathcal{A} = \langle Q, \delta, I, F, \Sigma\rangle$ where: Σ is an alphabet, Q is a finite set of states, $I \subseteq Q$ is a subset of initial states, $F \subseteq Q$ is a subset of final states, and $\delta\colon Q \times \Sigma \to \wp(Q)$ is a transition relation. The notation $q \xrightarrow{a} q'$ is also used to denote that $q' \in \delta(q, a)$. If $u \in \Sigma^*$ and $q, q' \in Q$ then $q \stackrel{u}{\rightsquigarrow} q'$ means that the state q' is reachable from q by following the string u. More formally, by induction on the length of $u \in \Sigma^*$: (i) if $u = \epsilon$ then $q \stackrel{\epsilon}{\rightsquigarrow} q'$ iff $q = q'$; (ii) if $u = av$ with $a \in \Sigma, v \in \Sigma^*$ then $q \stackrel{av}{\rightsquigarrow} q'$ iff $\exists q'' \in \delta(q, a),\ q'' \stackrel{v}{\rightsquigarrow} q'$. The *language generated* by a FA $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) \triangleq \{u \in \Sigma^* \mid \exists q_i \in I, \exists q_f \in F,\ q_i \stackrel{u}{\rightsquigarrow} q_f\}$. An example of FA is depicted in Fig. 1.

3 KLEENE ITERATES WITH ABSTRACT INCLUSION CHECK

Abstract interpretation can be applied to solve a generic inclusion checking problem by leveraging backward complete abstractions [Cousot and Cousot 1977, 1979; Giacobazzi et al. 2000; Ranzato 2013]. A closure $\rho \in \mathrm{uco}(C)$ is called *backward complete* for a concrete monotonic function $f : C \to C$ when $\rho f = \rho f \rho$ holds. Since $\rho f(c) \leq_C \rho f \rho(c)$ always holds for all $c \in C$ (because ρ is extensive and monotonic and f is monotonic), the intuition is that backward completeness models an ideal situation where no loss of precision is accumulated in the computations of ρf when its concrete input objects c are approximated by $\rho(c)$. It is well known [Cousot and Cousot 1979] that backward completeness implies completeness of least fixpoints, namely for all $x \in C$ such that $x \leq_C f(x)$,

$$\rho f = \rho f \rho \ \Rightarrow\ \rho(\mathrm{lfp}_x(f)) = \mathrm{lfp}_x(\rho f) = \mathrm{lfp}_x(\rho f \rho) \tag{4}$$

provided that these least fixpoints exist (this is the case, e.g., when C is a CPO).

Given an initial value $a \in C$, let us define the following iterative procedure:

$$\textsc{Kleene}(\mathrm{Conv}, f, a) \triangleq \begin{cases} x := a; \\ \textbf{while}\ \neg\, \mathrm{Conv}(f(x), x)\ \textbf{do}\ x := f(x); \\ \textbf{return}\ x; \end{cases}$$

which computes the Kleene iterates of f starting from a and stops when a convergence relation $\mathrm{Conv} \subseteq C \times C$ for two consecutive Kleene iterates $f^{n+1}(a)$ and $f^n(a)$ holds. When $\mathrm{Conv} = \mathrm{Incl} \triangleq \{(x, y) \mid x \leq_C y\}$ is the convergence relation and $a \leq_C f(a)$ holds, the procedure $\textsc{Kleene}(\mathrm{Incl}, f, a)$ returns $\mathrm{lfp}_a(f)$ if the Kleene iterates $\langle f^n(a)\rangle_{n\in\mathbb{N}}$ finitely converge. Hence, termination of $\textsc{Kleene}(\mathrm{Incl}, f, a)$ is guaranteed when C is an ACC CPO.

Given a closure $\rho \in \mathrm{uco}(C)$, let us consider the following abstract convergence relation induced by ρ:

$$\mathrm{Incl}_\rho \triangleq \{(x, y) \in C \times C \mid \rho(x) \leq_C \rho(y)\} \ .$$

Hence, $\textsc{Kleene}(\mathrm{Incl}_\rho, f, a)$ terminates if eventually $\rho(f(x)) \leq_C \rho(x)$ holds. Notice that $\mathrm{Incl} \subseteq \mathrm{Incl}_\rho$ always holds by monotonicity of ρ and $\mathrm{Incl} = \mathrm{Incl}_\rho$ iff $\rho = \mathrm{id}$.

Theorem 3.1. *Let $\rho \in \mathrm{uco}(C)$ be such that ρ is backward complete for f and $\rho(C)$ does not contain infinite ascending chains. Let $a \in C$ be such that $a \leq_C f(a)$ holds. Then, the procedure $\textsc{Kleene}(\mathrm{Incl}_\rho, f, a)$ terminates and $\rho(\textsc{Kleene}(\mathrm{Incl}_\rho, f, a)) = \rho(\mathrm{lfp}_a(f)) = \mathrm{lfp}_a(\rho f)$ holds.*

Proof. Let us first prove by induction the following property:

$$\forall n \in \mathbb{N},\ \rho \circ f^n = (\rho \circ f)^n \circ \rho \ . \tag{5}$$

For $n = 0$, we have that $\rho \circ f^0 = \rho = (\rho \circ f)^0 \circ \rho$. For $n + 1$,

$$\begin{aligned}
\rho \circ f^{n+1} = &\qquad \text{[by definition of } f^{n+1}\text{]}\\
\rho \circ f^n \circ f = &\qquad \text{[by inductive hypothesis]}\\
(\rho \circ f)^n \circ \rho \circ f = &\qquad \text{[by backward completeness]}\\
(\rho \circ f)^n \circ \rho \circ f \circ \rho = &\qquad \text{[by definition of } (\rho \circ f)^{n+1}\text{]}\\
(\rho \circ f)^{n+1} \circ \rho \ . &
\end{aligned}$$

Then, let us observe that $\mathrm{lfp}_a(\rho f) = \mathrm{lfp}_{\rho(a)}(\rho f)$: this is a consequence of the fact that $\rho(f(x)) = x \wedge a \leq_C x$ iff $\rho(f(x)) = x \wedge \rho(a) \leq_C x$, because $\rho(f(x)) = x \wedge a \leq_C x$ implies $\rho(f(x)) = x \wedge \rho(a) \leq_C \rho(x) = \rho(\rho(f(x))) = \rho(f(x)) = x$.
Since $a \leq_C f(a)$, we have that the sequence $\langle f^n(a)\rangle_{n\in\mathbb{N}}$ is an ascending chain, so that, by monotonicity of ρ, $\langle \rho(f^n(a))\rangle_{n\in\mathbb{N}}$ is an ascending chain in $\rho(C)$. Since $\rho(C)$ does not contain infinite ascending chains, there exists $N = \min(\{n \in \mathbb{N} \mid \rho(f^{n+1}(a)) \leq_C \rho(f^n(a))\})$. This means that $\textsc{Kleene}(\mathrm{Incl}_\rho, f, a)$ terminates after $N+1$ iterations and outputs $f^N(a)$. We prove by induction on $N \in \mathbb{N}$ that $N = \min(\{n \in \mathbb{N} \mid (\rho \circ f)^{n+1}(\rho(a)) \leq_C (\rho \circ f)^n(\rho(a))\})$.

$(N = 0)$: We have that $\rho(f^1(a)) \leq_C \rho(f^0(a))$ holds, namely, $\rho(f(a)) \leq_C \rho(a)$. Then, by backward completeness, $\rho(f(\rho(a))) \leq_C \rho(a)$, namely, $(\rho \circ f)^1(\rho(a)) \leq_C (\rho \circ f)^0(\rho(a))$.

$(N + 1)$: We have that $\rho(f^{N+2}(a)) \leq_C \rho(f^{N+1}(a))$ holds, so that by (5), $(\rho \circ f)^{N+2}(\rho(a)) \leq_C (\rho \circ f)^{N+1}(\rho(a))$. Moreover, $N + 1$ is the minimum $n \in \mathbb{N}$ such that $(\rho \circ f)^{n+1}(\rho(a)) \leq_C (\rho \circ f)^n(\rho(a))$ holds, because if $(\rho \circ f)^{k+1}(\rho(a)) \leq_C (\rho \circ f)^k(\rho(a))$ holds for some $k \leq N$, then, by (5), we would have that $\rho(f^{k+1}(a)) \leq_C \rho(f^k(a))$, thus contradicting the minimality of $N + 1$ for $\rho(f^{n+1}(a)) \leq_C \rho(f^n(a))$.

Since $a \leq_C f(a)$ implies, by backward completeness, $\rho(a) \leq_C \rho(f(a)) = (\rho \circ f)(\rho(a))$, and $N = \min(\{n \in \mathbb{N} \mid (\rho \circ f)^{n+1}(\rho(a)) \leq_C (\rho \circ f)^n(\rho(a))\})$, it turns out that $(\rho \circ f)^N(\rho(a)) = \mathrm{lfp}_{\rho(a)}(\rho f) = \mathrm{lfp}_a(\rho f)$. Thus, by (5), we obtain $\mathrm{lfp}_a(\rho f) = (\rho \circ f)^N(\rho(a)) = \rho(f^N(a)) = \rho(\textsc{Kleene}(\mathrm{Incl}_\rho, f, a))$. Finally, by (4), $\rho(\textsc{Kleene}(\mathrm{Incl}_\rho, f, a)) = \mathrm{lfp}_a(\rho f) = \rho(\mathrm{lfp}_a(f))$. $\square$

We will apply the order-theoretic algorithmic scheme provided by Kleene under the hypotheses of Theorem 3.1 to a number of different language inclusion problems $L_1 \subseteq L_2$, where the language L_1 can be expressed as least fixpoint of a monotonic function on $\wp(\Sigma^*)$. This approach will allow us to systematically design several language inclusion algorithms which rely on different backward complete abstractions ρ of the complete lattice $\langle \wp(\Sigma^*), \subseteq\rangle$.

4 AN ALGORITHMIC FRAMEWORK FOR LANGUAGE INCLUSION

4.1 Languages as Fixed Points

Let $\mathcal{A} = \langle Q, \delta, I, F, \Sigma\rangle$ be a FA. Given $S, T \subseteq Q$, define the set of words leading from some state in S to some state in T as follows:

$$W^{\mathcal{A}}_{S,T} \triangleq \{u \in \Sigma^* \mid \exists q \in S,\ \exists q' \in T, q \overset{u}{\rightsquigarrow} q'\} \ .$$

When $S = \{q\}$ or $T = \{q'\}$ we slightly abuse the notation and write $W^{\mathcal{A}}_{q,T}$, $W^{\mathcal{A}}_{S,q'}$, or $W^{\mathcal{A}}_{q,q'}$. Also, we omit the automaton $\mathcal{A}$ in superscripts when this is clear from the context. The language accepted by $\mathcal{A}$ is therefore $\mathcal{L}(\mathcal{A}) \triangleq W^{\mathcal{A}}_{I,F}$. Observe that

$$\mathcal{L}(\mathcal{A}) = \textstyle\bigcup_{q\in I} W^{\mathcal{A}}_{q,F} = \bigcup_{q\in F} W^{\mathcal{A}}_{I,q} \tag{6}$$

where, as usual, $\bigcup \varnothing = \varnothing$.

Let us recall how to define the language accepted by an automaton as a solution of a set of equations [Schützenberger 1963]. Given a generic Boolean predicate $p(x)$ for a variable x ranging in some set (typically a membership predicate $x \in^? Z$) and two generic sets T and F, we define the following parametric choice function:

$$\psi^T_F(p(x)) \triangleq \begin{cases} T & \text{if } p(x) \text{ holds} \\ F & \text{otherwise} \end{cases} \ .$$

The FA $\mathcal{A}$ induces the following set of equations, where the X_q's are variables of type $X_q \in \wp(\Sigma^*)$ and are indexed by states $q \in Q$ of $\mathcal{A}$:

$$\mathrm{Eqn}(\mathcal{A}) \triangleq \{X_q = \psi^{\{\epsilon\}}_{\varnothing}(q \in^? F) \cup \textstyle\bigcup_{a\in\Sigma,\, q'\in\delta(q,a)} aX_{q'} \mid q \in Q\} \ . \tag{7}$$

Thus, the functions $\lambda\langle X_{q'}\rangle_{q'\in Q}.\, \psi^{\{\epsilon\}}_{\varnothing}(q \in^? F) \cup \bigcup_{a\in\Sigma,\, q'\in\delta(q,a)} aX_{q'}$ in the right-hand side of the equations in $\mathrm{Eqn}(\mathcal{A})$ have type $\wp(\Sigma^*)^{|Q|} \longrightarrow \wp(\Sigma^*)$. Since $\langle \wp(\Sigma^*)^{|Q|}, \subseteq\rangle$ is a (product) complete lattice (because $\langle \wp(\Sigma^*), \subseteq\rangle$ is a complete lattice) and all the right-hand side functions in $\mathrm{Eqn}(\mathcal{A})$ are clearly monotonic, the least solution $\langle Y_q\rangle_{q\in Q} \in \wp(\Sigma^*)^{|Q|}$ of $\mathrm{Eqn}(\mathcal{A})$ does exist and it is easy to check that for every $q \in Q$, $Y_q = W^{\mathcal{A}}_{q,F}$ holds.

It is worth noticing that, by relying on right concatenations rather than left ones $aX_{q'}$ used in $\mathrm{Eqn}(\mathcal{A})$, one could also define a set of symmetric equations whose least solution coincides with $\langle W^{\mathcal{A}}_{I,q}\rangle_{q\in Q}$ instead of $\langle W^{\mathcal{A}}_{q,F}\rangle_{q\in Q}$.

Example 4.1. Let us consider the automaton $\mathcal{A}$ in Figure 1. The set of equations induced by $\mathcal{A}$ are as follows:

$$\mathrm{Eqn}(\mathcal{A}) = \begin{cases} X_1 = \{\epsilon\} \cup aX_1 \cup bX_2 \\ X_2 = \varnothing \cup aX_1 \cup bX_2 \end{cases} \ . \qquad \diamond$$

It is notationally convenient to formulate the equations in $\mathrm{Eqn}(\mathcal{A})$ by exploiting an "initial" vector $\vec{\boldsymbol{\epsilon}}^F \in \wp(\Sigma^*)^{|Q|}$ and a predecessor function $\mathrm{Pre}_{\mathcal{A}} \colon \wp(\Sigma^*)^{|Q|} \to \wp(\Sigma^*)^{|Q|}$ defined as follows:

$$\vec{\boldsymbol{\epsilon}}^F \triangleq \langle \psi^{\{\epsilon\}}_{\varnothing}(q \in^? F)\rangle_{q\in Q}\,, \qquad \mathrm{Pre}_{\mathcal{A}}(\langle X_{q'}\rangle_{q'\in Q}) \triangleq \langle \textstyle\bigcup_{a\in\Sigma,\, q'\in\delta(q,a)} aX_{q'}\rangle_{q\in Q} \ .$$

The intuition for the function $\mathrm{Pre}_{\mathcal{A}}$ is that given the language $W^{\mathcal{A}}_{q',F}$ and a transition $q' \in \delta(q, a)$, we have that $aW^{\mathcal{A}}_{q',F} \subseteq W^{\mathcal{A}}_{q,F}$ holds, i.e., given a subset X'_q of the language generated by $\mathcal{A}$ from some state q', the function $\mathrm{Pre}_{\mathcal{A}}$ computes a subset X_q of the language generated by $\mathcal{A}$ for its predecessor state q. Notice that if all the components of a vector $\vec{X} \in \wp(\Sigma^*)^{|Q|}$ are finite sets of words then $\mathrm{Pre}_{\mathcal{A}}(\vec{X})$ is still a vector of finite sets. Since $\epsilon \in W^{\mathcal{A}}_{q,F}$ for all $q \in F$, the least fixpoint computation can start from the vector $\vec{\epsilon}^F$ and iteratively apply $\mathrm{Pre}_{\mathcal{A}}$. Therefore, it turns out that

$$\langle W^{\mathcal{A}}_{q,F}\rangle_{q\in Q} = \mathrm{lfp}(\lambda\vec{X}.\ \vec{\epsilon}^F \cup \mathrm{Pre}_{\mathcal{A}}(\vec{X}))\ . \tag{8}$$

Together with Equation (6), it follows that $\mathcal{L}(\mathcal{A})$ is given by the union of the component languages of the vector $\mathrm{lfp}(\lambda\vec{X}.\ \vec{\epsilon}^F \cup \mathrm{Pre}_{\mathcal{A}}(\vec{X}))$ that are indexed by the initial states in I.

Example 4.2 (Continuation of Example 4.1). The fixpoint characterization of $\langle W^{\mathcal{A}}_{q,F}\rangle_{q\in Q}$ is:

$$\begin{pmatrix} W^{\mathcal{A}}_{q_1,q_1} \\ W^{\mathcal{A}}_{q_2,q_1} \end{pmatrix} = \mathrm{lfp}\left(\lambda \begin{pmatrix} X_1 \\ X_2 \end{pmatrix}. \begin{pmatrix} \{\epsilon\} \cup aX_1 \cup bX_2 \\ \varnothing \cup aX_1 \cup bX_2 \end{pmatrix}\right) = \begin{pmatrix} (a + (b^+a))^* \\ (a+b)^*a \end{pmatrix}\ . \qquad \diamond$$

4.2 Language Inclusion using Fixed Points

Consider a language inclusion problem $L_1 \subseteq L_2$, where $L_1 = \mathcal{L}(\mathcal{A})$ for some FA $\mathcal{A} = \langle Q, \delta, I, F, \Sigma\rangle$. The language L_2 can be formalized as a vector in $\wp(\Sigma^*)^{|Q|}$ as follows:

$$\vec{L_2^I} \triangleq \langle \psi^{L_2}_{\Sigma^*}(q \in^? I)\rangle_{q\in Q} \tag{9}$$

whose components indexed by initial states in I are L_2 and those indexed by noninitial states are Σ^*. Then, as a consequence of (6), (8) and (9), we have that

$$\mathcal{L}(\mathcal{A}) \subseteq L_2 \Leftrightarrow \mathrm{lfp}(\lambda\vec{X}.\ \vec{\epsilon}^F \cup \mathrm{Pre}_{\mathcal{A}}(\vec{X})) \subseteq \vec{L_2^I}\ . \tag{10}$$

Theorem 4.3. *If $\rho \in \mathrm{uco}(\wp(\Sigma^*))$ is backward complete for $\lambda X \in \wp(\Sigma^*).\ aX$ for all $a \in \Sigma$, then, for all FAs $\mathcal{A} = \langle Q, \delta, I, F, \Sigma\rangle$ on the alphabet Σ, ρ is backward complete for $\mathrm{Pre}_{\mathcal{A}}$ and $\lambda\vec{X}.\ \vec{\epsilon}^F \cup \mathrm{Pre}_{\mathcal{A}}(\vec{X})$.*

Proof. First, it turns out that:

$$\begin{aligned}
\rho(\mathrm{Pre}_{\mathcal{A}}(\langle X_q\rangle_{q\in Q})) = &\quad \text{[by definition]}\\
\rho(\textstyle\bigcup_{a\in\Sigma,\, q'\in\delta(q,a)} aX_{q'}) = &\quad \text{[by (3)]}\\
\rho(\textstyle\bigcup_{a\in\Sigma,\, q'\in\delta(q,a)} \rho(aX_{q'})) = &\quad \text{[by backward completeness of } \rho \text{ for } \lambda X.\, aX]\\
\rho(\textstyle\bigcup_{a\in\Sigma,\, q'\in\delta(q,a)} \rho(a\rho(X_{q'}))) = &\quad \text{[by (3)]}\\
\rho(\textstyle\bigcup_{a\in\Sigma,\, q'\in\delta(q,a)} a\rho(X_{q'})) = &\quad \text{[by definition]}\\
\rho(\mathrm{Pre}_{\mathcal{A}}(\rho(\langle X_q\rangle_{q\in Q})))\ . &
\end{aligned}$$

As a consequence, ρ is backward complete for $\lambda\vec{X}.\ \vec{\epsilon}^F \cup \mathrm{Pre}_{\mathcal{A}}(\vec{X})$:

$$\begin{aligned}
\rho(\vec{\epsilon}^F \cup \mathrm{Pre}_{\mathcal{A}}(\rho(\vec{X}))) = &\quad \text{[by (3)]}\\
\rho(\rho(\vec{\epsilon}^F) \cup \rho(\mathrm{Pre}_{\mathcal{A}}(\rho(\vec{X})))) = &\quad \text{[by backward completeness of } \rho \text{ for } \mathrm{Pre}_{\mathcal{A}}]\\
\rho(\rho(\vec{\epsilon}^F) \cup \rho(\mathrm{Pre}_{\mathcal{A}}(\vec{X}))) = &\quad \text{[by (3)]}\\
\rho(\vec{\epsilon}^F \cup\ \mathrm{Pre}_{\mathcal{A}}(\vec{X}))\ . &
\end{aligned}$$
□

Then, by resorting to the least fixpoint transfer of completeness (4), we also obtain the following consequence.

Corollary 4.4. *If $\rho \in \mathrm{uco}(\wp(\Sigma^*))$ is backward complete for $\lambda X \in \wp(\Sigma^*).\ aX$ for all $a \in \Sigma$, then $\rho(\mathrm{lfp}(\lambda\vec{X}.\ \vec{\epsilon}^F \cup \mathrm{Pre}_{\mathcal{A}}(\vec{X}))) = \mathrm{lfp}(\lambda\vec{X}.\ \rho(\vec{\epsilon}^F \cup \mathrm{Pre}_{\mathcal{A}}(\vec{X})))$.*

Note that if ρ is backward complete for $\lambda X.aX$, for all $a \in \Sigma$, and $L_2 \in \rho$ then, by Theorem 3.1 and Corollary 4.4, the equivalence (10) becomes

$$\mathcal{L}(\mathcal{A}) \subseteq L_2 \Leftrightarrow \mathrm{lfp}(\lambda\vec{X}.\rho(\vec{\epsilon}^F \cup \mathrm{Pre}_{\mathcal{A}}(\vec{X}))) \subseteq \vec{L_2^I}\ . \tag{11}$$

4.2.1 Right Concatenation. Let us consider the symmetric case of right concatenation $\lambda X.Xa$. Recall that $W_{I,q} = \{w \in \Sigma^* \mid \exists q_i \in I, q_i \overset{w}{\rightsquigarrow} q\}$ and that $W_{I,q} = \psi_{\varnothing}^{\{\epsilon\}}(q \in^? I) \cup \bigcup_{a\in\Sigma, a\in W_{q',q}} W_{I,q'}a$ holds. Correspondingly, we define a set of fixpoint equations on $\wp(\Sigma^*)$ which is based on right concatenation and is symmetric to the equations defined in (7):

$$\mathrm{Eqn}^{\mathrm{r}}(\mathcal{A}) \triangleq \{X_q = \psi_{\varnothing}^{\{\epsilon\}}(q \in^? I) \cup \textstyle\bigcup_{a\in\Sigma,\, q\in\delta(q',a)} X_{q'}a \mid q \in Q\} \ .$$

In this case, if $\vec{Y} = \langle Y_q\rangle_{q\in Q}$ is the least fixpoint solution of $\mathrm{Eqn}^{\mathrm{r}}(\mathcal{A})$ then $Y_q = W_{I,q}^{\mathcal{A}}$ for every $q \in Q$. Also, by defining $\vec{\boldsymbol{\epsilon}}^I \in \wp(\Sigma^*)^{|Q|}$ and $\mathrm{Post}_{\mathcal{A}} \colon \wp(\Sigma^*)^{|Q|} \to \wp(\Sigma^*)^{|Q|}$ as follows:

$$\vec{\boldsymbol{\epsilon}}^I \triangleq \langle \psi_{\varnothing}^{\{\epsilon\}}(q \in^? I)\rangle_{q\in Q} \qquad\qquad \mathrm{Post}_{\mathcal{A}}(\langle X_q\rangle_{q\in Q}) \triangleq \langle \textstyle\bigcup_{a\in\Sigma,\, q\in\delta(q',a)} X_{q'}a\rangle_{q\in Q}$$

we have that

$$\langle W_{I,q}\rangle_{q\in Q} = \mathrm{lfp}(\lambda\vec{\boldsymbol{X}}.\ \vec{\boldsymbol{\epsilon}}^I \cup \mathrm{Post}_{\mathcal{A}}(\vec{\boldsymbol{X}})) \ . \tag{12}$$

Thus, by (6), it turns out that $\mathcal{L}(\mathcal{A}) = \bigcup_{q_f\in F} W_{I,q_f}$ holds, that is, $\mathcal{L}(\mathcal{A})$ is the union of the component languages of the vector $\mathrm{lfp}(\lambda\vec{\boldsymbol{X}}.\ \vec{\boldsymbol{\epsilon}}^I \cup \mathrm{Post}_{\mathcal{A}}(\vec{\boldsymbol{X}}))$ indexed by the final states in F.

Example 4.5. Consider again the FA $\mathcal{A}$ in Figure 1. The set of right equations for $\mathcal{A}$ is as follows:

$$\mathrm{Eqn}^{\mathrm{r}}(\mathcal{A}) = \begin{cases} X_1 = \{\epsilon\} \cup X_1 a \cup X_2 a \\ X_2 = \varnothing \cup X_1 b \cup X_2 b \end{cases}$$

so that

$$\begin{pmatrix} W_{q_1,q_1} \\ W_{q_1,q_2} \end{pmatrix} = \mathrm{lfp}\left(\lambda \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} . \begin{pmatrix} \{\epsilon\} \cup X_1 a \cup X_2 a \\ \varnothing \cup X_1 b \cup X_2 b \end{pmatrix}\right) = \begin{pmatrix} (a + (b^+a))^* \\ a^*b(b + a^+b)^* \end{pmatrix} \ . \qquad \diamond$$

In a language inclusion problem $\mathcal{L}(\mathcal{A}) \subseteq L_2$, we consider the vector $\vec{\boldsymbol{L_2}}^F \triangleq \langle \psi_{\Sigma^*}^{L_2}(q \in^? F)\rangle_{q\in Q} \in \wp(\Sigma^*)^{|Q|}$, so that, by (12), it turns out that:

$$\mathcal{L}(\mathcal{A}) \subseteq L_2 \ \Leftrightarrow\ \mathrm{lfp}(\lambda\vec{\boldsymbol{X}}.\ \vec{\boldsymbol{\epsilon}}^I \cup \mathrm{Post}_{\mathcal{A}}(\vec{\boldsymbol{X}})) \subseteq \vec{\boldsymbol{L_2}}^F \ .$$

We therefore have the following symmetric version of Theorem 4.3 for right concatenation.

Theorem 4.6. *If $\rho \in \mathrm{uco}(\wp(\Sigma^*))$ is backward complete for $\lambda X.\, Xa$ for all $a \in \Sigma$ then, for all FAs $\mathcal{A}$ on the alphabet Σ, ρ is backward complete for $\lambda\vec{\boldsymbol{X}}.\ \vec{\boldsymbol{\epsilon}}^I \cup \mathrm{Post}_{\mathcal{A}}(\vec{\boldsymbol{X}})$.*

4.3 A Language Inclusion Algorithm with Abstract Inclusion Check

Let us now apply the general Theorem 3.1 to design an algorithm that solves a language inclusion problem $\mathcal{L}(\mathcal{A}) \subseteq L_2$ by exploiting a language abstraction ρ that satisfies a list of requirements of backward completeness and computability.

Theorem 4.7. *Let $\mathcal{A} = \langle Q, \delta, I, F, \Sigma\rangle$ be a FA, $L_2 \in \wp(\Sigma^*)$ and $\rho \in \mathrm{uco}(\wp(\Sigma^*))$. Assume that the following properties hold:*

(i) *The closure ρ is backward complete for $\lambda X \in \wp(\Sigma^*).\, aX$, for all $a \in \Sigma$, and satisfies $\rho(L_2) = L_2$.*
(ii) *$\rho(\wp(\Sigma^*))$ does not contain infinite ascending chains.*
(iii) *If $X, Y \in \wp(\Sigma^*)$ are finite sets of words then the inclusion $\rho(X) \subseteq^? \rho(Y)$ is decidable.*
(iv) *If $Y \in \wp(\Sigma^*)$ is a finite set of words then the inclusion $\rho(Y) \subseteq^? L_2$ is decidable.*

Then,

$\langle Y_q\rangle_{q\in Q} := \textsc{Kleene}(\mathrm{Incl}_\rho, \lambda\vec{\boldsymbol{X}}.\vec{\boldsymbol{\epsilon}}^F \cup \mathrm{Pre}_{\mathcal{A}}(\vec{\boldsymbol{X}}), \vec{\boldsymbol{\varnothing}})$;
return $\mathrm{Incl}_\rho(\langle Y_q\rangle_{q\in Q}, \vec{\boldsymbol{L_2}}^I)$;

is a decision algorithm for $\mathcal{L}(\mathcal{A}) \subseteq L_2$.

Proof. Conditions (i), (ii) and (iii) guarantee that the hypotheses of Theorem 3.1 are satisfied. Thus, $\textsc{Kleene}(\mathrm{Incl}_\rho, \lambda\vec{X}.\, \vec{\epsilon}^F \cup \mathrm{Pre}_{\mathcal{A}}(\vec{X}), \vec{\varnothing})$ is an algorithm that terminates with output $\langle Y_q\rangle_{q\in Q}$ and

$$\rho(\mathrm{lfp}(\lambda\vec{X}.\, \vec{\epsilon}^F \cup \mathrm{Pre}_{\mathcal{A}}(\vec{X}))) = \rho(\langle Y_q\rangle_{q\in Q}) \enspace .$$

Moreover, by (10), $\mathcal{L}(\mathcal{A}) \subseteq L_2 \Leftrightarrow \rho(\mathcal{L}(\mathcal{A})) \subseteq \rho(L_2) = L_2 \Leftrightarrow \rho(\mathrm{lfp}(\lambda\vec{X}.\, \vec{\epsilon}^F \cup \mathrm{Pre}_{\mathcal{A}}(\vec{X}))) \subseteq \vec{L_2^I} \Leftrightarrow \rho(\langle Y_q\rangle_{q\in Q}) \subseteq \rho(\vec{L_2^I}) \Leftrightarrow \mathrm{Incl}_\rho(\langle Y_q\rangle_{q\in Q}, \vec{L_2^I})$. Finally, by condition (iv), $\mathrm{Incl}_\rho(\langle Y_q\rangle_{q\in Q}, \vec{L_2^I})$ is decidable. ☐

It is worth noticing that Theorem 4.7 can also be stated in a symmetric version for $\lambda\vec{X}.\, \vec{\epsilon}^I \cup \mathrm{Post}_{\mathcal{A}}(\vec{X})$ similarly to Theorem 4.6.

5 INSTANTIATING THE FRAMEWORK WITH QUASIORDERS

We instantiate the general algorithmic framework of Section 4 to the class of closure operators induced by quasiorder relations on words.

5.1 Word-based Abstractions

Let $\leqslant \,\subseteq \Sigma^* \times \Sigma^*$ be a quasiorder relation on words in Σ^*. The corresponding closure operator $\rho_\leqslant \in \mathrm{uco}(\wp(\Sigma^*))$ is defined as follows:

$$\rho_\leqslant(X) \triangleq \{v \in \Sigma^* \mid \exists u \in X,\ u \leqslant v\} \enspace . \tag{13}$$

Thus, $\rho_\leqslant(X)$ is the $\leqslant$-upward closure of X and it is easy to check that $\rho_\leqslant$ is indeed a closure on the complete lattice $\langle \wp(\Sigma^*), \subseteq\rangle$.

Following [de Luca and Varricchio 1994], a quasiorder $\leqslant$ on Σ^* is *left-monotonic* (resp. *right-monotonic*) if

$$\forall y, x_1, x_2 \in \Sigma^*,\ x_1 \leqslant x_2 \Rightarrow yx_1 \leqslant yx_2 \quad (\text{resp. } x_1y \leqslant x_2y) \enspace .$$

Also, $\leqslant$ is called monotonic if it is both left- and right-monotonic. It turns out that $\leqslant$ is left-monotonic (resp. right-monotonic) iff

$$\forall x_1, x_2 \in \Sigma^*, \forall a \in \Sigma,\ x_1 \leqslant x_2 \Rightarrow ax_1 \leqslant ax_2 \quad (\text{resp. } x_1a \leqslant x_2a) \enspace . \tag{14}$$

In fact, if $x_1 \leqslant x_2$ then (14) implies that for all $y \in \Sigma^*$, $yx_1 \leqslant yx_2$: by induction on the length $|y| \in \mathbb{N}$, we have that: (i) if $y = \epsilon$ then $yx_1 \leqslant yx_2$; (ii) if $y = av$ with $a \in \Sigma, v \in \Sigma^*$ then, by inductive hypothesis, $vx_1 \leqslant vx_2$, so that by (14), $yx_1 = avx_1 \leqslant avx_2 = yx_2$.

Definition 5.1 (L-Consistent Quasiorder). Let $L \in \wp(\Sigma^*)$. A quasiorder $\leqslant_L \subseteq \Sigma^* \times \Sigma^*$ is called *left* (resp. *right*) *L-consistent* when:
(a) $\leqslant_L \cap (L \times \neg L) = \varnothing$;
(b) $\leqslant_L$ is left-monotonic (resp. right-monotonic).
Moreover, $\leqslant_L$ is called *L-consistent* when it is both left and right L-consistent. ■

It turns out that a quasiorder is L-consistent iff it induces a closure which includes L in its image and it is backward complete for concatenation.

Lemma 5.2. *Let $L \in \wp(\Sigma^*)$ and $\leqslant_L$ be a quasiorder on Σ^*. Then, $\leqslant_L$ is a left (resp. right) L-consistent quasiorder on Σ^* if and only if*
(a) $\rho_{\leqslant_L}(L) = L$, and
(b) $\rho_{\leqslant_L}$ is backward complete for $\lambda X.\, aX$ (resp. $\lambda X.\, Xa$) for all $a \in \Sigma$.

Proof. We consider the left case, the right case is symmetric.

(a) The inclusion $L \subseteq \rho_{\leqslant_L}(L)$ always holds because $\rho_{\leqslant_L}$ is an upper closure. Then, it turns out that $\rho_{\leqslant_L}(L) = L$ iff $\rho_{\leqslant_L}(L) \subseteq L$ iff $\forall v \in \Sigma^*, (\exists u \in L,\, u \leqslant_L v) \Rightarrow v \in L$ iff $\leqslant_L \cap (L \times \neg L) = \varnothing$. Thus, $\rho_{\leqslant_L}(L) = L$ iff condition (a) of Definition 5.1 holds.

(b) We first prove that if $\leqslant_L$ is left-monotonic then, for all $X \in \wp(\Sigma^*)$, $\rho_{\leqslant_L}(aX) = \rho_{\leqslant_L}(a\rho_{\leqslant_L}(X))$ for all $a \in \Sigma$. Monotonicity of concatenation together with monotonicity and extensivity of $\rho_{\leqslant_L}$ imply that $\rho_{\leqslant_L}(aX) \subseteq \rho_{\leqslant_L}(a\rho_{\leqslant_L}(X))$ holds. For the reverse inclusion, we have that:

$$\begin{aligned} \rho_{\leqslant_L}(a\rho_{\leqslant_L}(X)) = &\quad \text{[by def. of } \rho_{\leqslant_L}] \\ \rho_{\leqslant_L}(\{ay \mid \exists x \in X, x \leqslant_L y\}) = &\quad \text{[by def. of } \rho_{\leqslant_L}] \\ \{z \mid \exists x \in X, y \in \Sigma^*,\, x \leqslant_L y \wedge ay \leqslant_L z\} \subseteq &\quad \text{[by left monotonicity of } \leqslant_L] \\ \{z \mid \exists x \in X, y \in \Sigma^*,\, ax \leqslant_L ay \wedge ay \leqslant_L z\} = &\quad \text{[by transitivity of } \leqslant_L] \\ \{z \mid \exists x \in X, ax \leqslant_L z\} = &\quad \text{[by def. of } \rho_{\leqslant_L}] \\ \rho_{\leqslant_L}(aX) &\quad . \end{aligned}$$

Conversely, assume that for all $X \in \wp(\Sigma^*)$ and $a \in \Sigma$, $\rho_{\leqslant_L}(aX) = \rho_{\leqslant_L}(a\rho_{\leqslant_L}(X))$. Consider $x_1, x_2 \in \Sigma^*$ and $a \in \Sigma$. If $x_1 \leqslant_L x_2$ then $\{x_2\} \subseteq \rho_{\leqslant_L}(\{x_1\})$, and, in turn, $a\{x_2\} \subseteq a\rho_{\leqslant_L}(\{x_1\})$. Then, by applying the monotonic function $\rho_{\leqslant_L}$, $\rho_{\leqslant_L}(a\{x_2\}) \subseteq \rho_{\leqslant_L}(a\rho_{\leqslant_L}(\{x_1\}))$, so that, by backward completeness, $\rho_{\leqslant_L}(a\{x_2\}) \subseteq \rho_{\leqslant_L}(a\{x_1\})$. Hence, $a\{x_2\} \subseteq \rho_{\leqslant_L}(a\{x_1\})$, namely, $ax_1 \leqslant_L ax_2$. By (14), this shows that $\leqslant_L$ is left-monotonic. □

We can apply Theorem 4.7 to the closure $\rho_{\leqslant^l_{L_2}}$ induced by a left L_2-consistent well-quasiorder, since it satisfies all the required hypotheses, thus obtaining the following Algorithm `FAIncW` which solves the language inclusion problem $\mathcal{L}(\mathcal{A}) \subseteq L_2$ for any automaton $\mathcal{A}$. This algorithm is called "word-based" because the output vector $\langle Y_q \rangle_{q \in Q}$ computed by Kleene consists of finite sets of words. Here, the convergence relation $\mathrm{Incl}_{\rho_{\leqslant^l_{L_2}}}$ of Kleene coincides with the relation $\sqsubseteq_{\leqslant^l_{L_2}}$ because $\mathrm{Incl}_{\rho_{\leqslant^l_{L_2}}}(X, Y)$ iff $\rho_{\leqslant^l_{L_2}}(X) \subseteq \rho_{\leqslant^l_{L_2}}(Y)$ iff $X \sqsubseteq_{\leqslant^l_{L_2}} Y$.

`FAIncW`: Word-based algorithm for $\mathcal{L}(\mathcal{A}) \subseteq L_2$

Data: FA $\mathcal{A} = \langle Q, \delta, I, F, \Sigma \rangle$; decision procedure for $u \in^? L_2$; decidable left L_2-consistent wqo $\leqslant^l_{L_2}$.

```
⟨Y_q⟩_{q∈Q} := Kleene(⊑_{≤^l_{L_2}}, λX⃗. ε⃗^F ∪ Pre_𝒜(X⃗), ∅⃗);
forall q ∈ I do
    forall u ∈ Y_q do
        if u ∉ L_2 then return false;
return true;
```

Theorem 5.3. *Let $\mathcal{A} = \langle Q, \delta, I, F, \Sigma \rangle$ be a FA and $L_2 \in \wp(\Sigma^*)$ be a language such that:* (i) *membership in L_2 is decidable;* (ii) *there exists a decidable left L_2-consistent wqo on Σ^*. Then, Algorithm* `FAIncW` *decides the inclusion problem $\mathcal{L}(\mathcal{A}) \subseteq L_2$.*

Proof. Let $\leqslant^l_{L_2}$ be the decidable left L_2-consistent wqo on Σ^*. Let us check that the hypotheses (i)-(ii)-(iii) of Theorem 4.7 are satisfied.

(i) It follows from hypothesis (ii) and Lemma 5.2 that $\leqslant^l_{L_2}$ is backward complete for left concatenation and satisfies $\rho_{\leqslant^l_{L_2}}(L_2) = L_2$.

(ii) Since $\leqslant^l_{L_2}$ is a well-quasiorder, it follows that $\{\rho_{\leqslant^l_{L_2}}(S) \mid S \in \wp(\Sigma^*)\}$ does not contain infinite ascending chains.

(iii) For finite sets X and Y, the abstract inclusion $\mathrm{Incl}_{\rho_{\leqslant^l_{L_2}}}(X, Y) \Leftrightarrow X \sqsubseteq_{\leqslant^l_{L_2}} Y$ is decidable since $\leqslant^l_{L_2}$ is a decidable wqo.

Moreover, it turns out that the check $\mathrm{Incl}_{\rho_{\leqslant^l_{L_2}}}(\langle Y_q \rangle_{q\in Q}, \overrightarrow{L_2}^I)$ of Theorem 4.7 is decidable and is performed by lines 2-5 of Algorithm `FAIncW`. Indeed, since, by Theorem 4.7, $\textsc{Kleene}(\sqsubseteq_{\leqslant^l_{L_2}}, \lambda\vec{X}.\, \vec{\epsilon}^F \cup \mathrm{Pre}_{\mathcal{A}}(\vec{X}), \vec{\varnothing})$ terminates after a finite number of steps with output $\langle Y_q \rangle_{q\in Q}$, each set of words Y_q of the output turns out to be finite. Also, since $\overrightarrow{L_2}^I = \langle \psi^{L_2}_{\Sigma^*}(q \in^? I)\rangle_{q\in Q}$, the abstract inclusion trivially holds for all components Y_q with $q \notin I$. Therefore, it suffices to check whether $Y_q \sqsubseteq_{\leqslant^l_{L_2}} L_2$ holds for all $q \in I$. Since $Y_q \sqsubseteq_{\leqslant^l_{L_2}} L_2$ iff $\rho_{\leqslant^l_{L_2}}(Y_q) \subseteq \rho_{\leqslant^l_{L_2}}(L_2) = L_2$ iff $Y_q \subseteq L_2$, and since Y_q is a finite set, $Y_q \sqsubseteq_{\leqslant^l_{L_2}} L_2$ can be decided by performing the finitely many membership check $u \in^? L_2$ at lines 2-5, where, by hypothesis (ii), any membership check is decidable. Thus, hypothesis (iv) of Theorem 4.7 is satisfied.

Summing up, we have shown that Algorithm `FAIncW` decides the inclusion $\mathcal{L}(\mathcal{A}) \subseteq L_2$. □

Remark 5.4. It is worth noticing that in each iteration of $\textsc{Kleene}(\sqsubseteq_{\leqslant^l_{L_2}}, \lambda\vec{X}.\, \vec{\epsilon}^F \cup \mathrm{Pre}_{\mathcal{A}}(\vec{X}), \vec{\varnothing})$ in Algorithm `FAIncW`, in the current vector $\langle Y_q \rangle_{q\in Q}$ one could safely remove from a component Y_q any word $w \in Y_q$ such that there exists a word $u \in Y_q$ such that $u \leqslant^l_{L_2} w$ and $u \neq w$. This enables replacing each finite set Y_q occurring in Kleene iterates with a minor subset $\lfloor Y_q \rfloor$ w.r.t. $\leqslant^l_{L_2}$. This replacement is correct, namely, Theorem 5.3 still holds for the corresponding modified language inclusion algorithm, because an inclusion check $X \sqsubseteq_{\leqslant^l_{L_2}} Y$ holds iff the check $\lfloor X \rfloor \sqsubseteq_{\leqslant^l_{L_2}} \lfloor Y \rfloor$ for the corresponding minor subsets holds. ◇

5.1.1 Right Concatenation. Following Section 4.2.1, a symmetric version, called `FAIncWr`, of the algorithm `FAIncW` (and of Theorem 5.3) for *right* L_2-consistent wqos can be given as follows.

`FAIncWr`: Word-based algorithm for $\mathcal{L}(\mathcal{A}) \subseteq L_2$

Data: FA $\mathcal{A} = \langle Q, \delta, I, F, \Sigma\rangle$; decision procedure for $u \in^? L_2$; decidable right L_2-consistent wqo $\leqslant^r_{L_2}$.

$\langle Y_q \rangle_{q\in Q} := \textsc{Kleene}(\sqsubseteq_{\leqslant^r_{L_2}}, \lambda\vec{X}.\, \vec{\epsilon}^I \cup \mathrm{Post}_{\mathcal{A}}(\vec{X}), \vec{\varnothing})$;
forall $q \in F$ **do**
  forall $u \in Y_q$ **do**
    if $u \notin L_2$ **then return** *false*;
return *true*;

Theorem 5.5. *Let $\mathcal{A} = \langle Q, \delta, I, F, \Sigma\rangle$ be a FA and $L_2 \in \wp(\Sigma^*)$ be a language such that:* (i) *membership in L_2 is decidable;* (ii) *there exists a decidable right L_2-consistent wqo on Σ^*.Then, Algorithm* `FAIncWr` *decides the inclusion problem $\mathcal{L}(\mathcal{A}) \subseteq L_2$.*

In the following, we will consider different quasiorders on Σ^* and we will show that they fulfill the requirements of Theorem 5.3, therefore yielding algorithms for solving language inclusion problems.

5.2 Nerode Quasiorders

The notions of *left* and *right quotient* of a language $L \in \wp(\Sigma^*)$ w.r.t. a word $w \in \Sigma^*$ are standard:

$$w^{-1}L \triangleq \{u \in \Sigma^* \mid wu \in L\}, \qquad Lw^{-1} \triangleq \{u \in \Sigma^* \mid uw \in L\} \ .$$

Correspondingly, let us define the following quasiorder relations on Σ^*:

$$u \leqq_L^l v \overset{\triangle}{\iff} Lu^{-1} \subseteq Lv^{-1}, \qquad u \leqq_L^r v \overset{\triangle}{\iff} u^{-1}L \subseteq v^{-1}L \ . \tag{15}$$

De Luca and Varricchio [1994, Section 2] call them, resp., the *left* ($\leqq_L^l$) and *right* ($\leqq_L^r$) *Nerode quasiorders relative to L*. The following result shows that Nerode quasiorders are the weakest (i.e., greatest w.r.t. set inclusion of binary relations) L_2-consistent quasiorders for which the algorithm FAIncW can be instantiated to decide a language inclusion $\mathcal{L}(\mathcal{A}) \subseteq L_2$.

Lemma 5.6. *Let $L \in \wp(\Sigma^*)$.*

(a) *$\leqq_L^l$ and $\leqq_L^r$ are, resp., left and right L-consistent quasiorders. If L is regular then, additionally, $\leqq_L^l$ and $\leqq_L^r$ are decidable wqos.*
(b) *If $\leqslant$ is a left (resp. right) L-consistent quasiorder on Σ^* then $\rho_{\leqq_L^l}(\wp(\Sigma^*)) \subseteq \rho_{\leqslant}(\wp(\Sigma^*))$ (resp. $\rho_{\leqq_L^r}(\wp(\Sigma^*)) \subseteq \rho_{\leqslant}(\wp(\Sigma^*))$).*

Proof. Let us consider point (a). De Luca and Varricchio [1994, Section 2] observe that $\leqq_L^l$ and $\leqq_L^r$ are, resp., left and right monotonic. Moreover, De Luca and Varricchio [1994, Theorem 2.4] show that if L is regular then both $\leqq_L^l$ and $\leqq_L^r$ are wqos. Let us also observe that given $u \in L$ and $v \notin L$ we have that $\epsilon \in Lu^{-1}$ and $\epsilon \in u^{-1}L$ while $\epsilon \notin Lv^{-1}$ and $\epsilon \notin v^{-1}L$. Hence, $\leqq_L^l$ ($\leqq_L^r$) is a left (right) L-consistent quasiorder. Finally, if L is regular then both relations are clearly decidable.
Let us now consider point (b) for the left case (the right case is symmetric). By the characterization of left consistent quasiorders given by Lemma 5.2, De Luca and Varricchio [1994, Section 2, point 4] observe that $\leqq_L^l$ is maximum in the set of all left L-consistent quasiorders, i.e., every left L-consistent quasiorder $\leqslant$ is such that $x \leqslant y \Rightarrow x \leqq_L^l y$. As a consequence, $\rho_{\leqslant}(X) \subseteq \rho_{\leqq_L^l}(X)$ holds for all $X \in \wp(\Sigma^*)$, namely, $\rho_{\leqq_L^l}(\wp(\Sigma^*)) \subseteq \rho_{\leqslant}(\wp(\Sigma^*))$. □

This allows us to derive a first instantiation of Theorem 5.3. Because membership is decidable for regular languages L_2, Lemma 5.6 (a) for $\leqq_{L_2}^l$ implies that the hypotheses (i) and (ii) of Theorem 5.3 are satisfied, so that the algorithm FAIncW instantiated to $\leqq_{L_2}^l$ decides the inclusion $\mathcal{L}(\mathcal{A}) \subseteq L_2$ when L_2 is regular. Furthermore, under these hypotheses, Lemma 5.6 (b) shows that $\leqq_{L_2}^l$ is the weakest left L_2-consistent quasiorder relation on Σ^* for which the algorithm FAIncW can be instantiated for deciding an inclusion $\mathcal{L}(\mathcal{A}) \subseteq L_2$.

Example 5.7. We illustrate the use of the left Nerode quasiorder in Algorithm FAIncW for solving the language inclusion $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$, where $\mathcal{A}_1$ and $\mathcal{A}_2$ are the FAs shown in Figure 2. The equations for $\mathcal{A}_1$ are as follows:

$$\mathrm{Eqn}(\mathcal{A}_1) = \begin{cases} X_1 = \varnothing \cup aX_1 \cup aX_2 \cup bX_2 \cup cX_2 \\ X_2 = \{\epsilon\} \end{cases} .$$

We have the following quotients (among others) for $L = \mathcal{L}(\mathcal{A}_2) = a^*(a(a+b)^*a + a^+c + ab + bb)$:

$$\begin{aligned} L\epsilon^{-1} &= a^*(a(a+b)^*a + a^+c + ab + bb) & Lb^{-1} &= a^*(a+b) \\ La^{-1} &= a^*a(a+b)^* = a^+(a+b)^* & Lc^{-1} &= a^*a^+ = a^+ \\ Lw^{-1} &= a^* \text{ iff } w \in (a(a+b)^*a + ac + ab + bb) \end{aligned}$$

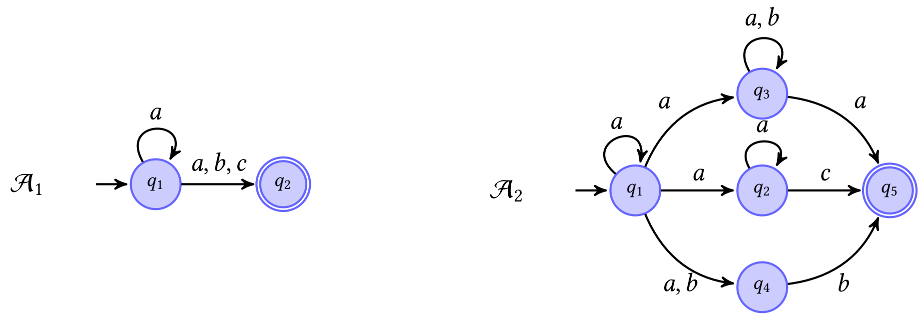


Fig. 2. Two automata $\mathcal{A}_1$ (left) and $\mathcal{A}_2$ (right) generating the regular languages $\mathcal{L}(\mathcal{A}_1) = a^*(a+b+c)$ and $\mathcal{L}(\mathcal{A}_2) = a^*(a(a+b)^*a + a^+c + ab + bb)$.

Hence, among others, the following relations hold: $c \leqq^l_L a$, $c \leqq^l_L b$ and $c \leqq^l_L w$ for all $w \in (a(a+b)^*a + ac + ab + bb)$. Then, let us show the computation of the Kleene iterates performed by the Algorithm `FAIncW`.

$$\begin{aligned}
\vec{Y}^{(0)} &= \vec{\varnothing} \\
\vec{Y}^{(1)} &= \vec{\epsilon}^F = \langle \varnothing, \{\epsilon\} \rangle \\
\vec{Y}^{(2)} &= \vec{\epsilon}^F \cup \mathrm{Pre}_{\mathcal{A}_1}(\vec{Y}^{(1)}) = \langle \varnothing, \{\epsilon\} \rangle \cup \langle \varnothing \cup a\varnothing \cup a\{\epsilon\} \cup b\{\epsilon\} \cup c\{\epsilon\}, \{\epsilon\} \rangle \\
&= \langle \{a, b, c\}, \{\epsilon\} \rangle \\
\vec{Y}^{(3)} &= \vec{\epsilon}^F \cup \mathrm{Pre}_{\mathcal{A}_1}(\vec{Y}^{(2)}) = \langle \varnothing, \{\epsilon\} \rangle \cup \langle \varnothing \cup a\{a, b, c\} \cup a\{\epsilon\} \cup b\{\epsilon\} \cup c\{\epsilon\}, \{\epsilon\} \rangle \\
&= \langle \{aa, ab, ac, a, b, c\}, \{\epsilon\} \rangle
\end{aligned}$$

It turns out that $\langle \{aa, ab, ac, a, b, c\}, \{\epsilon\} \rangle \sqsubseteq_{\leqq^l_L} \langle \{a, b, c\}, \{\epsilon\} \rangle$ because $c \leqq^l_L aa$, $c \leqq^l_L ab$ and $c \leqq^l_L ac$ hold, so that Kleene stops with $\vec{Y}^{(3)}$ and outputs $\vec{Y} = \langle \{a, b, c\}, \{\epsilon\} \rangle$. Since $c \in \vec{Y}_1$ and $c \notin \mathcal{L}(\mathcal{A}_2)$, the Algorithm `FAIncW` correctly concludes that $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$ does not hold. ◇

5.2.1 On the Complexity of Nerode quasiorders. For the inclusion problem between languages generated by finite automata, deciding the (left or right) Nerode quasiorder relation between words can be easily shown to be as hard as the language inclusion problem itself, which is PSPACE-complete. In fact, given the automata $\mathcal{A}_1 = (Q_1, \delta_1, I_1, F_1, \Sigma)$ and $\mathcal{A}_2 = (Q_2, \delta_2, I_2, F_2, \Sigma)$, one can define the union automaton $\mathcal{A}_3 \triangleq (Q_1 \cup Q_2 \cup \{q^\iota\}, \delta_3, \{q^\iota\}, F_1 \cup F_2)$ where δ_3 maps (q^ι, a) to I_1, (q^ι, b) to I_2 and behaves like δ_1 or δ_2 elsewhere. Then, it turns out that $a \leqq^r_{\mathcal{L}(\mathcal{A}_3)} b \Leftrightarrow a^{-1}\mathcal{L}(\mathcal{A}_3) \subseteq b^{-1}\mathcal{L}(\mathcal{A}_3) \Leftrightarrow \mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$.

Also, for the inclusion problem of a language generated by an automaton within the trace set of a one-counter net (cf. Section 5.4), the right Nerode quasiorder is a right language-consistent well-quasiorder but it turns out to be undecidable (cf. Lemma 5.16).

5.3 State-based Quasiorders

Consider an inclusion problem $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$ where $\mathcal{A}_1$ and $\mathcal{A}_2$ are FAs. In the following, we study a class of well-quasiorders based on $\mathcal{A}_2$, that we call state-based quasiorders. These quasiorders are strictly stronger (i.e., lower w.r.t. set inclusion of binary relations) than the Nerode quasiorders defined in Section 5.2 and sidestep the untractability or undecidability of Nerode quasiorders (cf. Section 5.2.1) yet allowing to define an algorithm solving the language inclusion $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$.

5.3.1 Inclusion in Regular Languages. We define the quasiorders $\leq^l_{\mathcal{A}}$ and $\leq^r_{\mathcal{A}}$ on Σ^* induced by a FA $\mathcal{A} = \langle Q, \delta, I, F, \Sigma \rangle$ as follows: for all $u, v \in \Sigma^*$,

$$u \leq^l_{\mathcal{A}} v \overset{\triangle}{\iff} \mathrm{pre}^{\mathcal{A}}_u(F) \subseteq \mathrm{pre}^{\mathcal{A}}_v(F) \qquad\qquad u \leq^r_{\mathcal{A}} v \overset{\triangle}{\iff} \mathrm{post}^{\mathcal{A}}_u(I) \subseteq \mathrm{post}^{\mathcal{A}}_v(I) \tag{16}$$

where, for all $X \in \wp(Q)$, $\mathrm{pre}^{\mathcal{A}}_u(X) \triangleq \{q \in Q \mid u \in W^{\mathcal{A}}_{q,X}\}$ and $\mathrm{post}^{\mathcal{A}}_u(X) \triangleq \{q' \in Q \mid u \in W^{\mathcal{A}}_{X,q'}\}$ denote, resp., the standard predecessor and successor state transformers in $\mathcal{A}$. The superscripts in $\leq^l_{\mathcal{A}}$ and $\leq^r_{\mathcal{A}}$ stand, resp., for left/right because the following result holds.

Lemma 5.8. *The relations $\leq^l_{\mathcal{A}}$ and $\leq^r_{\mathcal{A}}$ are, resp., decidable left and right $\mathcal{L}(\mathcal{A})$-consistent wqos.*

Proof. Since, for every $u \in \Sigma^*$, $\mathrm{pre}^{\mathcal{A}}_u(F)$ is a finite and computable set, it turns out that $\leq^l_{\mathcal{A}}$ is a decidable wqo. Let us check that $\leq^l_{\mathcal{A}}$ is left $\mathcal{L}(A)$-consistent according to Definition 5.1 (a)-(b).
(a) By picking $u \in \mathcal{L}(\mathcal{A})$ and $v \notin \mathcal{L}(\mathcal{A})$ we have that $\mathrm{pre}^{\mathcal{A}}_u(F)$ contains some initial state while $\mathrm{pre}^{\mathcal{A}}_v(F)$ does not, hence $u \not\leq^l_{\mathcal{A}} v$.
(b) Let us check that $\leq^l_{\mathcal{A}}$ is left monotonic. Observe that, for all $x \in \Sigma^*$, $\mathrm{pre}^{\mathcal{A}}_x$ is a monotonic function and that

$$\mathrm{pre}^{\mathcal{A}}_{uv} = \mathrm{pre}^{\mathcal{A}}_u \circ \mathrm{pre}^{\mathcal{A}}_v \quad . \tag{17}$$

Therefore, for all $x_1, x_2 \in \Sigma^*$ and $a \in \Sigma$,

$$\begin{aligned} x_1 \leq^l_{\mathcal{A}} x_2 \Rightarrow &\quad [\text{by def. of } \leq^l_{\mathcal{A}}] \\ \mathrm{pre}^{\mathcal{A}}_{x_1}(F) \subseteq \mathrm{pre}^{\mathcal{A}}_{x_2}(F) \Rightarrow &\quad [\text{as } \mathrm{pre}^{\mathcal{A}}_a \text{ is monotonic}] \\ \mathrm{pre}^{\mathcal{A}}_a(\mathrm{pre}^{\mathcal{A}}_{x_1}(F)) \subseteq \mathrm{pre}^{\mathcal{A}}_a(\mathrm{pre}^{\mathcal{A}}_{x_2}(F)) \Leftrightarrow &\quad [\text{by (17)}] \\ \mathrm{pre}^{\mathcal{A}}_{ax_1}(F) \subseteq \mathrm{pre}^{\mathcal{A}}_{ax_2}(F) \Leftrightarrow &\quad [\text{by def. of } \leq^l_{\mathcal{A}}] \\ ax_1 \leq^l_{\mathcal{A}} ax_2 \quad . & \end{aligned}$$

The proof that $\leq^r_{\mathcal{A}}$ is a decidable right $\mathcal{L}(\mathcal{A})$-consistent quasiorder is symmetric. □

As a consequence, Theorem 5.3 applies to the wqo $\leq^l_{\mathcal{A}_2}$ (and $\leq^r_{\mathcal{A}_2}$), so that one can instantiate the algorithm `FAIncW` to $\leq^l_{\mathcal{A}_2}$ for deciding an inclusion $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$.

Turning back to the left Nerode wqo $\leqq^l_{\mathcal{L}(\mathcal{A}_2)}$, it turns out that the following equivalences hold:

$$u \leqq^l_{\mathcal{L}(\mathcal{A}_2)} v \Leftrightarrow \mathcal{L}(\mathcal{A}_2)u^{-1} \subseteq \mathcal{L}(\mathcal{A}_2)v^{-1} \Leftrightarrow W_{I,\mathrm{pre}^{\mathcal{A}_2}_u(F)} \subseteq W_{I,\mathrm{pre}^{\mathcal{A}_2}_v(F)} \quad .$$

Since $\mathrm{pre}^{\mathcal{A}_2}_u(F) \subseteq \mathrm{pre}^{\mathcal{A}_2}_v(F)$ entails $W_{I,\mathrm{pre}^{\mathcal{A}_2}_u(F)} \subseteq W_{I,\mathrm{pre}^{\mathcal{A}_2}_v(F)}$, it follows that $u \leq^l_{\mathcal{A}_2} v \Rightarrow u \leqq^l_{\mathcal{L}(\mathcal{A}_2)} v$ and, in turn, $\rho_{\leqq^l_{\mathcal{L}(\mathcal{A}_2)}}(\wp(\Sigma^*)) \subseteq \rho_{\leq^l_{\mathcal{A}_2}}(\wp(\Sigma^*))$.

Example 5.9. We illustrate the left state-based quasiorder by using it to solve the language inclusion $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$ of Example 5.7. We have, among others, the following sets of predecessors of $F_{\mathcal{A}_2}$:

$$\begin{array}{llll} \mathrm{pre}^{\mathcal{A}_2}_{\epsilon}(F_{\mathcal{A}_2}) = \{q_5\} & \mathrm{pre}^{\mathcal{A}_2}_{a}(F_{\mathcal{A}_2}) = \{q_3\} & \mathrm{pre}^{\mathcal{A}_2}_{b}(F_{\mathcal{A}_2}) = \{q_4\} & \mathrm{pre}^{\mathcal{A}_2}_{c}(F_{\mathcal{A}_2}) = \{q_2\} \\ \mathrm{pre}^{\mathcal{A}_2}_{aa}(F_{\mathcal{A}_2}) = \{q_1, q_3\} & \mathrm{pre}^{\mathcal{A}_2}_{ab}(F_{\mathcal{A}_2}) = \{q_1\} & \mathrm{pre}^{\mathcal{A}_2}_{ac}(F_{\mathcal{A}_2}) = \{q_1, q_2\} & \mathrm{pre}^{\mathcal{A}_2}_{aaa}(F_{\mathcal{A}_2}) = \{q_1, q_3\} \\ \mathrm{pre}^{\mathcal{A}_2}_{aab}(F_{\mathcal{A}_2}) = \{q_1\} & \mathrm{pre}^{\mathcal{A}_2}_{aac}(F_{\mathcal{A}_2}) = \{q_1, q_2\} & & \end{array}$$

Recall from Example 5.7 that, for the Nerode quasiorder, we have $c \leqq^l_{\mathcal{L}(\mathcal{A}_2)} b$, $c \leqq^l_{\mathcal{L}(\mathcal{A}_2)} a$ while none of these relations hold for $\leq^l_{\mathcal{A}_2}$.

Let us next show the Kleene iterates computed by Algorithm FAIncW when using the quasiorder $\leq^l_{\mathcal{A}_2}$.

$$\begin{aligned}
\vec{Y}^{(0)} &= \vec{\varnothing} \\
\vec{Y}^{(1)} &= \vec{\epsilon}^F = \langle \varnothing, \{\epsilon\}\rangle \\
\vec{Y}^{(2)} &= \vec{\epsilon}^F \cup \mathrm{Pre}_{\mathcal{A}_1}(\vec{Y}^{(1)}) = \langle \{a,b,c\}, \{\epsilon\}\rangle \\
\vec{Y}^{(3)} &= \vec{\epsilon}^F \cup \mathrm{Pre}_{\mathcal{A}_1}(\vec{Y}^{(2)}) = \langle \{aa,ab,ac,a,b,c\}, \{\epsilon\}\rangle \\
\vec{Y}^{(4)} &= \vec{\epsilon}^F \cup \mathrm{Pre}_{\mathcal{A}_1}(\vec{Y}^{(3)}) = \langle \{aaa,aab,aac,aa,ab,ac,a,b,c\}, \{\epsilon\}\rangle
\end{aligned}$$

It turns out that $\langle \{aaa,aab,aac,aa,ab,ac,a,b,c\}, \{\epsilon\}\rangle \sqsubseteq_{\leq^l_{\mathcal{A}_2}} \langle \{aa,ab,ac,a,b,c\}, \{\epsilon\}\rangle$, so that Kleene outputs the vector $\vec{Y} = \langle \{aa,ab,ac,a,b,c\}, \{\epsilon\}\rangle$. Since $c \in \vec{Y}_0$ and $c \notin \mathcal{L}(\mathcal{A}_2)$, Algorithm FAIncW concludes that the language inclusion $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$ does not hold. $\diamond$

5.3.2 Simulation-based Quasiorders. Recall that a *simulation* on a FA $\mathcal{A} = \langle Q, \delta, I, F, \Sigma\rangle$ is a binary relation $\leq \subseteq Q \times Q$ such that for all $p, q \in Q$ such that $p \leq q$ the following two conditions hold:

(i) if $p \in F$ then $q \in F$;

(ii) for every transition $p \xrightarrow{a} p'$, there exists a transition $q \xrightarrow{a} q'$ such that $p' \leq q'$.

It is well known that simulation relations are closed under arbitrary unions, where the greatest (w.r.t. inclusion) simulation relation $\leq_A \triangleq \cup\{\leq \subseteq Q\times Q \mid \leq \text{ is a simulation on } \mathcal{A}\}$ is a quasiorder, called simulation quasiorder of $\mathcal{A}$. It is also well known that simulation implies language inclusion, i.e., if $\leq$ is a simulation on $\mathcal{A}$ then

$$q \leq q' \Rightarrow W^{\mathcal{A}}_{q,F} \subseteq W^{\mathcal{A}}_{q',F} \ .$$

A relation $\leq \subseteq Q \times Q$ on states can be lifted in the standard universal/existential way to a relation $\leq^{\forall\exists} \subseteq \wp(Q) \times \wp(Q)$ on sets of states as follows:

$$X \leq^{\forall\exists} Y \stackrel{\triangle}{\iff} \forall x \in X, \exists y \in Y,\ x \leq y \ .$$

In particular, if $\leq$ is a quasiorder then $\leq^{\forall\exists}$ is a quasiorder as well. Also, if $\leq$ is a simulation relation then its lifting $\leq^{\forall\exists}$ is such that $X \leq^{\forall\exists} Y \Rightarrow W^{\mathcal{A}}_{X,F} \subseteq W^{\mathcal{A}}_{Y,F}$ holds. This suggests us to define a *right simulation-based quasiorder* $\leq^r_{\mathcal{A}}$ on Σ^* induced by a simulation $\leq$ on $\mathcal{A}$ as follows: for all $u, v \in \Sigma^*$,

$$u \leq^r_{\mathcal{A}} v \stackrel{\triangle}{\iff} \mathrm{post}^{\mathcal{A}}_u(I) \leq^{\forall\exists} \mathrm{post}^{\mathcal{A}}_v(I) \ . \tag{18}$$

Lemma 5.10. *Given a simulation relation $\leq$ on $\mathcal{A}$, the right simulation-based quasiorder $\leq^r_{\mathcal{A}}$ is a decidable right $\mathcal{L}(\mathcal{A})$-consistent wqo.*

Proof. Let $u \in \mathcal{L}(\mathcal{A})$ and $v \notin \mathcal{L}(\mathcal{A})$, so that $F \cap \mathrm{post}^{\mathcal{A}}_u(I) \neq \varnothing$ and $(F \cap \mathrm{post}^{\mathcal{A}}_v(I)) = \varnothing$ hold. Hence, there exists $q \in \mathrm{post}^{\mathcal{A}}_u(F) \cap F$ such that $q \leq^r_{\mathcal{A}} q'$ for no $q' \in \mathrm{post}^{\mathcal{A}}_v(F)$ since, by simulation, this would imply $q' \in \mathrm{post}^{\mathcal{A}}_v(F) \cap F$, which would contradict $F \cap \mathrm{post}^{\mathcal{A}}_v(I) = \varnothing$. Therefore, $u \not\leq^r_{\mathcal{A}} v$ holds.

Next, we show that $\leq^r_{\mathcal{A}}$ is right monotonic. By (14), we check that for all $u, v \in \Sigma^*$ and $a \in \Sigma$, $u \leq^r_{\mathcal{A}} v \Rightarrow ua \leq^r_{\mathcal{A}} va$:

$$\begin{aligned}
u \leq^r_{\mathcal{A}} v \Leftrightarrow &\quad [\text{by def. } \leq^r_{\mathcal{A}}] \\
\mathrm{post}^{\mathcal{A}}_u(I) \leq^{\forall\exists} \mathrm{post}^{\mathcal{A}}_v(I) \Leftrightarrow &\quad [\text{by def. of } \leq^{\forall\exists}] \\
\forall x \in \mathrm{post}^{\mathcal{A}}_u(I), \exists y \in \mathrm{post}^{\mathcal{A}}_v(I), x \leq y \Rightarrow &\quad [\text{by def. of } \leq] \\
\forall x \xrightarrow{a} x',\ x \in \mathrm{post}^{\mathcal{A}}_u(I),\ \exists y \xrightarrow{a} y',\ y \in \mathrm{post}^{\mathcal{A}}_v(u), x' \leq y' \Leftrightarrow &\quad [\text{by } \mathrm{post}^{\mathcal{A}}_u \circ \mathrm{post}^{\mathcal{A}}_a = \mathrm{post}^{\mathcal{A}}_{ua}(I)]
\end{aligned}$$

$$
\begin{aligned}
\forall x' \in \mathrm{post}^{\mathcal{A}}_{ua}(I), \exists y' \in \mathrm{post}^{\mathcal{A}}_{va}(I),\ x' \leq y' \Leftrightarrow &\quad \text{[by def. of } \leq^{\forall\exists}\text{]}\\
\mathrm{post}^{\mathcal{A}}_{ua}(I) \leq^{\forall\exists} \mathrm{post}^{\mathcal{A}}_{va}(I) \Leftrightarrow &\quad \text{[by def. of } \leq^{r}_{\mathcal{A}}\text{]}\\
ua \leq^{r}_{\mathcal{A}} va\ .&
\end{aligned}
$$

Thus, $\leq^{r}_{\mathcal{A}}$ is a right $\mathcal{L}(\mathcal{A})$-consistent quasiorder.
Finally, since $\wp(Q)$ is finite, it follows that $\leq^{r}_{\mathcal{A}}$ is a well-quasiorder, and, since $\mathrm{post}^{\mathcal{A}}_{u}(I)$ is finite and computable for every $u \in \Sigma^*$, it follows that $\leq^{r}_{\mathcal{A}}$ is decidable. □

Thus, once again, Theorem 5.5 applies to $\leq^{r}_{\mathcal{A}_2}$ and this allows us to instantiate the algorithm `FAIncWr` to the quasiorder $\leq^{r}_{\mathcal{A}_2}$ for deciding an inclusion $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$.

Note that it is possible to define a left simulation $\leq^{\forall\exists}_{R}$ on an automaton $\mathcal{A}$ by applying $\leq^{\forall\exists}$ on the reverse automaton $\mathcal{A}^R$ of $\mathcal{A}$ where arrows are flipped and initial/final states are swapped. This left simulation induces a *left simulation-based quasiorder* on Σ^* as follows: for all $u, v \in \Sigma^*$,

$$
u \leq^{l}_{\mathcal{A}} v \stackrel{\triangle}{\iff} \mathrm{pre}^{\mathcal{A}}_{u}(F) \leq^{\forall\exists}_{R} \mathrm{pre}^{\mathcal{A}}_{v}(F)\ . \tag{19}
$$

It is straightforward to check that Theorem 5.3 applies to $\leq^{l}_{\mathcal{A}_2}$ and, therefore, we can instantiate the Algorithm `FAIncW` for deciding $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$.

Example 5.11. Let us illustrate the use of the left simulation-based quasiorder to solve the language inclusion $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$ of Example 5.7. For the set $F_{\mathcal{A}_2}$ of final states $\mathcal{A}_2$ we have the same sets of predecessors computed in Example 5.9 and, among others, the following left simulations between these sets w.r.t. the simulation quasiorder $\leq_{\mathcal{A}_2^R}$ of the reverse of $\mathcal{A}_2$ (recall that $\leq^{\forall\exists}$ is defined w.r.t. simulations of $\mathcal{A}_2^R$):

$$
\begin{array}{ll}
\mathrm{pre}^{\mathcal{A}_2}_{c}(F_{\mathcal{A}_2}) = \{q_2\} \leq^{\forall\exists}_{R} \{q_3\} = \mathrm{pre}^{\mathcal{A}_2}_{a}(F_{\mathcal{A}_2}) & \mathrm{pre}^{\mathcal{A}_2}_{c}(F_{\mathcal{A}_2}) = \{q_2\} \leq^{\forall\exists}_{R} \{q_4\} = \mathrm{pre}^{\mathcal{A}_2}_{b}(F_{\mathcal{A}_2})\\
\mathrm{pre}^{\mathcal{A}_2}_{c}(F_{\mathcal{A}_2}) = \{q_2\} \leq^{\forall\exists}_{R} \{q_1, q_3\} = \mathrm{pre}^{\mathcal{A}_2}_{aa}(F_{\mathcal{A}_2}) & \mathrm{pre}^{\mathcal{A}_2}_{c}(F_{\mathcal{A}_2}) = \{q_2\} \leq^{\forall\exists}_{R} \{q_1\} = \mathrm{pre}^{\mathcal{A}_2}_{ab}(F_{\mathcal{A}_2})\\
\mathrm{pre}^{\mathcal{A}_2}_{c}(F_{\mathcal{A}_2}) = \{q_2\} \leq^{\forall\exists}_{R} \{q_1, q_2\} = \mathrm{pre}^{\mathcal{A}_2}_{ac}(F_{\mathcal{A}_2}) &
\end{array}
$$

because $q_2 \leq_{\mathcal{A}_2^R} q_1$, $q_2 \leq_{\mathcal{A}_2^R} q_3$ and $q_2 \leq_{\mathcal{A}_2^R} q_4$ hold.
Let us show the computation of the Kleene iterates performed by Algorithm `FAIncW` when using the quasiorder $\sqsubseteq_{\leq^{l}_{\mathcal{A}_2}}$ as abstract inclusion check:

$$
\begin{aligned}
\vec{Y}^{(0)} &= \vec{\varnothing}\\
\vec{Y}^{(1)} &= \vec{\epsilon}^{F} = \langle \varnothing, \{\epsilon\}\rangle\\
\vec{Y}^{(2)} &= \vec{\epsilon}^{F} \cup \mathrm{Pre}_{\mathcal{A}_1}(\vec{Y}^{(1)}) = \langle \{a, b, c\}, \{\varepsilon\}\rangle\\
\vec{Y}^{(3)} &= \vec{\epsilon}^{F} \cup \mathrm{Pre}_{\mathcal{A}_1}(\vec{Y}^{(2)}) = \langle \{aa, ab, ac, a, b, c\}, \{\varepsilon\}\rangle
\end{aligned}
$$

It turns out that $\langle \{aa, ab, ac, a, b, c\}, \{\epsilon\}\rangle \sqsubseteq_{\leq^{l}_{\mathcal{A}_2}} \langle \{a, b, c\}, \{\epsilon\}\rangle$ holds, so that Kleene outputs the vector $\vec{Y} = \langle \{a, b, c\}, \{\epsilon\}\rangle$. Thus, once again, since $c \in \vec{Y}_0$ and $c \notin \mathcal{L}(\mathcal{A}_2)$, Algorithm `FAIncW` concludes that $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$ does not hold. ◇

Let us observe that $u \leq^{r}_{\mathcal{A}_2} v$ implies $W_{\mathrm{post}^{\mathcal{A}_2}_{u}(I),F} \subseteq W_{\mathrm{post}^{\mathcal{A}_2}_{v}(I),F}$, which is equivalent to the right Nerode quasiorder $u \leqq^{r}_{\mathcal{L}(\mathcal{A}_2)} v$ for $\mathcal{L}(\mathcal{A}_2)$ defined in (15), so that $u \leq^{r}_{\mathcal{A}_2} v \Rightarrow u \leqq^{r}_{\mathcal{L}(\mathcal{A}_2)} v$ holds. Furthermore, for the state-based quasiorder defined in (16), we have that $u \leq^{r}_{\mathcal{A}_2} v \Rightarrow u \leq^{r}_{\mathcal{A}_2} v$ trivially holds. Summing up, the following containments relate (the right versions of) state-based, simulation-based and Nerode quasiorders:

$$
\leq^{r}_{\mathcal{A}_2} \subseteq \leq^{r}_{\mathcal{A}_2} \subseteq \leqq^{r}_{\mathcal{L}(\mathcal{A}_2)}\ .
$$

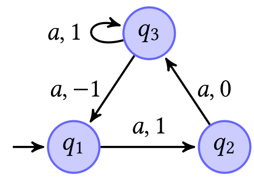


Fig. 3. A one-counter net O.

All these quasiorders are decidable $\mathcal{L}(\mathcal{A}_2)$-consistent wqos so that the algorithm FAIncW can be instantiated to each of them for deciding an inclusion $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$. Examples 5.7, 5.9 and 5.11 show how FAIncW behaves for each of the three quasiorders considered in this section. Despite their simplicity, the examples show the differences in the behavior of the algorithm when considering the different quasiorders. In particular, we observe that the iterations of Kleene for $\leqq^r_{\mathcal{L}(\mathcal{A}_2)}$ coincide with those for $\preceq^r_{\mathcal{A}_2}$ and, as expected, these Kleene iterates converge faster than those for $\leq^r_{\mathcal{A}_2}$. Recall that $\leqq^r_{\mathcal{L}(\mathcal{A}_2)}$ is the coarsest well-quasiorder for which Algorithm FAIncW works, hence its corresponding Kleene iterates exhibit optimal behavior in terms of number of iterations to converge. The drawback of using the Nerode quasiorder $\leqq^r_{\mathcal{L}(\mathcal{A}_2)}$ is that it requires checking language inclusion in order to decide whether two words are related, and this is a PSPACE-complete problem. Therefore, the coincidence of the Kleene iterates for $\leqq^r_{\mathcal{L}(\mathcal{A}_2)}$ and $\preceq^r_{\mathcal{A}_2}$ is of special interest since it highlights that Algorithm FAIncW might exhibit optimal behavior while using a "simpler" (i.e., finer) well-quasiorder such as $\preceq^r_{\mathcal{A}_2}$, which is a polynomial approximation of $\leqq^r_{\mathcal{L}(\mathcal{A}_2)}$.

5.4 Inclusion in Traces of One-Counter Nets.

We show that our framework can be instantiated to systematically derive an algorithm for deciding an inclusion $\mathcal{L}(\mathcal{A}) \subseteq L_2$ where L_2 is the trace set of a one-counter net (OCN). This is accomplished by defining a decidable L_2-consistent quasiorder so that Theorem 5.3 can be applied.

Intuitively, an OCN is a FA endowed with a nonnegative integer counter which can be incremented, decremented or left unchanged by a transition. Formally, a one-counter net [Hofman and Totzke 2018] is a tuple $O = \langle Q, \Sigma, \delta \rangle$ where Q is a finite set of states, Σ is an alphabet and $\delta \subseteq Q \times \Sigma \times \{-1, 0, 1\} \times Q$ is a set of transitions. A configuration of O is a pair qn consisting of a state $q \in Q$ and a value $n \in \mathbb{N}$ for the counter. Given two configurations $qn, q'n' \in Q \times \mathbb{N}$ we write $qn \xrightarrow{a} q'n'$ and call it a a-step (or simply step) if there exists a transition $(q, a, d, q') \in \delta$ such that $n' = n + d$. Given $qn \in Q \times \mathbb{N}$, the *trace set* $T(qn) \subseteq \Sigma^*$ of an OCN is defined as follows:

$$T(qn) \triangleq \{u \in \Sigma^* \mid Z^{qn}_u \neq \varnothing\} \quad \text{where}$$
$$Z^{qn}_u \triangleq \{q_k n_k \in Q \times \mathbb{N} \mid qn = q_0 n_0 \xrightarrow{a_1} q_1 n_1 \xrightarrow{a_2} \cdots \xrightarrow{a_k} q_k n_k,\ a_1 \cdots a_k = u\} \enspace .$$

Observe that $Z^{qn}_\epsilon = \{qn\}$ and Z^{qn}_u is a finite set for every word $u \in \Sigma^*$.

Let us consider the poset $\langle \mathbb{N}_\bot \triangleq \mathbb{N} \cup \{\bot\}, \leq_{\mathbb{N}_\bot} \rangle$ where $\bot \leq_{\mathbb{N}_\bot} n$ holds for all $n \in \mathbb{N}_\bot$, while for all $n, n' \in \mathbb{N}$, $n \leq_{\mathbb{N}_\bot} n'$ is the standard ordering relation between numbers. For a finite set of states $S \subseteq Q \times \mathbb{N}$, define the so-called macro state $M_S : Q \to \mathbb{N}_\bot$ as follows:

$$M_S(q) \triangleq \max\{n \in \mathbb{N} \mid qn \in S\}\,,$$

where $\max \varnothing \triangleq \bot$. Let us define the following quasiorder $\leq^r_{qn} \subseteq \Sigma^* \times \Sigma^*$:

$$u \leq^r_{qn} v \overset{\triangle}{\iff} \forall q \in Q,\ M_{Z^{qn}_u}(q) \leq_{\mathbb{N}_\bot} M_{Z^{qn}_v}(q) \enspace . \tag{20}$$

Example 5.12. Figure 3 depicts an OCN O over the singleton alphabet $\Sigma = \{a\}$. For O we have the following sets:

$$Z^{q_1 0}_{\epsilon} = \{q_1 0\} \qquad Z^{q_1 0}_{a} = \{q_2 1\} \qquad Z^{q_1 0}_{aa} = \{q_3 1\}$$
$$Z^{q_1 0}_{aaa} = \{q_3 2, q_1 0\} \qquad Z^{q_1 0}_{aaaa} = \{q_3 3, q_1 1, q_2 1\} \qquad Z^{q_1 0}_{b} = \varnothing$$

Hence, we have that:

$$M_{Z^{q_1 0}_{\epsilon}} = \begin{pmatrix} q_1 \mapsto 0 \\ q_2 \mapsto \bot \\ q_3 \mapsto \bot \end{pmatrix} \quad M_{Z^{q_1 0}_{a}} = \begin{pmatrix} q_1 \mapsto \bot \\ q_2 \mapsto 1 \\ q_3 \mapsto \bot \end{pmatrix} \quad M_{Z^{q_1 0}_{aa}} = \begin{pmatrix} q_1 \mapsto \bot \\ q_2 \mapsto \bot \\ q_3 \mapsto 1 \end{pmatrix} \quad M_{Z^{q_1 0}_{aaa}} = \begin{pmatrix} q_1 \mapsto 0 \\ q_2 \mapsto \bot \\ q_3 \mapsto 2 \end{pmatrix}$$

Therefore, the words ϵ, a and aa are pairwise incomparable for $\leq^r_{q_1 0}$, while we have that $aa \leq^r_{q_1 0} aaa$ and $\epsilon \leq^r_{q_1 0} aaa$. ◇

Lemma 5.13. *Let O be an OCN. For any configuration qn of O, $\leq^r_{qn}$ is a right $T(qn)$-consistent decidable wqo.*

Proof. It follows from Dickson's Lemma [Sakarovitch 2009, Section II.7.1.2] that $\leq^r_{qn}$ is a wqo. Since Z^{qn}_u and Z^{qn}_v are finite sets of configurations, the macro state functions $M_{Z^{qn}_u}$ and $M_{Z^{qn}_v}$ are computable, hence the relation $\leq^r_{qn}$ is decidable. If $u \in T(qn)$ and $v \notin T(qn)$ then $u \not\leq^r_{qn} v$, otherwise we would have that $M_{Z^{qn}_u}(q') \neq \bot$ for some $q' \in Q$, hence $M_{Z^{qn}_v}(q') \neq \bot$, and this would be a contradiction because $Z^{qn}_v = \varnothing$, so that $M_{Z^{qn}_v}(q') = \bot$.
Finally, let us show that $u \leq^r_{qn} v$ implies $ua \leq^r_{qn} va$ for all $a \in \Sigma$, since, by (14), this is equivalent to the fact that $\leq^r_{qn}$ is right monotonic. We proceed by contradiction. Assume that $u \leq^r_{qn} v$ and $\exists q' \in Q, M_{Z^{qn}_{ua}}(q') \not\leq_{\mathbb{N}_\bot} M_{Z^{qn}_{va}}(q')$. Then, $m_1 \triangleq \max\{n \mid pn \in Z^{qn}_{ua}\} \not\leq_{\mathbb{N}_\bot} m_2 \triangleq \max\{n \mid pn \in Z^{qn}_{va}\}$, which implies, since $m_1 \neq \bot$, that $m_1, m_2 \in \mathbb{N}$ and $m_1 > m_2$. Thus, for all $(q, a, d, q') \in \delta$ we have $q'(m_1 - d) \in Z^{qn}_u$ and $q'(m_2 - d) \in Z^{qn}_v$. Since $m_1 - d > m_2 - d$ we have that $\max\{n \mid pn \in Z^{qn}_u\} > \max\{n \mid pn \in Z^{qn}_v\}$, which contradicts $u \leq^r_{qn} v$. □

By Theorem 5.3, Lemma 5.13 and the decidability of membership $u \in^? T(qn)$, the following known decidability result for inclusion of regular languages into traces of OCNs [Jančar et al. 1999, Theorem 3.2] is systematically derived as a consequence of our algorithmic framework.

Corollary 5.14. *Let $\mathcal{A}$ be a FA and O be an OCN. For any configuration qn of O, the language inclusion problem $\mathcal{L}(\mathcal{A}) \subseteq T(qn)$ is decidable.*

Example 5.15. Consider the OCN of Figure 3 and the problem of deciding whether $\Sigma^* = a^*$ is included into $T(q_1 0)$, i.e., whether the trace set of O is universal. By considering the equation $X = Xa \cup \{\epsilon\}$ which defines Σ^*, it turns out that the Kleene iterates computed by Algorithm `FAIncW` when using the abstract inclusion check given by $\sqsubseteq_{\leq^r_{q_1 0}}$ are as follows:

$$Y^{(0)} = \varnothing \qquad Y^{(1)} = \{\epsilon\} \qquad Y^{(2)} = \{a, \epsilon\} \qquad Y^{(3)} = \{aa, a, \epsilon\} \qquad Y^{(4)} = \{aaa, aa, a, \epsilon\} \ .$$

We have that $Y^{(4)} \sqsubseteq_{\leq^r_{q_1 0}} Y^{(3)}$ because $aa \leq^r_{q_1 0} aaa$ holds, as shown in Example 5.12, so that the output of Kleene is $Y^{(3)} = \{aa, a, \epsilon\}$. Since $\{aa, a, \epsilon\}$ is a set of traces of O (i.e. $\{aa, a, \epsilon\} \subseteq T(q_1 0)$) we conclude that O is universal. ◇

Moreover, by exploiting Lemma 5.13 and [Hofman et al. 2013, Theorem 20], the following result settles a conjecture made by de Luca and Varricchio [1994, Section 6] on the right Nerode quasiorder for traces of OCNs.

Lemma 5.16. *The right Nerode quasiorder $\leqq^r_{T(qn)}$ is an undecidable well-quasiorder.*

Proof. As already recalled, de Luca and Varricchio [1994, Section 2, point 4] show that $\leqq^r_{T(qn)}$ is maximum in the set of all right $T(qn)$-consistent quasiorders, so that $u \leq^r_{qn} v \Rightarrow u \leqq^r_{T(qn)} v$, for all $u, v \in \Sigma^*$. By Lemma 5.13, $\leq^r_{qn}$ is a wqo, so that $\leqq^r_{T(qn)}$ is a wqo as well. Undecidability of $\leqq^r_{T(qn)}$ follows from the undecidability of the trace inclusion problem for nondeterministic OCNs [Hofman et al. 2013, Theorem 20] by an argument similar to the automata case. □

It is worth remarking that, by Lemma 5.6 (a), the left and right Nerode quasiorders $\leqq^l_{T(qn)}$ and $\leqq^r_{T(qn)}$ are $T(qn)$-consistent. However, the left Nerode quasiorder does not need to be a wqo, otherwise $T(qn)$ would be regular.

We conclude this section by conjecturing that our framework could be instantiated for extending Corollary 5.14 to traces of Petri Nets, a result which is already known to be true [Jančar et al. 1999].

6 A NOVEL PERSPECTIVE ON THE ANTICHAIN ALGORITHM

In this section, we show how to solve the language inclusion problem by computing Kleene iterates in an abstract domain of $\wp(\Sigma^*)$ as defined by a Galois connection. This is of practical interest since it allows us to decide a language inclusion problem $\mathcal{L}(\mathcal{A}) \subseteq L_2$ by manipulating an automaton representation for L_2.

6.1 A Language Inclusion Algorithm Using Galois Connections

The next result provides a formulation of Theorem 4.7 by using a Galois Connection $\langle \wp(\Sigma^*), \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle D, \leq_D\rangle$ rather than a closure operator $\rho \in \mathrm{uco}(\wp(\Sigma^*))$ and shows how to design an algorithm that solves a language inclusion $\mathcal{L}(\mathcal{A}) \subseteq L_2$ by computing the Kleene iterates on the abstract domain D.

Theorem 6.1. *Let $\mathcal{A} = \langle Q, \delta, I, F, \Sigma\rangle$ be a FA and $L_2 \in \wp(\Sigma^*)$. Let $\langle D, \leq_D\rangle$ be a poset and $\langle \wp(\Sigma^*), \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle D, \leq_D\rangle$ be a GC. Assume that the following properties hold:*

(i) *$L_2 \in \gamma(D)$ and for all $a \in \Sigma$ and $X \in \wp(\Sigma^*)$, $\gamma\alpha(aX) = \gamma\alpha(a\gamma\alpha(X))$.*
(ii) *$\langle D, \leq_D, \sqcup, \bot_D\rangle$ 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.*
(iii) *There is an algorithm, say $\mathrm{Pre}^\sharp$, which computes $\alpha \circ \mathrm{Pre}_{\mathcal{A}} \circ \gamma$.*
(iv) *There is an algorithm, say $\epsilon^\sharp$, which computes $\alpha(\vec{\boldsymbol{\epsilon}}^F)$.*
(v) *There is an algorithm, say $\mathrm{Incl}^\sharp$, which decides $\vec{\boldsymbol{X}}^\sharp \leq_D \alpha(\vec{\boldsymbol{L_2^I}})$, for all $\vec{\boldsymbol{X}}^\sharp \in \alpha(\wp(\Sigma^*))^{|Q|}$.*

Then,

$\langle Y^\sharp_q\rangle_{q\in Q} := \textsc{Kleene}(\leq_D, \lambda\vec{\boldsymbol{X}}^\sharp.\, \epsilon^\sharp \sqcup \mathrm{Pre}^\sharp(\vec{\boldsymbol{X}}^\sharp), \vec{\boldsymbol{\bot_D}})$;
return $\mathrm{Incl}^\sharp(\langle Y^\sharp_q\rangle_{q\in Q})$;

is a decision algorithm for $\mathcal{L}(\mathcal{A}) \subseteq L_2$.

Proof. Let $\rho \triangleq \gamma\alpha \in \mathrm{uco}(\wp(\Sigma^*))$, so that hypothesis (i) can be stated as $\rho(L_2) = L_2$ and $\rho(aX) = \rho(a\rho(X))$, and this allows us to apply Corollary 4.4. It turns out that:

$$
\begin{aligned}
\mathcal{L}(\mathcal{A}) \subseteq L_2 \Leftrightarrow &\quad \text{[by Corollary 4.4 and (11)]}\\
\mathrm{lfp}(\lambda\vec{\boldsymbol{X}}.\, \gamma\alpha(\vec{\boldsymbol{\epsilon}}^F \cup \mathrm{Pre}_{\mathcal{A}}(\vec{\boldsymbol{X}}))) \subseteq \vec{\boldsymbol{L_2^I}} \Leftrightarrow &\quad \text{[by Lemma 2.1]}\\
\gamma(\mathrm{lfp}(\lambda\vec{\boldsymbol{X}}^\sharp.\, \alpha(\vec{\boldsymbol{\epsilon}}^F \cup \mathrm{Pre}_{\mathcal{A}}(\gamma(\vec{\boldsymbol{X}}^\sharp))))) \subseteq \vec{\boldsymbol{L_2^I}} \Leftrightarrow &\quad \text{[by GC]}\\
\gamma(\mathrm{lfp}(\lambda\vec{\boldsymbol{X}}^\sharp.\, \alpha(\vec{\boldsymbol{\epsilon}}^F) \sqcup \alpha(\mathrm{Pre}_{\mathcal{A}}(\gamma(\vec{\boldsymbol{X}}^\sharp))))) \subseteq \vec{\boldsymbol{L_2^I}} \Leftrightarrow &\quad \text{[by GC and since, by (i), } L_2 \in \gamma(D)]\\
\mathrm{lfp}(\lambda\vec{\boldsymbol{X}}^\sharp.\, \alpha(\vec{\boldsymbol{\epsilon}}^F) \sqcup \alpha(\mathrm{Pre}_{\mathcal{A}}(\gamma(\vec{\boldsymbol{X}}^\sharp)))) \leq_D \alpha(\vec{\boldsymbol{L_2^I}}) &\quad .
\end{aligned}
$$

Thus, by hypotheses (ii), (iii) and (iv), it turns out that $\textsc{Kleene}(\leq_D, \lambda\vec{X}^\sharp.\, \epsilon^\sharp \sqcup \mathrm{Pre}^\sharp(\vec{X}^\sharp), \overrightarrow{\bot_D})$ is an algorithm computing the least fixpoint $\mathrm{lfp}(\lambda\vec{X}^\sharp.\, \alpha(\vec{\epsilon}^F) \sqcup \alpha(\mathrm{Pre}_{\mathcal{A}}(\gamma(\vec{X}^\sharp))))$. In particular, (ii), (iii) and (iv) ensure that the Kleene iterates of $\lambda\vec{X}^\sharp.\, \epsilon^\sharp \sqcup \mathrm{Pre}^\sharp(\vec{X}^\sharp)$ starting from $\overrightarrow{\bot_D}$ are computable and finitely many and that it is decidable when the iterates converge for $\leq_D$, namely, reach the least fixpoint. Finally, hypothesis (v) ensures the decidability of the $\leq_D$-inclusion check of this least fixpoint in $\alpha(\vec{L_2^I})$. □

It is worth pointing out that, analogously to Theorem 4.6, the above Theorem 6.1 can be also stated in a symmetric version for right (rather than left) concatenation.

6.2 Antichains as a Galois Connection

Let $\mathcal{A}_1 = \langle Q_1, \delta_1, I_1, F_1, \Sigma\rangle$ and $\mathcal{A}_2 = \langle Q_2, \delta_2, I_2, F_2, \Sigma\rangle$ be two FAs and consider the state-based left $\mathcal{L}(\mathcal{A}_2)$-consistent wqo $\leqslant^l_{\mathcal{A}_2}$ defined by (16). Theorem 5.3 shows that Algorithm `FAIncW` decides $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$ by computing vectors of finite sets of words. Since $u \leqslant^l_{\mathcal{A}_2} v \Leftrightarrow \mathrm{pre}^{\mathcal{A}_2}_u(F_2) \subseteq \mathrm{pre}^{\mathcal{A}_2}_v(F_2)$, we can equivalently consider the set of states $\mathrm{pre}^{\mathcal{A}_2}_u(F_2) \in \wp(Q_2)$ rather than a word $u \in \Sigma^*$. This observation suggests to design a version of Algorithm `FAIncW` that computes Kleene iterates on the poset $\langle \mathrm{AC}_{\langle\wp(Q_2),\subseteq\rangle}, \sqsubseteq\rangle$ of antichains of sets of states of the complete lattice $\langle\wp(Q_2),\subseteq\rangle$. To achieve this, $\langle \mathrm{AC}_{\langle\wp(Q_2),\subseteq\rangle}, \sqsubseteq\rangle$ is viewed as an abstract domain through the following maps $\alpha\colon \wp(\Sigma^*) \to \mathrm{AC}_{\langle\wp(Q_2),\subseteq\rangle}$ and $\gamma\colon \mathrm{AC}_{\langle\wp(Q_2),\subseteq\rangle} \to \wp(\Sigma^*)$. Moreover, we use the abstract function $\mathrm{Pre}^{\mathcal{A}_2}_{\mathcal{A}_1} : (\mathrm{AC}_{\langle\wp(Q_2),\subseteq\rangle})^{|Q_1|} \to (\mathrm{AC}_{\langle\wp(Q_2),\subseteq\rangle})^{|Q_1|}$ defined as follows:

$$
\begin{aligned}
&\alpha(X) \triangleq \lfloor\{\mathrm{pre}^{\mathcal{A}_2}_u(F_2) \in \wp(Q_2) \mid u \in X\}\rfloor\,,\\
&\gamma(Y) \triangleq \{v \in \Sigma^* \mid \exists y \in Y,\ y \subseteq \mathrm{pre}^{\mathcal{A}_2}_v(F_2)\}\,,\\
&\mathrm{Pre}^{\mathcal{A}_2}_{\mathcal{A}_1}(\langle X_q\rangle_{q\in Q_1}) \triangleq \langle\lfloor\{\mathrm{pre}^{\mathcal{A}_2}_a(S) \in \wp(Q_2) \mid \exists a \in \Sigma, q' \in Q_1, q' \in \delta_1(q,a) \wedge S \in X_{q'}\}\rfloor\rangle_{q\in Q_1},
\end{aligned}
\tag{21}
$$

where $\lfloor X\rfloor$ is the unique minor set w.r.t. subset inclusion of $X \subseteq \wp(Q_2)$. Observe that the functions α and $\mathrm{Pre}^{\mathcal{A}_2}_{\mathcal{A}_1}$ are well-defined because minors of finite subsets of $\wp(Q_2)$ are uniquely defined antichains.

Lemma 6.2. *The following properties hold:*

(a) $\langle\wp(\Sigma^*),\subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathrm{AC}_{\langle\wp(Q_2),\subseteq\rangle}, \sqsubseteq\rangle$ *is a GC;*
(b) $\gamma \circ \alpha = \rho_{\leqslant^l_{\mathcal{A}_2}}$;
(c) $\mathrm{Pre}^{\mathcal{A}_2}_{\mathcal{A}_1} = \alpha \circ \mathrm{Pre}_{\mathcal{A}_1} \circ \gamma$.

Proof.

(a) Let us first observe that α is well-defined: $\alpha(X)$ is an antichain of $\langle\wp(Q_2),\subseteq\rangle$ since it is a minor for the well-quasiorder $\subseteq$ and, therefore, it is finite. Then, for all $X \in \wp(\Sigma^*)$ and $Y \in \mathrm{AC}_{\langle\wp(Q_2),\subseteq\rangle}$, it turns out that:

$$
\begin{aligned}
\alpha(X) \sqsubseteq Y \Leftrightarrow\quad & [\text{by definition of } \sqsubseteq]\\
\forall z \in \alpha(X), \exists y \in Y,\ y \subseteq z \Leftrightarrow\quad & [\text{by definition of } \alpha]\\
\forall z \in \lfloor\{\mathrm{pre}^{\mathcal{A}_2}_u(F_2) \in \wp(Q_2) \mid u \in X\}\rfloor, \exists y \in Y,\ y \subseteq z \Leftrightarrow\quad & [\text{by definition of } \lfloor\cdot\rfloor]\\
\forall u \in X, \exists y \in Y,\ y \subseteq \mathrm{pre}^{\mathcal{A}_2}_u(F_2) \Leftrightarrow\quad & [\text{by definition of } \gamma]\\
X \subseteq \gamma(Y)\ .\quad &
\end{aligned}
$$

(b) For all $X \in \wp(\Sigma^*)$:

$$\gamma(\alpha(X)) = \quad [\text{by definition of } \alpha, \gamma]$$

$$
\begin{aligned}
\{v \in \Sigma^* \mid \exists u \in \Sigma^*, \mathrm{pre}_u^{\mathcal{A}_2}(F_2) \in \lfloor \{\mathrm{pre}_w^{\mathcal{A}_2}(F_2) \mid w \in X\} \rfloor \wedge \mathrm{pre}_u^{\mathcal{A}_2}(F_2) \subseteq \mathrm{pre}_v^{\mathcal{A}_2}(F_2)\} & \\
= & \quad \text{[by definition of minor]} \\
\{v \in \Sigma^* \mid \exists u \in X,\ \mathrm{pre}_u^{\mathcal{A}_2}(F_2) \subseteq \mathrm{pre}_v^{\mathcal{A}_2}(F_2)\} = & \quad \text{[by definition of } \leqslant^l_{\mathcal{A}_2}] \\
\{v \in \Sigma^* \mid \exists u \in X,\ u \leqslant^l_{\mathcal{A}_2} v\} = & \quad \text{[by definition of } \rho_{\leqslant^l_{\mathcal{A}_2}}] \\
\rho_{\leqslant^l_{\mathcal{A}_2}}(X) \ . &
\end{aligned}
$$

(c) For all $\vec{X} \in (\mathrm{AC}_{\langle \wp(Q_2), \subseteq \rangle})^{|Q_1|}$:

$$
\begin{aligned}
& \alpha(\mathrm{Pre}_{\mathcal{A}_1}(\gamma(\vec{X}))) = \\
& \qquad \text{[by definition of } \mathrm{Pre}_{\mathcal{A}_1}] \\
& \langle \alpha(\textstyle\bigcup_{a \in \Sigma, q \xrightarrow{a}_{\mathcal{A}_1} q'} a\gamma(\vec{X}_{q'})) \rangle_{q \in Q_1} = \\
& \qquad \text{[by definition of } \alpha] \\
& \langle \lfloor \{\mathrm{pre}_u^{\mathcal{A}_2}(F_2) \mid u \in \textstyle\bigcup_{a \in \Sigma, q \xrightarrow{a}_{\mathcal{A}_1} q'} a\gamma(\vec{X}_{q'}) \rfloor \rangle_{q \in Q_1} = \\
& \qquad \text{[by } \mathrm{pre}_{av}^{\mathcal{A}_2} = \mathrm{pre}_a^{\mathcal{A}_2} \circ \mathrm{pre}_v^{\mathcal{A}_2}] \\
& \langle \lfloor \{\mathrm{pre}_a^{\mathcal{A}_2}(\{\mathrm{pre}_v^{\mathcal{A}_2}(F_2) \mid v \in \textstyle\bigcup_{q \xrightarrow{a}_{\mathcal{A}_1} q'} \gamma(\vec{X}_{q'})\}) \mid a \in \Sigma\} \rfloor \rangle_{q \in Q_1} = \\
& \qquad \text{[by rewriting]} \\
& \langle \lfloor \{\mathrm{pre}_a^{\mathcal{A}_2}(S) \mid a \in \Sigma, q \xrightarrow{a}_{\mathcal{A}_1} q', S \in \{\mathrm{pre}_v^{\mathcal{A}_2}(F_2) \mid v \in \gamma(\vec{X}_{q'})\}\} \rfloor \rangle_{q \in Q_1} = \\
& \qquad \text{[by } \lfloor \{\mathrm{pre}_a^{\mathcal{A}_2}(S) \mid S \in Y\} \rfloor = \lfloor \{\mathrm{pre}_a^{\mathcal{A}_2}(S) \mid S \in \lfloor Y \rfloor\} \rfloor] \\
& \langle \lfloor \{\mathrm{pre}_a^{\mathcal{A}_2}(S) \mid a \in \Sigma, q \xrightarrow{a}_{\mathcal{A}_1} q', S \in \lfloor \{\mathrm{pre}_v^{\mathcal{A}_2}(F_2) \mid v \in \gamma(\vec{X}_{q'})\} \rfloor\} \rfloor \rangle_{q \in Q_1} = \\
& \qquad \text{[by definition of } \alpha] \\
& \langle \lfloor \{\mathrm{pre}_a^{\mathcal{A}_2}(S) \mid a \in \Sigma, q \xrightarrow{a}_{\mathcal{A}_1} q', S \in \alpha(\gamma(\vec{X}_{q'})) \rfloor \rangle_{q \in Q_1} = \\
& \qquad \text{[since } \vec{X} \in \alpha,\ \alpha(\gamma(\vec{X}_{q'})) = \vec{X}_{q'}] \\
& \langle \lfloor \{\mathrm{pre}_a^{\mathcal{A}_2}(S) \mid a \in \Sigma, q \xrightarrow{a}_{\mathcal{A}_1} q', S \in \vec{X}_{q'}\} \rfloor \rangle_{q \in Q_1} = \\
& \qquad \text{[by definition of } \mathrm{Pre}_{\mathcal{A}_1}^{\mathcal{A}_2}] \\
& \mathrm{Pre}_{\mathcal{A}_1}^{\mathcal{A}_2}(\vec{X}) \ .
\end{aligned}
$$

□

Thus, by Lemmata 5.8 and 6.2, it turns out that the GC $\langle \wp(\Sigma^*), \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathrm{AC}_{\langle \wp(Q_2), \subseteq \rangle}, \sqsubseteq \rangle$ and the abstract function $\mathrm{Pre}_{\mathcal{A}_1}^{\mathcal{A}_2}$ satisfy the hypotheses (i)-(iv) of Theorem 6.1. To obtain an algorithm for deciding $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$, it remains to show that the hypothesis (v) of Theorem 6.1 holds, i.e., there is an algorithm to decide whether $\vec{Y} \sqsubseteq \alpha(\overrightarrow{L_2^{I_2}})$ for every $\vec{Y} \in \alpha(\wp(\Sigma^*))^{|Q_1|}$.

Notice that the Kleene iterates of $\lambda \vec{X}^\sharp . \alpha(\vec{\epsilon}^{F_1}) \sqcup \mathrm{Pre}_{\mathcal{A}_1}^{\mathcal{A}_2}(\vec{X}^\sharp)$ of Theorem 6.1 are vectors of antichains in $\langle \mathrm{AC}_{\langle \wp(Q_2), \subseteq \rangle}, \sqsubseteq \rangle$, where each component is indexed by some $q \in Q_1$ and represents, through its minor, a set of sets of states that are predecessors of F_2 in $\mathcal{A}_2$ through a word u generated by $\mathcal{A}_1$ from that state q, i.e., $\mathrm{pre}_u^{\mathcal{A}_2}(F_2)$ with $u \in W_{q,F_1}^{\mathcal{A}_1}$. Since $\epsilon \in W_{q,F_1}^{\mathcal{A}_1}$ for all $q \in F_1$ and $\mathrm{pre}_\epsilon^{\mathcal{A}_2}(F_2) = F_2$, the first iteration of Kleene gives the vector $\alpha(\vec{\epsilon}^{F_1}) = \langle \psi_{\varnothing}^{F_2}(q \overset{?}{\in} F_1) \rangle_{q \in Q_1}$. Let us also observe that by taking the minor of each vector component, we are considering smaller sets which still preserve the relation $\sqsubseteq$ since the following equivalences hold:

$$A \sqsubseteq B \Leftrightarrow \lfloor A \rfloor \sqsubseteq B \Leftrightarrow A \sqsubseteq \lfloor B \rfloor \Leftrightarrow \lfloor A \rfloor \sqsubseteq \lfloor B \rfloor .$$

Let $\langle Y_q\rangle_{q\in Q_1}$ be the output of $\textsc{Kleene}(\sqsubseteq, \lambda\vec{X}^\sharp.\, \alpha(\vec{\epsilon}^{F_1}) \sqcup \mathrm{Pre}^{\mathcal{A}_2}_{\mathcal{A}_1}(\vec{X}^\sharp), \vec{\varnothing})$. Hence, we have that, for each component $q \in Q_1$, $Y_q = \lfloor\{\mathrm{pre}^{\mathcal{A}_2}_u(F_2) \mid u \in W^{\mathcal{A}_1}_{q,F_1}\}\rfloor$ holds. Whenever the inclusion $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$ holds, all the sets of states in Y_q, for some initial state $q \in I_1$, are predecessors of F_2 in $\mathcal{A}_2$ through words in $\mathcal{L}(\mathcal{A}_2)$, so that for each $q \in I_1$ and $S \in Y_q$, $S \cap I_2 \neq \varnothing$ must hold. As a result, the following state-based algorithm `FAIncS` (`S` stands for state) decides the inclusion $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$ by computing on the abstract domain of antichains $\langle \mathrm{AC}_{\langle\wp(Q_2),\subseteq\rangle}, \sqsubseteq\rangle$.

`FAIncS`: State-based algorithm for $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$

Data: FAs $\mathcal{A}_1 = \langle Q_1, \delta_1, I_1, F_1, \Sigma\rangle$ and $\mathcal{A}_2 = \langle Q_2, \delta_2, I_2, F_2, \Sigma\rangle$.

1 $\langle Y_q\rangle_{q\in Q_1} := \textsc{Kleene}(\sqsubseteq, \lambda\vec{X}^\sharp.\, \alpha(\vec{\epsilon}^{F_1}) \sqcup \mathrm{Pre}^{\mathcal{A}_2}_{\mathcal{A}_1}(\vec{X}^\sharp), \vec{\varnothing})$;
2 **forall** $q \in I_1$ **do**
3   **forall** $S \in Y_q$ **do**
4   **if** $S \cap I_2 = \varnothing$ **then return** *false*;
5 **return** *true*;

Theorem 6.3. *The algorithm* `FAIncS` *decides the inclusion problem* $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$.

Proof. We show that all the hypotheses (i)-(v) of Theorem 6.1 are satisfied for the abstract domain $\langle D, \leq_D\rangle = \langle \mathrm{AC}_{\langle\wp(Q_2),\subseteq\rangle}, \sqsubseteq\rangle$ as defined by the GC of Lemma 6.2.

(i) Since $\rho_{\leq^l_{\mathcal{A}_2}}(X) = \gamma(\alpha(X))$, it follows from Lemmata 5.2 and 5.8 that $L_2 \in \gamma(D)$. Moreover, by Lemma 5.2 (b) with $\rho_{\leq^l_{\mathcal{A}_2}} = \gamma\alpha$, we have that for all $a \in \Sigma$, $X \in \wp(\Sigma^*)$, $\gamma(\alpha(aX)) = \gamma(\alpha(a\gamma(\alpha(X))))$.

(ii) $\langle \mathrm{AC}_{\langle\wp(Q_2),\subseteq\rangle}, \sqsubseteq, \sqcup, \varnothing\rangle$ is an effective domain because Q_2 is finite.

(iii) By Lemma 6.2 (c), we have that $\alpha(\mathrm{Pre}_{\mathcal{A}_1}(\gamma(\vec{X}^\sharp))) = \mathrm{Pre}^{\mathcal{A}_2}_{\mathcal{A}_1}(\vec{X}^\sharp)$, for all $\vec{X}^\sharp \in \alpha(\wp(\Sigma^*))^{|Q_1|}$, and $\mathrm{Pre}^{\mathcal{A}_2}_{\mathcal{A}_1}$ is computable.

(iv) $\alpha(\{\epsilon\}) = \{F_2\}$ and $\alpha(\varnothing) = \varnothing$, hence $\alpha(\vec{\epsilon}^{F_1})$ is trivial to compute.

(v) Since $\alpha(\overrightarrow{L_2}^{I_1}) = \langle \alpha(\psi^{L_2}_{\Sigma^*}(q \overset{?}{\in} I_1))\rangle_{q\in Q_1}$, for all $\vec{Y} \in \alpha(\wp(\Sigma^*))^{|Q_1|}$, the relation $\langle Y_q\rangle_{q\in Q_1} \sqsubseteq \alpha(\overrightarrow{L_2}^{I_1})$ trivially holds for all components $q \notin I_1$, since $\alpha(\Sigma^*)$ is the greatest antichain. For the components $q \in I_1$, it suffices to show that $Y_q \sqsubseteq \alpha(L_2) \Leftrightarrow \forall S \in Y_q,\ S \cap I_2 \neq \varnothing$, which is the check performed by lines 2-5 of algorithm `FAIncS`:

$$\begin{aligned}
Y_q \sqsubseteq \alpha(L_2) \Leftrightarrow\quad & [\text{because } Y_q = \alpha(U) \text{ for some } U \in \wp(\Sigma^*)]\\
\alpha(U) \sqsubseteq \alpha(L_2) \Leftrightarrow\quad & [\text{by GC}]\\
U \subseteq \gamma(\alpha(L_2)) \Leftrightarrow\quad & [\text{by (i), } \gamma(\alpha(L_2)) = L_2]\\
U \subseteq L_2 \Leftrightarrow\quad & [\text{by definition of } \mathrm{pre}^{\mathcal{A}_2}_u]\\
\forall u \in U, \mathrm{pre}^{\mathcal{A}_2}_u(F_2) \cap I_2 \neq \varnothing \Leftrightarrow\quad & [\text{because } Y_q = \alpha(U) = \lfloor\{\mathrm{pre}^{\mathcal{A}_2}_u(F_2) \mid u \in U\}\rfloor]\\
\forall S \in Y_q, S \cap I_2 \neq \varnothing\ \ . &
\end{aligned}$$

Thus, by Theorem 6.1, the algorithm `FAIncS` solves the inclusion problem $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$. □

6.3 Relationship to the Antichain Algorithm

De Wulf et al. [2006] introduced two so-called antichain algorithms, called forward and backward, for deciding the universality of the language accepted by a FA, i.e., whether the language is Σ^* or not. Then, they extended the backward algorithm in order to decide inclusion of languages accepted by FAs. In what follows, we show that our algorithm FAIncS is equivalent to the corresponding extension of the forward antichain algorithm and, therefore, dual to the backward antichain algorithm for language inclusion put forward by De Wulf et al. [2006, Theorem 6]. To achieve this, we first define the poset of antichains in which the forward antichain algorithm computes its fixpoint. Then, we give a formal definition of the forward antichain algorithm for deciding language inclusion and show that this algorithm coincides with FAIncS when applied to the reverse automata. Since language inclusion between the languages generated by two FAs holds iff inclusion holds between the languages generated by their reverse FAs, this entails that our algorithm FAIncS is equivalent to the forward antichain algorithm.

Consider a language inclusion problem $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$ with $\mathcal{A}_1 = \langle Q_1, \delta_1, I_1, F_1, \Sigma\rangle$ and $\mathcal{A}_2 = \langle Q_2, \delta_2, I_2, F_2, \Sigma\rangle$. Let us consider the following poset of antichains $\langle \mathrm{AC}_{\langle\wp(Q_2),\subseteq\rangle}, \widetilde{\sqsubseteq}\rangle$ where

$$X \mathrel{\widetilde{\sqsubseteq}} Y \overset{\triangle}{\iff} \forall y \in Y, \exists x \in X,\ x \subseteq y$$

and notice that $\widetilde{\sqsubseteq}$ coincides with the reverse $\sqsubseteq^{-1}$ of the relation defined by (1). As observed by De Wulf et al. [2006, Lemma 1], it turns out that $\langle \mathrm{AC}_{\langle\wp(Q_2),\subseteq\rangle}, \widetilde{\sqsubseteq}, \widetilde{\sqcup}, \widetilde{\sqcap}, \{\varnothing\}, \varnothing\rangle$ is a finite lattice, where $\widetilde{\sqcup}$ and $\widetilde{\sqcap}$ denote, resp., lub and glb, and $\{\varnothing\}$ and $\varnothing$ are, resp., the least and greatest elements. This lattice $\langle \mathrm{AC}_{\langle\wp(Q_2),\subseteq\rangle}, \widetilde{\sqsubseteq}\rangle$ is the domain in which the forward antichain algorithm computes on for deciding language universality [De Wulf et al. 2006, Theorem 3]. The following result extends this forward algorithm in order to decide language inclusion.

Theorem 6.4 (**[De Wulf et al. 2006, Theorems 3 and 6]**). *Let*

$$\overrightarrow{\mathcal{FP}} \triangleq \widetilde{\bigsqcup}\{\vec{X} \in (\mathrm{AC}_{\langle\wp(Q_2),\subseteq\rangle})^{|Q_1|} \mid \vec{X} = \mathrm{Post}_{\mathcal{A}_1}^{\mathcal{A}_2}(\vec{X}) \mathrel{\widetilde{\sqcap}} \langle \psi_{\varnothing}^{\{I_2\}}(q \in^{?} I_1)\rangle_{q\in Q_1}\}$$

where $\mathrm{Post}_{\mathcal{A}_1}^{\mathcal{A}_2}(\langle X_q\rangle_{q\in Q_1}) \triangleq \langle \lfloor\{\mathrm{post}_a^{\mathcal{A}_2}(S) \in \wp(Q_2) \mid \exists a \in \Sigma, q' \in Q_1,\ q \in \delta_1(q', a) \wedge S \in X_{q'}\}\rfloor\rangle_{q\in Q_1}$. *Then,* $\mathcal{L}(\mathcal{A}_1) \nsubseteq \mathcal{L}(\mathcal{A}_2)$ *if and only if there exists* $q \in F_1$ *such that* $\overrightarrow{\mathcal{FP}}_q \mathrel{\widetilde{\sqsubseteq}} \{F_2^c\}$.

Proof. Let us first introduce some notation to describe the forward antichain algorithm by De Wulf et. al [2006] which decides $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$. Let us consider the poset $\langle Q_1 \times \wp(Q_2), \subseteq_\times\rangle$ where $(q_1, S_1) \subseteq_\times (q_2, S_2) \overset{\triangle}{\Leftrightarrow} q_1 = q_2 \wedge S_1 \subseteq S_2$. Then, let $\langle \mathrm{AC}_{\langle Q_1\times\wp(Q_2),\subseteq_\times\rangle}, \widetilde{\sqsubseteq}_\times, \widetilde{\sqcup}_\times, \widetilde{\sqcap}_\times\rangle$ be the lattice of antichains over $\langle Q_1 \times \wp(Q_2), \subseteq_\times\rangle$ where:

$$\begin{aligned}
&X \mathrel{\widetilde{\sqsubseteq}_\times} Y \overset{\triangle}{\Leftrightarrow} \forall (q,T) \in Y, \exists (q,S) \in X, S \subseteq T\\
&X \mathrel{\widetilde{\sqcup}_\times} Y \triangleq \min_\times(\{(q, S \cup T) \mid (q,S) \in X,\ (q,T) \in Y\})\\
&X \mathrel{\widetilde{\sqcap}_\times} Y \triangleq \min_\times(\{(q,S) \mid (q,S) \in X \cup Y\})\\
\text{with}\quad &\min_\times(X) \triangleq \{(q,S) \in X \mid \forall (q',S') \in X, q = q' \Rightarrow S' \not\subset S\}\ .
\end{aligned}$$

Also, let $\mathrm{Post} : \mathrm{AC}_{\langle Q_1\times\wp(Q_2),\subseteq_\times\rangle} \to \mathrm{AC}_{\langle Q_1\times\wp(Q_2),\subseteq_\times\rangle}$ be defined as follows:

$$\mathrm{Post}(X) \triangleq \min_\times(\{(q, \mathrm{post}_a^{\mathcal{A}_2}(S)) \in Q_1 \times \wp(Q_2) \mid \exists a \in \Sigma, q \in Q_1, (q', S) \in X, q' \xrightarrow{a}_{\mathcal{A}_1} q\})\ .$$

Then, the dual of the backward antichain algorithm in [De Wulf et al. 2006, Theorem 6] states that $\mathcal{L}(\mathcal{A}_1) \nsubseteq \mathcal{L}(\mathcal{A}_2)$ iff there exists $q \in F_1$ such that $\mathcal{FP} \mathrel{\widetilde{\sqsubseteq}_\times} \{(q, F_2^c)\}$ where

$$\mathcal{FP} = \widetilde{\bigsqcup}_\times\{X \in \mathrm{AC}_{\langle Q_1\times\wp(Q_2),\subseteq_\times\rangle} \mid X = \mathrm{Post}(X) \mathrel{\widetilde{\sqcap}_\times} (I_1 \times \{I_2\})\}\ .$$

We observe that for some $X \in \mathrm{AC}_{\langle Q_1\times\wp(Q_2),\subseteq_\times\rangle}$, a pair $(q,S) \in Q_1 \times \wp(Q_2)$ such that $(q,S) \in X$ is used by [De Wulf et al. 2006, Theorem 6] simply as a way to associate states q of $\mathcal{A}_1$ with sets S of states of $\mathcal{A}_2$. In fact, an antichain $X \in \mathrm{AC}_{\langle Q_1\times\wp(Q_2),\subseteq_\times\rangle}$ can be equivalently formalized by a vector $\langle\{S \in \wp(Q_2) \mid (q,S) \in X\}\rangle_{q\in Q_1} \in (\mathrm{AC}_{\langle\wp(Q_2),\subseteq\rangle})^{|Q_1|}$ whose components are indexed by states $q \in Q_1$ and are antichains of sets of states in $\mathrm{AC}_{\langle \wp(Q_2),\subseteq\rangle}$. Correspondingly, we consider the lattice $\langle \mathrm{AC}_{\langle \wp(Q_2),\subseteq\rangle}, \widetilde{\sqsubseteq}\rangle$, where for all $X, Y \in \mathrm{AC}_{\langle \wp(Q_2),\subseteq\rangle}$:

$$\begin{aligned} X \mathrel{\widetilde{\sqsubseteq}} Y &\overset{\triangle}{\Leftrightarrow} \forall T \in Y, \exists S \in X, S \subseteq T \\ X \mathrel{\widetilde{\sqcup}} Y &\triangleq \min(\{S \cup T \in \wp(Q_2) \mid S \in X, T \in Y\}) \\ X \mathrel{\widetilde{\sqcap}} Y &\triangleq \min(\{S \in \wp(Q_2) \mid S \in X \cup Y\}) \\ \text{with} \quad \min(X) &\triangleq \{S \in X \mid \forall S' \in X, S' \not\subset S\} \ . \end{aligned}$$

Then, these definitions allow us to replace Post by an equivalent function

$$\mathrm{Post}_{\mathcal{A}_1}^{\mathcal{A}_2} : (\mathrm{AC}_{\langle \wp(Q_2),\subseteq\rangle})^{|Q_1|} \to (\mathrm{AC}_{\langle \wp(Q_2),\subseteq\rangle})^{|Q_1|}$$

that transforms vectors of antichains as follows:

$$\mathrm{Post}_{\mathcal{A}_1}^{\mathcal{A}_2}(\langle X_q\rangle_{q\in Q_1}) \triangleq \langle \min(\{\mathrm{post}_a^{\mathcal{A}_2}(S) \in \wp(Q_2) \mid \exists a \in \Sigma, q' \in Q_1, S \in X_{q'}, q' \xrightarrow{a}_{\mathcal{A}_1} q\})\rangle_{q\in Q_1} \ .$$

In turn, the above $\mathcal{FP} \in \mathrm{AC}_{\langle Q_1\times\wp(Q_2),\subseteq_\times\rangle}$ is replaced by the following equivalent vector:

$$\overrightarrow{\mathcal{FP}} \triangleq \widetilde{\bigsqcup}\{\vec{X} \in (\mathrm{AC}_{\langle \wp(Q_2),\subseteq\rangle})^{|Q_1|} \mid \vec{X} = \mathrm{Post}_{\mathcal{A}_1}^{\mathcal{A}_2}(\vec{X}) \mathrel{\widetilde{\sqcap}} \langle \psi_{\varnothing}^{\{I_2\}}(q \in^{?} I_1)\rangle_{q\in Q_1}\} \ .$$

Finally, the condition $\exists q \in F_1, \mathcal{FP} \mathrel{\widetilde{\sqsubseteq}_\times} \{(q, F_2^c)\}$ is equivalent to $\exists q \in F_1, \overrightarrow{\mathcal{FP}}_q \mathrel{\widetilde{\sqsubseteq}} \{F_2^c\}$. □

Let us recall that $\mathcal{A}^R$ denotes the reverse automaton of $\mathcal{A}$, where arrows are flipped and the initial/final states become final/initial. Note that language inclusion can be decided by considering the reverse automata since $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2) \Leftrightarrow \mathcal{L}(\mathcal{A}_1^R) \subseteq \mathcal{L}(\mathcal{A}_2^R)$ holds. Furthermore, let us observe that $\mathrm{Post}_{\mathcal{A}_1}^{\mathcal{A}_2} = \mathrm{Pre}_{\mathcal{A}_1^R}^{\mathcal{A}_2^R}$. We therefore obtain the following consequence of Theorem 6.4.

Corollary 6.5. *Let*

$$\overrightarrow{\mathcal{FP}} \triangleq \widetilde{\bigsqcup}\{\vec{X} \in (\mathrm{AC}_{\langle \wp(Q_2),\subseteq\rangle})^{|Q_1|} \mid \vec{X} = \mathrm{Pre}_{\mathcal{A}_1}^{\mathcal{A}_2}(\vec{X}) \mathrel{\widetilde{\sqcap}} \langle \psi_{\varnothing}^{\{F_2\}}(q \in^{?} F_1)\rangle_{q\in Q_1}\} \ .$$

Then, $\mathcal{L}(\mathcal{A}_1) \nsubseteq \mathcal{L}(\mathcal{A}_2)$ *iff* $\exists q \in I_1, \overrightarrow{\mathcal{FP}}_q \mathrel{\widetilde{\sqsubseteq}} \{I_2^c\}$.

Since $\widetilde{\sqsubseteq} = \sqsubseteq^{-1}$, we have that $\widetilde{\sqcap} = \sqcup$, $\widetilde{\sqcup} = \sqcap$ and the greatest element $\varnothing$ for $\widetilde{\sqsubseteq}$ is the least element for $\sqsubseteq$. Moreover, by (21), $\alpha(\vec{\epsilon}^{F_1}) = \langle \psi_{\varnothing}^{\{F_2\}}(q \in^{?} F_1)\rangle_{q\in Q_1}$. Therefore, we can rewrite the vector $\overrightarrow{\mathcal{FP}}$ of Corollary 6.5 as

$$\overrightarrow{\mathcal{FP}} = \bigsqcap\{\vec{X} \in (\mathrm{AC}_{\langle \wp(Q_2),\subseteq\rangle})^{|Q_1|} \mid \vec{X} = \mathrm{Pre}_{\mathcal{A}_1}^{\mathcal{A}_2}(\vec{X}) \sqcup \alpha(\vec{\epsilon}^{F_1})\} \ ,$$

which is precisely the least fixpoint in $\langle (\mathrm{AC}_{\langle \wp(Q_2),\subseteq\rangle})^{|Q_1|}, \sqsubseteq\rangle$ of $\mathrm{Pre}_{\mathcal{A}_1}^{\mathcal{A}_2}$ above $\alpha(\vec{\epsilon}^{F_1})$. Hence, it turns out that the Kleene iterates of the least fixpoint computation that converge to $\overrightarrow{\mathcal{FP}}$ exactly coincide with the iterates computed by the Kleene procedure of the state-based algorithm `FAIncS`. In particular, if $\vec{Y}$ is the output vector of $\textsc{Kleene}(\sqsubseteq, \lambda\vec{X}.\, \alpha(\vec{\epsilon}^{F_1}) \sqcup \mathrm{Pre}_{\mathcal{A}_1}^{\mathcal{A}_2}(\vec{X}), \vec{\varnothing})$ at line 1 of `FAIncS` then $\vec{Y} = \overrightarrow{\mathcal{FP}}$. Furthermore, $\exists q \in I_1, \overrightarrow{\mathcal{FP}}_q \mathrel{\widetilde{\sqsubseteq}} \{I_2^c\} \Leftrightarrow \exists q \in I_1, \exists S \in \overrightarrow{\mathcal{FP}}_q,\ S \cap I_2 = \varnothing$. Summing up, the $\sqsubseteq$-lfp algorithm `FAIncS` exactly coincides with the $\widetilde{\sqsubseteq}$-gfp antichain algorithm as given by Corollary 6.5.

We can easily derive an antichain algorithm which is perfectly equivalent to `FAIncS` by considering the antichain lattice $\langle \mathrm{AC}_{\langle \wp(Q_2),\supseteq\rangle}, \sqsubseteq\rangle$ for the dual lattice $\langle \wp(Q_2), \supseteq\rangle$ and by replacing the functions α, γ and $\mathrm{Pre}_{\mathcal{A}_1}^{\mathcal{A}_2}$ of Lemma 6.2, resp., with the following dual versions:

$$\alpha^c(X) \triangleq \lfloor\{\mathrm{cpre}_u^{\mathcal{A}_2}(F_2^c) \in \wp(Q_2) \mid u \in X\}\rfloor, \qquad \gamma^c(Y) \triangleq \{v \in \Sigma^* \mid \exists y \in Y, y \supseteq \mathrm{cpre}_v^{\mathcal{A}_2}(F_2^c)\},$$

$$\mathrm{CPre}_{\mathcal{A}_1}^{\mathcal{A}_2}(\langle X_q\rangle_{q\in Q_1}) \triangleq \langle \lfloor\{\mathrm{cpre}_a^{\mathcal{A}_2}(S) \in \wp(Q_2) \mid \exists a \in \Sigma, q' \in Q_1,\ q' \in \delta_1(q, a) \wedge S \in X_{q'}\}\rfloor\rangle_{q\in Q_1},$$

where $\mathrm{cpre}_u^{\mathcal{A}_2}(S) \triangleq (\mathrm{pre}_u^{\mathcal{A}_2}(S^c))^c$ for $u \in \Sigma^*$. When using these functions, the corresponding algorithm computes on the abstract domain $\langle \mathrm{AC}_{\langle \wp(Q_2), \supseteq \rangle}, \sqsubseteq \rangle$ and it turns out that $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$ iff $\textsc{Kleene}(\sqsubseteq, \lambda \vec{X}^{\sharp}.\, \alpha^c(\vec{\epsilon}^{F_1}) \sqcup \mathrm{CPre}_{\mathcal{A}_1}^{\mathcal{A}_2}(\vec{X}^{\sharp}), \vec{\varnothing}) \sqsubseteq \alpha^c(\vec{L_2}^{I_1})$. This language inclusion algorithm coincides with the backward antichain algorithm defined by De Wulf et al. [2006, Theorem 6] since both compute on the same lattice, $\lfloor X \rfloor$ corresponds to the maximal (w.r.t. set inclusion) elements of X, $\alpha^c(\{\epsilon\}) = \{F_2^c\}$ and for all $X \in \alpha^c(\wp(\Sigma^*))$, we have that $X \sqsubseteq \alpha^c(L_2) \Leftrightarrow \forall S \in X,\ I_2 \nsubseteq S$.

We have thus shown that the two forward/backward antichain algorithms introduced by De Wulf et al. [2006] can be systematically derived by instantiating our framework. The original antichain algorithms were later improved by Abdulla et al. [2010] and, subsequently, by Bonchi and Pous [2013]. Among their improvements, they showed how to exploit a precomputed binary relation between pairs of states of the input automata such that language inclusion holds for all the pairs in the relation. When that binary relation is a simulation relation, our framework allows to partially match their results by using the simulation-based quasiorder $\leq_{\mathcal{A}}^r$ defined in Section 5.3.2. However, this relation $\leq_{\mathcal{A}}^r$ does not consider pairs of states $Q_2 \times Q_2$ whereas the aforementioned algorithms do.

7 INCLUSION FOR CONTEXT FREE LANGUAGES

A *context-free grammar* (CFG) is a tuple $\mathcal{G} = \langle \mathcal{V}, \Sigma, P \rangle$ where $\mathcal{V} = \{X_0, \ldots, X_n\}$ is a finite set of variables including a start symbol X_0, Σ is a finite alphabet of terminals and P is a finite set of productions $X_i \to \beta$ 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 $X_i \to \beta \in P$ is such that $\beta \in (\mathcal{V} \times \mathcal{V}) \cup \Sigma \cup \{\epsilon\}$ and if $\beta = \epsilon$ then $i = 0$ [Chomsky 1959]. We also assume that for all $X_i \in \mathcal{V}$ there exists a production $X_i \to \beta \in P$, otherwise X_i can be safely removed from $\mathcal{V}$. Given two strings $w, w' \in (\mathcal{V} \cup \Sigma)^*$ we write $w \to w'$ iff there exists $u, v \in (\mathcal{V} \cup \Sigma)^*$ and $X \to \beta \in P$ such that $w = uXv$ and $w' = u\beta v$. We denote by $\to^*$ the reflexive-transitive closure of $\to$. The language generated by a CFG $\mathcal{G}$ is $\mathcal{L}(\mathcal{G}) \triangleq \{w \in \Sigma^* \mid X_0 \to^* w\}$.

7.1 Extending the Framework to CFGs

Similarly to the case of automata, a CFG $\mathcal{G} = (\mathcal{V}, \Sigma, P)$ in CNF induces a set of equations:

$$\mathrm{Eqn}(\mathcal{G}) \triangleq \{X_i = \textstyle\bigcup_{X_i \to \beta_j \in P} \beta_j \mid i \in [0, n]\} \ .$$

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

$$W_S^{\mathcal{G}} \triangleq \{w \in \Sigma^* \mid \exists X \in S,\ X \to^* w\} \ .$$

When $S = \{X\}$ we slightly abuse the notation and write $W_X^{\mathcal{G}}$. Also, we drop the superscript $\mathcal{G}$ when the grammar is clear from the context. The language generated by $\mathcal{G}$ is therefore $\mathcal{L}(\mathcal{G}) = W_{X_0}^{\mathcal{G}}$.

We define the vector $\vec{b} \in \wp(\Sigma^*)^{|\mathcal{V}|}$ and the function $\mathrm{Fn}_{\mathcal{G}} : \wp(\Sigma^*)^{|\mathcal{V}|} \to \wp(\Sigma^*)^{|\mathcal{V}|}$, which are used to formalize the fixpoint equations in $\mathrm{Eqn}(\mathcal{G})$, as follows:

$$\vec{b} \triangleq \langle b_i \rangle_{i \in [0,n]} \in \wp(\Sigma^*)^{|\mathcal{V}|} \qquad \text{where } b_i \triangleq \{\beta \mid X_i \to \beta \in P,\ \beta \in \Sigma \cup \{\epsilon\}\} \ ,$$

$$\mathrm{Fn}_{\mathcal{G}}(\langle X_i \rangle_{i \in [0,n]}) \triangleq \langle \beta_1^{(i)} \cup \ldots \cup \beta_{k_i}^{(i)} \rangle_{i \in [0,n]} \qquad \text{where } \beta_j^{(i)} \in \mathcal{V}^2 \text{ and } X_i \to \beta_j^{(i)} \in P \ .$$

Notice that $\lambda \vec{X}.\, \vec{b} \cup \mathrm{Fn}_{\mathcal{G}}(\vec{X})$ is a well-defined monotonic function in $\wp(\Sigma^*)^{|\mathcal{V}|} \to \wp(\Sigma^*)^{|\mathcal{V}|}$, which therefore has the least fixpoint $\langle Y_i \rangle_{i \in [0,n]} = \mathrm{lfp}(\lambda \vec{X}.\, \vec{b} \cup \mathrm{Fn}_{\mathcal{G}}(\vec{X}))$. It is known [Ginsburg and Rice 1962] that the language $\mathcal{L}(\mathcal{G})$ accepted by $\mathcal{G}$ is such that $\mathcal{L}(\mathcal{G}) = Y_0$.

Example 7.1. Consider the CFG $\mathcal{G} = \langle\{X_0, X_1\}, \{a, b\}, \{X_0 \to X_0X_1 \mid X_1X_0 \mid b,\ X_1 \to a\}\rangle$ in CNF. The corresponding equation system is

$$\mathrm{Eqn}(\mathcal{G}) = \begin{cases} X_0 = X_0X_1 \cup X_1X_0 \cup \{b\} \\ X_1 = \{a\} \end{cases}$$

so that

$$\begin{pmatrix} W_{X_0} \\ W_{X_1} \end{pmatrix} = \mathrm{lfp}\left(\lambda \begin{pmatrix} X_0 \\ X_1 \end{pmatrix} . \begin{pmatrix} X_0X_1 \cup X_1X_0 \cup \{b\} \\ \{a\} \end{pmatrix}\right) = \begin{pmatrix} a^*ba^* \\ a \end{pmatrix} .$$

Moreover, we have that $\vec{\boldsymbol{b}} \in \wp(\Sigma^*)^2$ and $\mathrm{Fn}_{\mathcal{G}} : \wp(\Sigma^*)^2 \to \wp(\Sigma^*)^2$ are given by

$$\vec{\boldsymbol{b}} = \langle\{b\}, \{a\}\rangle \qquad\qquad \mathrm{Fn}_{\mathcal{G}}(\langle X_0, X_1\rangle) = \langle X_0X_1 \cup X_1X_0, \varnothing\rangle \ . \qquad \diamond$$

It turns out that

$$\mathcal{L}(\mathcal{G}) \subseteq L_2 \ \Leftrightarrow\ \mathrm{lfp}(\lambda\vec{\boldsymbol{X}}.\, \vec{\boldsymbol{b}} \cup \mathrm{Fn}_{\mathcal{G}}(\vec{\boldsymbol{X}})) \subseteq \vec{\boldsymbol{L}}_2^{X_0}$$

where $\vec{\boldsymbol{L}}_2^{X_0} \triangleq \langle\psi_{\Sigma^*}^{L_2}(i \stackrel{?}{=} 0)\rangle_{i\in[0,n]}$.

Theorem 7.2. *Let $\mathcal{G} = (\mathcal{V}, \Sigma, P)$ be a CFG in CNF. If $\rho \in \mathrm{uco}(\wp(\Sigma^*))$ is backward complete for both $\lambda X.Xa$ and $\lambda X.aX$, for all $a \in \Sigma$, then ρ is backward complete for $\lambda\vec{\boldsymbol{X}}.\, \vec{\boldsymbol{b}} \cup \mathrm{Fn}_{\mathcal{G}}(\vec{\boldsymbol{X}})$.*

Proof. Let us first show that backward completeness for left and right concatenation can be extended from letter to words. We give the proof for left concatenation, the right case is symmetric. We prove that $\rho(wX) = \rho(w\rho(X))$ for every $w \in \Sigma^*$. We proceed by induction on $|w| \geq 0$. The base case $|w| = 0$ iff $w = \epsilon$ is trivial because ρ is idempotent. For the inductive case $|w| > 0$ let $w = au$ for some $u \in \Sigma^*$ and $a \in \Sigma$, so that:

$$\begin{aligned} \rho(auX) = &\quad \text{[by backward completeness for } \lambda X.\, aX] \\ \rho(a\rho(uX)) = &\quad \text{[by inductive hypothesis]} \\ \rho(a\rho(u\rho(X))) = &\quad \text{[by backward completeness for } \lambda X.\, aX] \\ \rho(au\rho(X)) \ . & \end{aligned}$$

Next we turn to the binary concatenation case, i.e., we prove that $\rho(YZ) = \rho(\rho(Y)\rho(Z))$ for all $Y, Z \in \wp(\Sigma^*)$:

$$\begin{aligned} \rho(\rho(Y)\rho(Z)) = &\quad \text{[by definition of concatenation]} \\ \rho(\textstyle\bigcup_{u\in\rho(Y)} u\rho(Z)) = &\quad \text{[by (3)]} \\ \rho(\textstyle\bigcup_{u\in\rho(Y)} \rho(u\rho(Z))) = &\quad \text{[by backward completeness of } \lambda X.\, uX] \\ \rho(\textstyle\bigcup_{u\in\rho(Y)} \rho(uZ)) = &\quad \text{[by (3)]} \\ \rho(\textstyle\bigcup_{u\in\rho(Y)} uZ) = &\quad \text{[by definition of concatenation]} \\ \rho(\rho(Y)Z) = &\quad \text{[by definition of concatenation]} \\ \rho(\textstyle\bigcup_{v\in Z} \rho(Y)v) = &\quad \text{[by (3)]} \\ \rho(\textstyle\bigcup_{v\in Z} \rho(\rho(Y)v)) = &\quad \text{[by backward completeness of } \lambda X.\, Xv] \\ \rho(\textstyle\bigcup_{v\in Z} \rho(Yv)) = &\quad \text{[by (3)]} \\ \rho(\textstyle\bigcup_{v\in Z} Yv) = &\quad \text{[by definition of concatenation]} \\ \rho(YZ) \ . & \end{aligned}$$

Then, the proof follows the same lines of the proof of Theorem 4.3. Indeed, it follows from the definition of $\mathrm{Fn}_{\mathcal{G}}(\langle X_i\rangle_{i\in[0,n]})$ that:

$$\rho(\textstyle\bigcup_{j=1}^{k_i} \beta_j^{(i)}) = \quad \text{[by definition of } \beta_j^{(i)}]$$

$$\begin{aligned}
\rho(\textstyle\bigcup_{j=1}^{k_i} X_j^{(i)} Y_j^{(i)}) = &\quad \text{[by (3)]} \\
\rho(\textstyle\bigcup_{j=1}^{k_i} \rho(X_j^{(i)} Y_j^{(i)})) = &\quad \text{[by backward completeness of } \rho \text{ for binary concatenation]} \\
\rho(\textstyle\bigcup_{j=1}^{k_i} \rho(\rho(X_j^{(i)})\rho(Y_j^{(i)}))) = &\quad \text{[by (3)]} \\
\rho(\textstyle\bigcup_{j=1}^{k_i} \rho(X_j^{(i)})\rho(Y_j^{(i)})) &\quad .
\end{aligned}$$

Hence, by a straightforward componentwise application on vectors in $\wp(\Sigma^*)^{|\mathcal{V}|}$, we obtain that ρ is backward complete for $\mathrm{Fn}_{\mathcal{G}}$. Finally, ρ is backward complete for $\lambda\vec{\boldsymbol{X}}.\,(\vec{\boldsymbol{b}} \cup \mathrm{Fn}_{\mathcal{G}}(\vec{\boldsymbol{X}}))$, because:

$$\begin{aligned}
\rho(\vec{\boldsymbol{b}} \cup \mathrm{Fn}_{\mathcal{G}}(\rho(\vec{\boldsymbol{X}}))) = &\quad \text{[by (3)]} \\
\rho(\rho(\vec{\boldsymbol{b}}) \cup \rho(\mathrm{Fn}_{\mathcal{G}}(\rho(\vec{\boldsymbol{X}})))) = &\quad \text{[by backward completeness for } \mathrm{Fn}_{\mathcal{G}}] \\
\rho(\rho(\vec{\boldsymbol{b}}) \cup \rho(\mathrm{Fn}_{\mathcal{G}}(\vec{\boldsymbol{X}}))) = &\quad \text{[by (3)]} \\
\rho(\vec{\boldsymbol{b}} \cup \mathrm{Fn}_{\mathcal{G}}(\vec{\boldsymbol{X}})) &\quad .
\end{aligned}$$

□

The following result, which is an adaptation of Theorem 4.7 to grammars, relies on Theorem 7.2 for designing an algorithm that solves the inclusion problem $\mathcal{L}(\mathcal{G}) \subseteq L_2$ by exploiting a language abstraction ρ that satisfies some requirements of backward completeness and computability.

Theorem 7.3. *Let $\mathcal{G} = \langle \mathcal{V}, \Sigma, P \rangle$ be a CFG in CNF, $L_2 \in \wp(\Sigma^*)$ and $\rho \in \mathrm{uco}(\Sigma^*)$. Assume that the following properties hold:*

(i) *The closure ρ is backward complete for both $\lambda X \in \wp(\Sigma^*).\, aX$ and $\lambda X \in \wp(\Sigma^*).\, Xa$, for all $a \in \Sigma$, and satisfies $\rho(L_2) = L_2$.*
(ii) *$\rho(\wp(\Sigma^*))$ does not contain infinite ascending chains.*
(iii) *If $X, Y \in \wp(\Sigma^*)$ are finite sets of words then the inclusion $\rho(X) \subseteq^? \rho(Y)$ is decidable.*
(iv) *If $Y \in \wp(\Sigma^*)$ is a finite set of words then the inclusion $\rho(Y) \subseteq^? L_2$ is decidable.*

Then,

$\langle Y_i \rangle_{i \in [0,n]} := \textsc{Kleene}(\mathrm{Incl}_\rho, \lambda\vec{\boldsymbol{X}}.\, \vec{\boldsymbol{b}} \cup \mathrm{Fn}_{\mathcal{G}}(\vec{\boldsymbol{X}}), \vec{\boldsymbol{\varnothing}})$;
return $\mathrm{Incl}_\rho(\langle Y_i \rangle_{i \in [0,n]}, \overrightarrow{\boldsymbol{L_2}}^{X_0})$;

is a decision algorithm for $\mathcal{L}(\mathcal{G}) \subseteq L_2$.

Proof. Analogous to the proof of Theorem 4.7. □

7.2 Instantiating the Framework

Let us instantiate the general algorithmic framework provided by Theorem 7.3 to the class of closure operators induced by quasiorder relations on words. As a consequence of Lemma 5.2, we have the following characterization of L-consistent quasiorders.

Lemma 7.4. *Let $L \in \wp(\Sigma^*)$ and $\leqslant_L$ be a quasiorder on Σ^*. Then, $\leqslant_L$ is a L-consistent quasiorder on Σ^* if and only if*
(a) *$\rho_{\leqslant_L}(L) = L$, and*
(b) *$\rho_{\leqslant_L}$ is backward complete for $\lambda X.\, aX$ and $\lambda X.\, Xa$, for all $a \in \Sigma$.*

Analogously to Section 5.1 for automata, Theorem 7.3 induces an algorithm for deciding the language inclusion $\mathcal{L}(\mathcal{G}) \subseteq L_2$ for any CFG $\mathcal{G}$ and regular language L_2. More in general, given a language $L_2 \in \wp(\Sigma^*)$ whose membership problem is decidable and a decidable L_2-consistent wqo, the following algorithm `CFGIncW` (`CFG Inclusion` based on `Words`) decides $\mathcal{L}(\mathcal{G}) \subseteq L_2$.

CFGIncW: Word-based algorithm for $\mathcal{L}(\mathcal{G}) \subseteq L_2$

Data: CFG $\mathcal{G} = \langle \mathcal{V}, \Sigma, P \rangle$; decision procedure for $u \in^? L_2$; decidable L_2-consistent wqo $\leqslant_{L_2}$.

$\langle Y_i \rangle_{i \in [0,n]} := \textsc{Kleene}(\sqsubseteq_{\leqslant_{L_2}}, \lambda \vec{X}.\, \vec{b} \cup \mathrm{Fn}_{\mathcal{G}}(\vec{X}), \vec{\varnothing})$;
forall $u \in Y_0$ **do**
  if $u \notin L_2$ **then return** *false*;
return *true*;

Theorem 7.5. *Let $\mathcal{G} = \langle Q, \delta, I, F, \Sigma \rangle$ be a CFG and let $L_2 \in \wp(\Sigma^*)$ be a language such that:* (i) *membership $u \in^? L_2$ is decidable;* (ii) *there exists a decidable L_2-consistent wqo on Σ^*. Then, Algorithm* CFGIncW *decides the inclusion $\mathcal{L}(\mathcal{G}) \subseteq L_2$.*

Proof. The proof is analogous to the proof of Theorem 5.3: it applies Theorem 7.3 and Lemma 7.4 in the same way of the proof of Theorem 5.3 where the role of a left L_2-consistent wqo on Σ^* is replaced by a L_2-consistent wqo. □

7.2.1 Myhill and State-based Quasiorders. In the following, we will consider two quasiorders on Σ^* and we will show that they fulfill the requirements of Theorem 7.5, so that they correspondingly yield algorithms for deciding the language inclusion $\mathcal{L}(\mathcal{G}) \subseteq L_2$ for every CFG $\mathcal{G}$ and regular language L_2.

The *context* for a language $L \in \wp(\Sigma^*)$ w.r.t. a given word $w \in \Sigma^*$ is defined as usual:

$$\mathrm{ctx}_L(w) \triangleq \{(u, v) \in \Sigma^* \times \Sigma^* \mid uwv \in L\} \ .$$

Correspondingly, let us define the following quasiorder relation on $\leqq_L \subseteq \Sigma^* \times \Sigma^*$:

$$u \leqq_L v \stackrel{\triangle}{\iff} \mathrm{ctx}_L(u) \subseteq \mathrm{ctx}_L(v) \ . \tag{22}$$

De Luca and Varricchio [1994, Section 2] call $\leqq_L$ the *Myhill quasiorder relative to L*. The following result is the analogue of Lemma 5.6 for the Nerode quasiorder: it shows that the Myhill quasiorder is the weakest L_2-consistent quasiorder for which the above algorithm CFGIncW can be instantiated to decide a language inclusion $\mathcal{L}(\mathcal{G}) \subseteq L_2$.

Lemma 7.6. *Let $L \in \wp(\Sigma^*)$.*
(a) *$\leqq_L$ is a L-consistent quasiorder. If L is regular then, additionally, $\leqq_L$ is a decidable wqo.*
(b) *If $\leqslant$ is a L-consistent quasiorder on Σ^* then $\rho_{\leqq_L}(\wp(\Sigma^*)) \subseteq \rho_{\leqslant}(\wp(\Sigma^*))$.*

Proof. The proof follows the same lines of the proof of Lemma 5.6.
Let us consider (a). De Luca and Varricchio [1994, Section 2] observe that $\leqq_L$ is monotonic. Moreover, if L is regular then $\leqq_L$ is a wqo [de Luca and Varricchio 1994, Proposition 2.3]. Let us observe that given $u \in L$ and $v \notin L$ we have that $(\epsilon, \epsilon) \in \mathrm{ctx}_L(u)$ while $(\epsilon, \epsilon) \notin \mathrm{ctx}_L(v)$. Hence, $\leqq_L$ is a L-consistent quasiorder. Finally, if L is regular then $\leqq_L$ is clearly decidable.
Let us consider (b). By the characterization of L-consistent quasiorders of Lemma 7.4, De Luca and Varricchio [1994, Section 2, point 4] observe that $\leqq_L$ is maximum in the set of all L-consistent quasiorders, i.e. every L-consistent quasiorder $\leqslant$ is such that $x \leqslant y \Rightarrow x \leqq_L y$. As a consequence, $\rho_{\leqslant}(X) \subseteq \rho_{\leqq_L}(X)$ holds for all $X \in \wp(\Sigma^*)$, namely, $\rho_{\leqq_L}(\wp(\Sigma^*)) \subseteq \rho_{\leqslant}(\wp(\Sigma^*))$. □

Example 7.7. Let us illustrate the use of the Myhill quasiorder $\leqq_{\mathcal{L}(\mathcal{A})}$ in Algorithm CFGIncW for solving the language inclusion $\mathcal{L}(\mathcal{G}) \subseteq \mathcal{L}(\mathcal{A})$, where $\mathcal{G}$ is the CFG in Example 7.1 and $\mathcal{A}$ is the

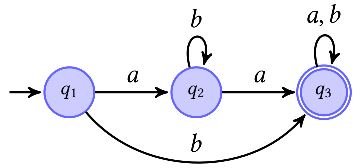


Fig. 4. A finite automaton $\mathcal{A}$ with $\mathcal{L}(\mathcal{A}) = (b + ab^*a)(a + b)^*$.

FA depicted in Figure 4. The equations for $\mathcal{G}$ are as follows:

$$\mathrm{Eqn}(\mathcal{G}) = \begin{cases} X_0 = X_0X_1 \cup X_1X_0 \cup \{b\} \\ X_1 = \{a\} \end{cases} .$$

We write $\{(S,T)\} \cup \{(X,Y)\}$ to compactly denote a set $\{(u,v) \mid (u,v) \in S \times T \cup X \times Y\}$. Then, we have the following contexts (among others) for $L = \mathcal{L}(\mathcal{A}) = (b + ab^*a)(a + b)^*$:

$$\begin{aligned}
\mathrm{ctx}_L(\epsilon) &= \{(\epsilon, L)\} \cup \{(ab^*, b^*a\Sigma^*)\} \cup \{(L, \Sigma^*)\} \\
\mathrm{ctx}_L(a) &= \{(\epsilon, b^*a\Sigma^*)\} \cup \{ab^*, \Sigma^*\} \cup \{(L, \Sigma^*)\} \\
\mathrm{ctx}_L(b) &= \{(\epsilon, \Sigma^*)\} \cup \{(ab^*, b^*a\Sigma^*)\} \cup \{(L, \Sigma^*)\} \\
\mathrm{ctx}_L(ba) &= \{(\epsilon, \Sigma^*)\} \cup \{(ab^*, \Sigma^*)\} \cup \{(L, \Sigma^*)\}
\end{aligned}$$

Notice that $a \leqq_L ba$, $\mathrm{ctx}_L(ab) = \mathrm{ctx}_L(a)$ and $\mathrm{ctx}_L(ba) = \mathrm{ctx}_L(baa) = \mathrm{ctx}_L(aab) = \mathrm{ctx}_L(aba)$. Next, we show the computation of the Kleene iterates according to Algorithm CFGIncW using $\sqsubseteq_{\leqq_L}$ by recalling from Example 7.1 that $\vec{b} = \langle\{b\}, \{a\}\rangle$ and $\mathrm{Fn}_{\mathcal{G}}(\langle X_0, X_1\rangle) = \langle X_0X_1 \cup X_1X_0, \varnothing\rangle$:

$\vec{Y}^{(0)} = \vec{\varnothing}$

$\vec{Y}^{(1)} = \vec{b} = \langle\{b\}, \{a\}\rangle$

$\vec{Y}^{(2)} = \vec{b} \cup \mathrm{Fn}_{\mathcal{G}}(\vec{Y}^{(1)}) = \langle\{b\}, \{a\}\rangle \cup \langle\{ba, ab\}, \varnothing\rangle = \langle\{ba, ab, b\}, \{a\}\rangle$

$\vec{Y}^{(3)} = \vec{b} \cup \mathrm{Fn}_{\mathcal{G}}(\vec{Y}^{(2)}) = \langle\{b\}, \{a\}\rangle \cup \langle\{baa, aba, ba, aab, ab\}, \varnothing\rangle = \langle\{baa, aba, ba, aab, ab, b\}, \{a\}\rangle$

It turns out that $\langle\{baa, aba, ba, aab, ab, b\}, \{a\}\rangle \sqsubseteq_{\leqq_L} \langle\{ba, ab, b\}, \{a\}\rangle$ because $a \leqq_L baa$, $a \leqq_L aba$, $a \leqq_L aab$ hold, so that $\textsc{Kleene}(\sqsubseteq_{\leqq_L}, \lambda\vec{X}.\, \vec{b} \cup \mathrm{Fn}_{\mathcal{G}}(\vec{X}), \vec{\varnothing})$ stops with $\vec{Y}^{(3)}$ and outputs $\vec{Y} = \langle\{ba, ab, b\}, \{a\}\rangle$. Since $ab \in \vec{Y}_0$ but $ab \notin \mathcal{L}(\mathcal{A})$, Algorithm CFGIncW correctly concludes that $\mathcal{L}(\mathcal{G}) \subseteq \mathcal{L}(\mathcal{A})$ does not hold. ◇

Similarly to Section 5.3, next we consider a state-based quasiorder that can be used with Algorithm CFGIncW. First, given a FA $\mathcal{A} = \langle Q, \delta, I, F, \Sigma\rangle$ we define the state-based equivalent of the context of a word $w \in \Sigma^*$ as follows:

$$\mathrm{ctx}_{\mathcal{A}}(w) \triangleq \{(q, q') \in Q \times Q \mid q \overset{w}{\rightsquigarrow} q'\} .$$

Then, the quasiorder $\leq_{\mathcal{A}}$ on Σ^* induced by $\mathcal{A}$ is defined as follows: for all $u, v \in \Sigma^*$,

$$u \leq_{\mathcal{A}} v \overset{\triangle}{\Longleftrightarrow} \mathrm{ctx}_{\mathcal{A}}(u) \subseteq \mathrm{ctx}_{\mathcal{A}}(v) . \tag{23}$$

The following result is the analogue of Lemma 5.8 and shows that $\leq_{\mathcal{A}}$ is a $\mathcal{L}(\mathcal{A})$-consistent well-quasiorder and, therefore, it can be used with Algorithm CFGIncW to solve a language inclusion $\mathcal{L}(\mathcal{G}) \subseteq \mathcal{L}(\mathcal{A})$.

Lemma 7.8. *The relation $\leq_{\mathcal{A}}$ is a decidable $\mathcal{L}(\mathcal{A})$-consistent wqo.*

Proof. For every $u \in \Sigma^*$, $\mathrm{ctx}_{\mathcal{A}}(u)$ is a finite and computable set, so that $\leq_{\mathcal{A}}$ is a decidable wqo. Next, we show that $\leq_{\mathcal{A}}$ is $\mathcal{L}(A)$-consistent according to Definition 5.1 (a)-(b).
(a) By picking $u \in \mathcal{L}(\mathcal{A})$ and $v \notin \mathcal{L}(\mathcal{A})$ we have that $\mathrm{ctx}_{\mathcal{A}}(u)$ contains a pair (q_i, q_f) with $q_i \in I$ and $q_f \in F$ while $\mathrm{ctx}_{\mathcal{A}}(v)$ does not, hence $u \nleq_{\mathcal{A}} v$.
(b) Let us check that $\leq_{\mathcal{A}}$ is monotonic. Observe that $\mathrm{ctx}_{\mathcal{A}} : \langle \Sigma^*, \leq_{\mathcal{A}} \rangle \to \langle \wp(Q^2), \subseteq \rangle$ is monotonic. Therefore, for all $x_1, x_2 \in \Sigma^*$ and $a, b \in \Sigma$,

$$
\begin{aligned}
x_1 \leq_{\mathcal{A}} x_2 \Rightarrow &\quad \text{[by definition of } \leq_{\mathcal{A}}] \\
\mathrm{ctx}_{\mathcal{A}}(x_1) \subseteq \mathrm{ctx}_{\mathcal{A}}(x_2) \Rightarrow &\quad \text{[as } \mathrm{ctx}_{\mathcal{A}} \text{ is monotonic]} \\
\mathrm{ctx}_{\mathcal{A}}(ax_1b) \subseteq \mathrm{ctx}_{\mathcal{A}}(ax_2b) \Rightarrow &\quad \text{[by definition of } \leq_{\mathcal{A}}] \\
ax_1b \leq_{\mathcal{A}} ax_2b\ . &
\end{aligned}
$$

□

For the Myhill wqo $\leqq_{\mathcal{L}(\mathcal{A})}$, it turns out that for all $u, v \in \Sigma^*$,

$$
\begin{aligned}
&u \leqq_{\mathcal{L}(\mathcal{A})} v \Leftrightarrow \mathrm{ctx}_{\mathcal{L}(\mathcal{A})}(u) \subseteq \mathrm{ctx}_{\mathcal{L}(\mathcal{A})}(v) \Leftrightarrow \\
&\{(x,y) \mid x \in W_{I,q} \wedge y \in W_{q',F} \wedge q \overset{u}{\rightsquigarrow} q'\} \subseteq \{(x,y) \mid x \in W_{I,q} \wedge y \in W_{q',F} \wedge q \overset{v}{\rightsquigarrow} q'\}\ .
\end{aligned}
$$

Therefore, $u \leq_{\mathcal{A}} v \Rightarrow u \leqq_{\mathcal{L}(\mathcal{A})} v$, and, consequently, $\rho_{\leqq_{\mathcal{L}(\mathcal{A})}}(\wp(\Sigma^*)) \subseteq \rho_{\leq^l_{\mathcal{A}_2}}(\wp(\Sigma^*))$ holds.

Example 7.9. Let us illustrate the use of the state-based quasiorder $\leq_{\mathcal{A}}$ to solve the language inclusion $\mathcal{L}(\mathcal{G}) \subseteq \mathcal{L}(\mathcal{A})$ of Example 7.7. Here, among others, we have the following contexts:

$$
\begin{aligned}
\mathrm{ctx}_{\mathcal{A}}(\epsilon) &= \{(q_1,q_1),(q_2,q_2),(q_3,q_3)\} & \mathrm{ctx}_{\mathcal{A}}(a) &= \{(q_1,q_2),(q_2,q_3),(q_3,q_3)\} \\
\mathrm{ctx}_{\mathcal{A}}(b) &= \{(q_1,q_3),(q_2,q_2),(q_3,q_3)\} & \mathrm{ctx}_{\mathcal{A}}(ba) &= \{(q_1,q_3),(q_2,q_3),(q_3,q_3)\}
\end{aligned}
$$

Moreover, $\mathrm{ctx}_{\mathcal{A}}(ba) = \mathrm{ctx}_{\mathcal{A}}(baa) = \mathrm{ctx}_{\mathcal{A}}(aab) = \mathrm{ctx}_{\mathcal{A}}(aba)$. Recall from Example 7.7 that for the Myhill quasiorder we have that $a \leqq_{\mathcal{L}(\mathcal{A})} ba$, while for the state-based quasiorder $a \nleq_{\mathcal{A}} ba$. The Kleene iterates computed by Algorithm `CFGIncW` when using $\sqsubseteq_{\leq_{\mathcal{A}}}$ are exactly the same of Example 7.7. Here, it turns out that `CFGIncW` outputs $\vec{Y}^{(2)} = \langle \{ba, ab, b\}, \{a\}\rangle$ because $\vec{Y}^{(3)} = \langle \{baa, aba, ba, aab, ab, b\}, \{a\}\rangle \sqsubseteq_{\leq_L} \langle \{ba, ab, b\}, \{a\}\rangle = \vec{Y}^{(2)}$ holds: indeed, we have that $ba \leq_{\mathcal{A}} baa$, $ba \leq_{\mathcal{A}} aba$, and $ba \leq_{\mathcal{A}} aab$ hold. Since $ab \in \vec{Y}^{(2)}_0$ but $ab \notin \mathcal{L}(\mathcal{A})$, Algorithm `CFGIncW` derives that $\mathcal{L}(\mathcal{G}) \nsubseteq \mathcal{L}(\mathcal{A})$. ◇

7.3 An Antichain Inclusion Algorithm for CFGs

We can easily formulate an equivalent of Theorem 6.1 for context-free languages, therefore defining an algorithm for solving $\mathcal{L}(\mathcal{G}) \subseteq L_2$ by computing on an abstract domain as defined by a Galois connection.

Theorem 7.10. *Let $\mathcal{G} = \langle \mathcal{V}, \Sigma, P\rangle$ be a CFG in CNF and let $L_2 \in \wp(\Sigma^*)$. Let $\langle D, \leq_D\rangle$ be a poset and $\langle \wp(\Sigma^*), \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle D, \sqsubseteq\rangle$ be a GC. Assume that the following properties hold:*

(i) *$L_2 \in \gamma(D)$ and for every $a \in \Sigma$, $X \in \wp(\Sigma^*)$, $\gamma(\alpha(aX)) = \gamma(\alpha(a\gamma(\alpha(X))))$ and $\gamma(\alpha(Xa)) = \gamma(\alpha(\gamma(\alpha(X))a))$.*
(ii) *$(D, \leq_D, \sqcup, \bot_D)$ is an effective domain, meaning that: $(D, \leq_D, \sqcup, \bot_D)$ 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.*
(iii) *There is an algorithm, say $\mathrm{Fn}^\sharp$, which computes $\alpha \circ \mathrm{Fn}_{\mathcal{G}} \circ \gamma$.*
(iv) *There is an algorithm, say $b^\sharp$, which computes $\alpha(\vec{b})$.*
(v) *There is an algorithm, say $\mathrm{Incl}^\sharp$, which decides $\vec{X}^\sharp \leq_D \alpha(\vec{L_2^{X_0}})$, for all $\vec{X}^\sharp \in \alpha(\wp(\Sigma^*))^{|\mathcal{V}|}$.*

Then,

$\langle Y_i^\sharp \rangle_{i\in[0,n]} := \text{Kleene}(\leq_D, \lambda\vec{X}^\sharp.\, b^\sharp \sqcup \mathrm{Fn}^\sharp(\vec{X}^\sharp), \overrightarrow{\bot_D});$
return $\mathrm{Incl}^\sharp(\langle Y_i^\sharp \rangle_{i\in[0,n]});$

is a decision algorithm for $\mathcal{L}(\mathcal{G}) \subseteq L_2$.

Proof. Analogous to the proof of Theorem 6.1. □

Similarly to what is done in Section 6.1, in order to solve an inclusion problem $\mathcal{L}(\mathcal{G}) \subseteq \mathcal{L}(\mathcal{A})$, where $\mathcal{A} = \langle Q, \delta, I, F, \Sigma\rangle$ is a FA, we leverage Theorem 7.10 to systematically design a "state-based" algorithm that computes Kleene iterates on the antichain poset $\langle \mathrm{AC}_{\langle \wp(Q\times Q),\subseteq\rangle}, \sqsubseteq\rangle$ viewed as an abstraction of $\langle \wp(\Sigma^*), \subseteq\rangle$. Here, the abstraction and concretization maps $\alpha\colon \wp(\Sigma^*) \to \mathrm{AC}_{\langle \wp(Q\times Q),\subseteq\rangle}$ and $\gamma\colon \mathrm{AC}_{\langle \wp(Q\times Q),\subseteq\rangle} \to \wp(\Sigma^*)$ and the function $\mathrm{Fn}_{\mathcal{G}}^{\mathcal{A}} : (\mathrm{AC}_{\langle \wp(Q\times Q),\subseteq\rangle})^{|\mathcal{V}|} \to (\mathrm{AC}_{\langle \wp(Q\times Q),\subseteq\rangle})^{|\mathcal{V}|}$ are defined as follows:

$$\alpha(X) \triangleq \lfloor \{\mathrm{ctx}_{\mathcal{A}}(u) \in \wp(Q\times Q) \mid u \in X\} \rfloor\,, \qquad \gamma(Y) \triangleq \{v \in \Sigma^* \mid \exists y \in Y, y \subseteq \mathrm{ctx}_{\mathcal{A}}(v)\}\,,$$
$$\mathrm{Fn}_{\mathcal{G}}^{\mathcal{A}}(\langle X_i\rangle_{i\in[0,n]}) \triangleq \langle \lfloor\{X_j \circ X_k \in \wp(Q\times Q) \mid X_i \to X_j X_k \in P\}\rfloor\rangle_{i\in[0,n]}\,,$$

where $\lfloor X \rfloor$ is the unique minor set w.r.t. subset inclusion of some $X \subseteq \wp(Q\times Q)$ and $X \circ Y \triangleq \{(q,q') \in Q\times Q \mid (q,q'') \in X,\ (q'',q') \in Y\}$ denotes the standard composition of two relations $X, Y \subseteq Q\times Q$. By the analogue of Lemma 6.2 (the proof follows the same pattern and is therefore omitted), it turns out that:

(a) $\langle \wp(\Sigma^*), \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathrm{AC}_{\langle \wp(Q\times Q),\subseteq\rangle}, \sqsubseteq\rangle$ is a GC,
(b) $\gamma \circ \alpha = \rho_{\leqslant_{\mathcal{A}}}$,
(c) $\mathrm{Fn}_{\mathcal{G}}^{\mathcal{A}} = \alpha \circ \mathrm{Fn}_{\mathcal{G}} \circ \gamma$.

Thus, the GC $\langle \wp(\Sigma^*), \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathrm{AC}_{\langle \wp(Q\times Q),\subseteq\rangle}, \sqsubseteq\rangle$ and the abstract function $\mathrm{Fn}_{\mathcal{G}}^{\mathcal{A}}$ satisfy the hypotheses (i)-(iv) of Theorem 7.10. Here, the inclusion check $\vec{X}^\sharp \leq_D \alpha(\overrightarrow{\mathcal{L}(\mathcal{A})}^{X_0})$ boils down to verify that for the start component Y_0 of the output $\langle Y_i\rangle_{i\in[0,n]}$ of $\text{Kleene}(\sqsubseteq, \lambda\vec{X}^\sharp.\, \alpha(\vec{b}) \sqcup \mathrm{Fn}_{\mathcal{G}}^{\mathcal{A}}(\vec{X}^\sharp), \vec{\varnothing})$, for all $R \in Y_0$, R does not contain a pair $(q_i, q_f) \in I \times F$. We therefore derive the following state-based algorithm `CFGIncS` (`S` stands for state) that decides an inclusion $L(\mathcal{G}) \subseteq L(\mathcal{A})$ on the abstract domain of antichains $\mathrm{AC}_{\langle \wp(Q\times Q),\subseteq\rangle}$.

`CFGIncS`**:** State-based algorithm for $L(\mathcal{G}) \subseteq L(\mathcal{A})$

Data: CFG $\mathcal{G} = \langle \mathcal{V}, \Sigma, P\rangle$ and FA $\mathcal{A} = \langle Q, \delta, I, F, \Sigma\rangle$

1 $\langle Y_i\rangle_{i\in[0,n]} := \text{Kleene}(\lambda\vec{X}^\sharp.\, \alpha(\vec{b}) \sqcup \mathrm{Fn}_{\mathcal{G}}^{\mathcal{A}}(\vec{X}^\sharp), \vec{\varnothing});$
2 **forall** $R \in Y_0$ **do**
3   **if** $R \cap (I \times F) = \varnothing$ **then return** *false*;
4 **return** *true*;

Theorem 7.11. *The algorithm* `CFGIncS` *decides the inclusion problem* $L(\mathcal{G}) \subseteq L(\mathcal{A})$.

Proof. The proof follows the same pattern of the proof of Theorem 6.3. We just focus on the inclusion check at lines 2-4, which is slightly different from the check at lines 2-5 of Algorithm `FAIncS`. Let $L_2 = \mathcal{L}(\mathcal{A})$. Since $\alpha(\overrightarrow{L_2}^{X_0}) = \langle \alpha(\psi_{\Sigma^*}^{L_2}(i \overset{?}{=} 0))\rangle_{i\in[0,n]}$, for all $\vec{Y} \in \alpha(\wp(\Sigma^*))^{|\mathcal{V}|}$ the relation $\vec{Y} \sqsubseteq \alpha(\overrightarrow{L_2}^{X_0})$ trivially holds for all components Y_i with $i \neq 0$. For Y_0, it is enough to prove that $Y_0 \sqsubseteq \alpha(L_2) \Leftrightarrow \forall R \in Y_q,\ R \cap (I \times F) \neq \varnothing$:

$$Y_0 \sqsubseteq \alpha(L_2) \Leftrightarrow \qquad [\text{since } Y_0 = \alpha(U) \text{ for some } U \in \wp(\Sigma^*)]$$

$$\begin{aligned}
\alpha(U) \sqsubseteq \alpha(L_2) \Leftrightarrow &\quad \text{[by GC]} \\
U \subseteq \gamma(\alpha(L_2)) \Leftrightarrow &\quad \text{[by } \gamma(\alpha(L_2)) = L_2\text{]} \\
U \subseteq L_2 \Leftrightarrow &\quad \text{[by definition of } \mathrm{ctx}_{\mathcal{A}}(u)\text{]} \\
\forall u \in U, \mathrm{ctx}_{\mathcal{A}}(u) \cap (I \times F) \neq \varnothing \Leftrightarrow &\quad \text{[since } Y_0 = \alpha(U) = \lfloor \{\mathrm{ctx}_{\mathcal{A}}(u) \mid u \in U\} \rfloor\text{]} \\
\forall R \in Y_0, R \cap I \neq \varnothing &\ .
\end{aligned}$$

Hence, Theorem 7.10 entails that Algorithm CFGIncS decides $\mathcal{L}(\mathcal{G}) \subseteq \mathcal{L}(\mathcal{A})$. ◻

The resulting algorithm CFGIncS shares some features with two previous related works. On the one hand, it is related to the work of Hofmann and Chen [2014] which defines an abstract interpretation-based language inclusion decision procedure similar to ours. Even though Hofmann and Chen's algorithm and ours both manipulate sets of pairs of states of an automaton, their abstraction is based on equivalence relations rather than quasiorders. Since quasiorders are strictly more general than equivalences, our framework can be instantiated to a larger class of abstractions, most importantly coarser ones. Finally, it is worth pointing out that Hofmann and Chen [2014] approach aims at including languages of finite and also infinite words.

A second related work is that of Holík and Meyer [2015] who define an antichain-based algorithm manipulating sets of pairs of states. However, they tackle the inclusion problem $\mathcal{L}(\mathcal{G}) \subseteq \mathcal{L}(\mathcal{A})$, where $\mathcal{G}$ is a grammar and $\mathcal{A}$ and automaton, by rephrasing it as a data flow analysis problem over a relational domain. In this scenario, the solution of the problem requires the computation of a least fixpoint on the relational domain, followed by an inclusion check between sets of relations. Then, they use the "antichain principle" to improve the performance of the fixpoint computation and, finally, they move from manipulating relations to manipulating pairs of states. As a result, Holík and Meyer [2015] devise an antichain algorithm for checking the inclusion $\mathcal{L}(\mathcal{G}) \subseteq \mathcal{L}(\mathcal{A})$.

By contrast to these two approaches, our design technique is direct and systematic, since the algorithm CFGIncS is derived from the known Myhill quasiorder. We believe that our approach reveals the relationship between the original antichain algorithm by De Wulf et al. [2006] for regular languages and the one by Holík and Meyer [2015] for context-free languages, which is the relation between our algorithms FAIncS and CFGIncS. Specifically, we have shown that these two algorithms are conceptually identical and just differ in the well-quasiorder used to define the abstract domain where computations take place.

8 AN EQUIVALENT GREATEST FIXPOINT ALGORITHM

Let us assume that $g\colon C \to C$ is a monotonic function on a complete lattice $\langle C, \leq, \vee, \wedge \rangle$ which admits its unique right-adjoint $\widetilde{g}\colon C \to C$, i.e., $\forall c, c' \in C$, $g(c) \leq c' \Leftrightarrow c \leq \widetilde{g}(c')$ holds. Then, Cousot [2000, Theorem 4] shows that the following equivalence holds: for all $c, c' \in C$,

$$\mathrm{lfp}(\lambda x.\, c \vee g(x)) \leq c' \;\Leftrightarrow\; c \leq \mathrm{gfp}(\lambda y.\, c' \wedge \widetilde{g}(y)) \ . \tag{24}$$

This property has been used in [Cousot 2000] to derive equivalent least/greatest fixpoint-based invariance proof methods for programs. In the following, we use (24) to derive an algorithm for deciding the inclusion $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$, which relies on the computation of a greatest fixpoint rather than a least fixpoint. This can be achieved by exploiting the following simple observation, which defines an adjunction between concatenation and quotients of sets of words.

Lemma 8.1. *For all $X, Y \in \wp(\Sigma^*)$ and $w \in \Sigma^*$, $wY \subseteq Z \Leftrightarrow Y \subseteq w^{-1}Z$ and $Yw \subseteq Z \Leftrightarrow Y \subseteq Zw^{-1}$.*

Proof. By definition, for all $u \in \Sigma^*$, $u \in w^{-1}Z$ iff $wu \in Z$. Hence, $Y \subseteq w^{-1}Z \Leftrightarrow \forall u \in Y,\ wu \in Z \Leftrightarrow wY \subseteq Z$. Symmetrically, $Yw \subseteq Z \Leftrightarrow Y \subseteq Zw^{-1}$ holds. ◻

Given a FA $\mathcal{A} = \langle Q, \delta, I, F, \Sigma\rangle$, we define the function $\widetilde{\mathrm{Pre}}_{\mathcal{A}} : \wp(\Sigma^*)^{|Q|} \to \wp(\Sigma^*)^{|Q|}$ on Q-indexed vectors of sets of words as follows:

$$\widetilde{\mathrm{Pre}}_{\mathcal{A}}(\langle X_q\rangle_{q\in Q}) \triangleq \langle \textstyle\bigcap_{a\in\Sigma, q'\in\delta(q,a)} a^{-1}X_q\rangle_{q'\in Q} \enspace ,$$

where, as usual, $\bigcap \varnothing = \Sigma^*$. It turns out that $\widetilde{\mathrm{Pre}}_{\mathcal{A}}$ is the usual weakest liberal precondition which is right-adjoint of $\mathrm{Pre}_{\mathcal{A}}$.

Lemma 8.2. *For all* $\vec{X}, \vec{Y} \in \wp(\Sigma^*)^{|Q|}$, $\mathrm{Pre}_{\mathcal{A}}(\vec{X}) \subseteq \vec{Y} \Leftrightarrow \vec{X} \subseteq \widetilde{\mathrm{Pre}}_{\mathcal{A}}(\vec{Y})$.

Proof.

$$\begin{array}{rl}
\mathrm{Pre}_{\mathcal{A}}(\langle X_q\rangle_{q\in Q}) \subseteq \langle Y_q\rangle_{q\in Q} \Leftrightarrow & [\text{by definition of } \mathrm{Pre}_{\mathcal{A}}] \\
\forall q \in Q, \bigcup_{q\xrightarrow{a}q'} aX_{q'} \subseteq Y_q \Leftrightarrow & [\text{by set theory}] \\
\forall q, q' \in Q, q \xrightarrow{a} q' \Rightarrow aX_{q'} \subseteq Y_q \Leftrightarrow & [\text{by Lemma 8.1}] \\
\forall q, q' \in Q, q \xrightarrow{a} q' \Rightarrow X_{q'} \subseteq a^{-1}Y_q \Leftrightarrow & [\text{by set theory}] \\
\forall q' \in Q, X_{q'} \subseteq \bigcap_{q\xrightarrow{a}q'} a^{-1}Y_q \Leftrightarrow & [\text{by definition of } \widetilde{\mathrm{Pre}}_{\mathcal{A}}] \\
\langle X_q\rangle_{q\in Q} \subseteq \widetilde{\mathrm{Pre}}_{\mathcal{A}}(\langle Y_q\rangle_{q\in Q}) \enspace . &
\end{array}$$

□

Hence, from equivalences (10) and (24), we obtain that for all FAs $\mathcal{A}_1$ and $L_2 \in \wp(\Sigma^*)$:

$$\mathcal{L}(\mathcal{A}_1) \subseteq L_2 \iff \vec{\epsilon}^{F_1} \subseteq \mathrm{gfp}(\lambda\vec{X}.\, \vec{L_2}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\vec{X})) \enspace . \tag{25}$$

The following algorithm `FAIncGfp` decides the inclusion $\mathcal{L}(\mathcal{A}_1) \subseteq L_2$, when L_2 is regular, by implementing the greatest fixpoint computation in equivalence (25).

`FAIncGfp`: Greatest fixpoint algorithm for $\mathcal{L}(\mathcal{A}_1) \subseteq L_2$

Data: FA $\mathcal{A}_1 = \langle Q_1, \delta_1, I_1, F_1, \Sigma\rangle$; regular language L_2.

$\langle Y_q\rangle_{q\in Q} := \textsc{Kleene}(\supseteq, \lambda\vec{X}.\, \vec{L_2}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\vec{X}), \vec{\Sigma^*})$;
forall $q \in F_1$ **do**
  if $\epsilon \notin Y_q$ **then return** *false*;
return *true*;

The intuition behind Algorithm `FAIncGfp` is that

$$\mathcal{L}(\mathcal{A}_1) \subseteq L_2 \Leftrightarrow \forall w \in \mathcal{L}(\mathcal{A}_1),\ (\epsilon \in w^{-1}L_2 \Leftrightarrow \epsilon \in \textstyle\bigcap_{w\in\mathcal{L}(\mathcal{A}_1)} w^{-1}L_2) \enspace .$$

Therefore, `FAIncGfp` computes the set $\bigcap\{w^{-1}L_2 \mid w \in \mathcal{L}(\mathcal{A}_1)\}$ by using the automaton $\mathcal{A}_1$ and by considering prefixes of $\mathcal{L}(\mathcal{A}_1)$ of increasing lengths. This means that after n iterations of Kleene, the algorithm `FAIncGfp` has computed

$$\textstyle\bigcap\{w^{-1}L_2 \mid wu \in \mathcal{L}(\mathcal{A}_1),\ |w| \leq n,\ q_0 \in I_1,\ q_0 \overset{w}{\leadsto} q\}$$

for every state $q \in Q_1$. The regularity of L_2 and the property of regular languages of being closed under intersections and quotients entail that each Kleene iterate of $\textsc{Kleene}(\supseteq, \lambda\vec{X}.\, \vec{L_2}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\vec{X}), \vec{\Sigma^*})$ is a (computable) regular language. To the best of our knowledge, this gfp-based language inclusion algorithm `FAIncGfp` has never been described in the literature before.

Next, we discharge the fundamental assumption guaranteeing the correctness of this algorithm `FAIncGfp`: the Kleene iterates computed by `FAIncGfp` are finitely many. To achieve this, we consider an abstract version of the greatest fixpoint computation exploiting a closure operator which ensures that the abstract Kleene iterates are finitely many. This closure operator $\rho_{\leq_{\mathcal{A}_2}}$ will be defined by using an ordering relation $\leq_{\mathcal{A}_2}$ induced by a FA $\mathcal{A}_2$ such that $L_2 = \mathcal{L}(\mathcal{A}_2)$ and will be shown to be *forward complete* for the function $\lambda\vec{X}.\,\vec{L_2}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\vec{X})$ used by `FAIncGfp`.

Forward completeness of abstract interpretations [Giacobazzi and Quintarelli 2001], also called exactness [Miné 2017, Definition 2.15], is different from and orthogonal to backward completeness introduced in Section 3 and crucially used throughout Sections 4–7. In particular, a remarkable consequence of exploiting a forward complete abstraction is that the Kleene iterates of the concrete and abstract greatest fixpoint computations coincide. The intuition here is that this forward complete closure $\rho_{\leq_{\mathcal{A}_2}}$ allows us to establish that all the Kleene iterates of $\lambda\vec{X}.\,\vec{L_2}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\vec{X})$ belong to the image of the closure $\rho_{\leq_{\mathcal{A}_2}}$, more precisely that every Kleene iterate is a language which is upward closed for $\leq_{\mathcal{A}_2}$. Interestingly, a similar phenomenon occurs in well-structured transition systems [Abdulla et al. 1996; Finkel and Schnoebelen 2001].

Let us now describe in detail this abstraction. A closure $\rho \in \mathrm{uco}(C)$ on a concrete domain C is forward complete for a monotonic function $f : C \to C$ if $\rho f \rho = f\rho$ holds. The intuition here is that forward completeness means that no loss of precision is accumulated when the output of a computation of $f\rho$ is approximated by ρ, or, equivalently, the concrete function f maps abstract elements of ρ into abstract elements of ρ. Dually to the case of backward completeness, forward completeness implies that $\mathrm{gfp}(f) = \mathrm{gfp}(f\rho) = \mathrm{gfp}(\rho f\rho)$ holds, when these greatest fixpoints exist (this is the case, e.g., when C is a complete lattice). When the function $f : C \to C$ admits the right-adjoint $\widetilde{f} : C \to C$, i.e., $f(c) \leq c' \Leftrightarrow c \leq \widetilde{f}(c')$ holds, it turns out that forward and backward completeness are related by the following duality [Giacobazzi and Quintarelli 2001, Corollary 1]:

$$\rho \text{ is backward complete for } f \text{ iff } \rho \text{ is forward complete for } \widetilde{f}. \tag{26}$$

Thus, by (26), in the following result instead of assuming the hypotheses implying that a closure ρ is forward complete for the right-adjoint $\widetilde{\mathrm{Pre}}_{\mathcal{A}_1}$, we state some hypotheses which guarantee that ρ is backward complete for its left-adjoint, that, by Lemma 8.2, is $\mathrm{Pre}_{\mathcal{A}_1}$.

Theorem 8.3. *Let $\mathcal{A}_1 = \langle Q_1, \delta_1, I_1, F_1, \Sigma\rangle$ be a FA , L_2 be a regular language and $\rho \in \mathrm{uco}(\wp(\Sigma^*))$. Let us assume that:*
(1) $\rho(L_2) = L_2$;
(2) ρ is backward complete for $\lambda X.\, aX$ for all $a \in \Sigma$.
Then, $\mathcal{L}(\mathcal{A}_1) \subseteq L_2$ iff $\vec{\epsilon}^{F_1} \subseteq \mathrm{gfp}(\lambda\vec{X}.\rho(\vec{L_2}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\rho(\vec{X}))))$. Moreover, the Kleene iterates of $\lambda\vec{X}.\,\rho(\vec{L_2}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\rho(\vec{X})))$ and $\lambda\vec{X}.\,\vec{L_2}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\vec{X})$ from the initial value $\vec{\Sigma^*}$ coincide in lockstep.

Proof. Theorem 4.3 shows that if ρ is backward complete for $\lambda X.\, aX$, for every $a \in \Sigma$, then it is backward complete for $\mathrm{Pre}_{\mathcal{A}_1}$. Thus, by (26), ρ is forward complete for $\widetilde{\mathrm{Pre}}_{\mathcal{A}_1}$. Then, it turns out that ρ is forward complete for $\lambda\vec{X}.\,\vec{L_2}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\vec{X})$, because:

$$\begin{aligned}
\rho(\vec{L_2}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\rho(\vec{X}))) = \quad & [\text{by forward completeness for } \widetilde{\mathrm{Pre}}_{\mathcal{A}_1} \text{ and } \rho(L_2) = L_2]\\
\rho(\rho(\vec{L_2}^{I_1}) \cap \rho(\widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\rho(\vec{X})))) = \quad & [\text{by (3)}]\\
\rho(\vec{L_2}^{I_1}) \cap \rho(\widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\rho(\vec{X}))) = \quad & [\text{by forward completeness for } \widetilde{\mathrm{Pre}}_{\mathcal{A}_1} \text{ and } \rho(L_2) = L_2]\\
\vec{L_2}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\rho(\vec{X})) \quad . &
\end{aligned}$$

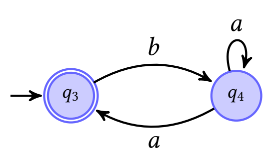
Since, by forward completeness, $\mathrm{gfp}(\lambda\vec{X}.\,\vec{L_2}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\vec{X})) = \mathrm{gfp}(\lambda\vec{X}.\,\rho(\vec{L_2}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\rho(\vec{X}))))$, by equivalence (25), we conclude that $\mathcal{L}(\mathcal{A}_1) \subseteq L_2$ iff $\vec{\epsilon}^{F_1} \subseteq \mathrm{gfp}(\lambda\vec{X}.\,\rho(\vec{L_2}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\rho(\vec{X}))))$.

Finally, we observe that the Kleene iterates of $\lambda\overrightarrow{X}.\, \overrightarrow{L_2}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\overrightarrow{X})$ and $\lambda\overrightarrow{X}.\, \rho(\overrightarrow{L_2}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\rho(\overrightarrow{X})))$ starting from $\overrightarrow{\Sigma^*}$ coincide in lockstep since $\rho(\overrightarrow{L_2}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\rho(\overrightarrow{X}))) = \overrightarrow{L_2}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\rho(\overrightarrow{X}))$ and $\rho(\overrightarrow{L_2}^{I_1}) = \overrightarrow{L_2}^{I_1}$. □

We can now establish that the iterates of $\mathrm{K{\scriptsize LEENE}}(\supseteq, \lambda\overrightarrow{X}.\, \overrightarrow{L_2}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\overrightarrow{X}), \overrightarrow{\Sigma^*})$ are finitely many. Let $L_2 = \mathcal{L}(\mathcal{A}_2)$, for some FA $\mathcal{A}_2$, and consider the corresponding left state-based quasiorder $\leq^l_{\mathcal{A}_2}$ on Σ^* as defined by (16). By Lemma 5.8, $\leq^l_{\mathcal{A}_2}$ is a left L_2-consistent wqo. Furthermore, since Q_2 is finite, we have that both $\leq^l_{\mathcal{A}_2}$ and $(\leq^l_{\mathcal{A}_2})^{-1}$ are wqos, so that, in turn, $\langle \rho_{\leq^l_{\mathcal{A}_2}}(\wp(\Sigma^*)), \subseteq\rangle$ is a poset which is both ACC and DCC. In particular, the definition of $\leq^l_{\mathcal{A}_2}$ implies that every chain in $\langle \rho_{\leq^l_{\mathcal{A}_2}}(\wp(\Sigma^*)), \subseteq\rangle$ has at most $2^{|Q_2|}$ elements, so that if we compute $2^{|Q_2|}$ Kleene iterates then we surely converge to the greatest fixpoint. Moreover, as a consequence of the DCC property, we have that $\mathrm{K{\scriptsize LEENE}}(\supseteq, \lambda\overrightarrow{X}.\, \rho_{\leq_{\mathcal{A}_2}}(\overrightarrow{L_2}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\rho_{\leq_{\mathcal{A}_2}}(\overrightarrow{X}))), \overrightarrow{\Sigma^*})$ always terminates, thus implying that $\mathrm{K{\scriptsize LEENE}}(\supseteq, \lambda\overrightarrow{X}.\, \overrightarrow{L_2}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\overrightarrow{X}), \overrightarrow{\Sigma^*})$ terminates as well, because their iterates go in lockstep as stated by Theorem 8.3. We have therefore shown the correctness of `FAIncGfp`.

COROLLARY 8.4. *The algorithm* `FAIncGfp` *decides the inclusion* $\mathcal{L}(\mathcal{A}_1) \subseteq L_2$

Example 8.5. Let us illustrate the greatest fixpoint algorithm `FAIncGfp` on the inclusion check $L(\mathcal{B}) \subseteq L(\mathcal{A})$ where $\mathcal{A}$ is the FA in Fig. 1 and $\mathcal{B}$ is the following FA:



By Corollary 8.4, the Kleene iterates of $\lambda\overrightarrow{Y}.\, \overrightarrow{\boldsymbol{L(\mathcal{A})}}^{\{q_3\}} \cap \widetilde{\mathrm{Pre}}_{\mathcal{B}}(\overrightarrow{Y})$ are guaranteed to converge in finitely many steps. We have that

$$\overrightarrow{\boldsymbol{L(\mathcal{A})}}^{\{q_3\}} \cap \widetilde{\mathrm{Pre}}_{\mathcal{B}}(\langle Y_3, Y_4\rangle) = \langle L(\mathcal{A}) \cap a^{-1}Y_4,\ b^{-1}Y_3 \cap a^{-1}Y_4\rangle \ .$$

Then, the Kleene iterates are as follows (we automatically checked them by the FAdo tool [Almeida et al. 2009]):

$$\begin{aligned}
Y_3^{(0)} &= \Sigma^* & Y_4^{(0)} &= \Sigma^* \\
Y_3^{(1)} &= L(\mathcal{A}) \cap a^{-1}\Sigma^* = L(\mathcal{A}) & Y_4^{(1)} &= b^{-1}\Sigma^* \cap a^{-1}\Sigma^* = \Sigma^* \\
Y_3^{(2)} &= L(\mathcal{A}) \cap a^{-1}\Sigma^* = L(\mathcal{A}) & Y_4^{(2)} &= b^{-1}L(\mathcal{A}) \cap a^{-1}\Sigma^* = b^{-1}L(\mathcal{A}) = (b^*a)^+ \\
Y_3^{(3)} &= L(\mathcal{A}) \cap a^{-1}(b^*a)^+ = L(\mathcal{A}) & Y_4^{(3)} &= b^{-1}L(\mathcal{A}) \cap a^{-1}(b^*a)^+ = (b^*a)^+
\end{aligned}$$

Thus, $\mathrm{K{\scriptsize LEENE}}$ outputs the vector $\langle Y_3, Y_4\rangle = \langle L(\mathcal{A}), (b^*a)^+\rangle$. Since $\epsilon \in L(\mathcal{A})$, `FAIncGfp` concludes that $L(\mathcal{B}) \subseteq L(\mathcal{A})$ holds. ◇

Finally, it is worth citing that Fiedor et al. [2019] put forward an algorithm for deciding WS1S formulae which relies on the same lfp computation used in `FAIncS`. Then, they derive a dual gfp computation by relying on Park's duality [Park 1969]: $\mathrm{lfp}(\lambda X.\, f(X)) = (\mathrm{gfp}(\lambda X.\, (f(X^c))^c))^c$. Their approach differs from ours since we use the equivalence (24) to compute a gfp, different from the lfp, which still allows us to decide the inclusion problem. Furthermore, their algorithm decides whether a given automaton accepts ϵ and it is not clear how their algorithm could be extended for deciding language inclusion.

9 FUTURE WORK

We believe that this work only scratched the surface of the use of well-quasiorders on words for solving language inclusion problems. In particular, our approach based on complete abstract interpretations allowed us to systematically derive well-known algorithms, such as the antichain algorithms by De Wulf et al. [2006], as well as novel algorithms, such as `FAIncGfp`, for deciding the inclusion of regular languages.

Future directions include leveraging well-quasiorders arising from languages of infinite words to shed new light on the inclusion problem between ω-languages [Arnold 1985]. Our results could also be extended to inclusion of tree languages by relying on the extensions of Myhill-Nerode theorems for tree languages [Kozen 1992]. Another interesting topic for future work is the enhancement of quasiorders using simulation relations. Even though we already showed in this paper that simulations can be used to refine our language inclusion algorithms, we are not on par with the thoughtful use of simulation relations made by Abdulla et al. [2010] and Bonchi and Pous [2013]. Finally, let us mention that the correspondence between least and greatest fixpoint-based inclusion checks assuming complete abstractions was studied by Bonchi et al. [2018] with the aim of formally connecting sound up-to techniques and complete abstract interpretations. Further possible developments include the study of our abstract interpretation-based algorithms for language inclusion from the viewpoint of sound up-to techniques.

ACKNOWLEDGMENTS

We would like to thank Patrick Cousot and Roberto Giacobazzi for the initial discussion of this research subject. Pierre Ganty completed this work with the support of the Spanish Ministry of Economy and Competitiveness project No. PGC2018-102210-B-I00, the Madrid Regional Government project No. S2018/TCS-4339 and the Ramón y Cajal fellowship RYC-2016-20281. The work of Francesco Ranzato has been partially funded by the University of Padova, under the SID2018 project "Analysis of STatic Analyses (ASTA)", by the Italian Ministry of Research MIUR, under the PRIN2017 project no. 201784YSZ5 "AnalysiS of PRogram Analyses (ASPRA)", and by Facebook inc. under the Probability and Programming Research Award "Adversarial Machine Learning by Morphological Abstract Interpretation".

REFERENCES

Parosh Aziz Abdulla, Karlis Cerans, Bengt Jonsson, and Yih-Kuen Tsay. 1996. General decidability theorems for infinite-state systems. In *Proc. of the 11th Annual IEEE Symp. on Logic in Computer Science (LICS'96)*. IEEE Computer Society, Washington, DC, USA, 313–321.

Parosh Aziz Abdulla, Yu-Fang Chen, Lukáš Holík, Richard Mayr, and Tomáš Vojnar. 2010. When Simulation Meets Antichains. In *Proceedings of the 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'10)*. Springer Berlin Heidelberg, 158–174. https://doi.org/10.1007/978-3-642-12002-2_14

André Almeida, Marco Almeida, José Alves, Nelma Moreira, and Rogério Reis. 2009. FAdo and GUItar: Tools for Automata Manipulation and Visualization. In *Implementation and Application of Automata*. Springer Berlin Heidelberg, 65–74. https://doi.org/10.1007/978-3-642-02979-0_10

André Arnold. 1985. A Syntactic Congruence for Rational ω-Languages. *Theoretical Computer Science* 39 (Jan. 1985), 333–335. https://doi.org/10.1016/0304-3975(85)90148-3

Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking*. The MIT Press.

Friedrich L. Bauer and Jürgen Eickel. 1976. *Compiler Construction, An Advanced Course, 2nd Ed.* Springer-Verlag, Berlin, Heidelberg.

Filippo Bonchi, Pierre Ganty, Roberto Giacobazzi, and Dusko Pavlovic. 2018. Sound up-to techniques and Complete abstract domains. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'18)*. ACM Press. https://doi.org/10.1145/3209108.3209169

Filippo Bonchi and Damien Pous. 2013. Checking NFA Equivalence with Bisimulations Up to Congruence. In *Proceedings of the 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'13)*. ACM Press, 457–468. https://doi.org/10.1145/2429069.2429124

Noam Chomsky. 1959. On Certain Formal Properties of Grammars. *Information and Control* 2, 2 (1959), 137–167.

Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem. 2018. *Handbook of Model Checking* (1st ed.). Springer Publishing Company, Incorporated.

Patrick Cousot. 2000. Partial Completeness of Abstract Fixpoint Checking. In *Proceedings of the 4th International Symposium on Abstraction, Reformulation, and Approximation (SARA'02)*. Springer-Verlag, 1–25. https://doi.org/10.1007/3-540-44914-0_1

Patrick Cousot and Radhia Cousot. 1977. 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 (POPL'77)*. ACM Press, 238–252. http://doi.acm.org/10.1145/512950.512973

Patrick Cousot and Radhia Cousot. 1979. Systematic design of program analysis frameworks. In *Proceedings of the 6th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL'79)*. ACM, New York, NY, USA, 269–282. https://doi.org/10.1145/567752.567778

Aldo de Luca and Stefano Varricchio. 1994. Well quasi-orders and regular languages. *Acta Informatica* 31, 6 (1994), 539–557. https://doi.org/10.1007/BF01213206

Aldo de Luca and Stefano Varricchio. 2011. *Finiteness and Regularity in Semigroups and Formal Languages*. Springer. https://doi.org/10.1007/978-3-642-59849-4

Martin De Wulf, Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. 2006. Antichains: A New Algorithm for Checking Universality of Finite Automata. In *Proceedings of the 18th International Conference on Computer Aided Verification (CAV'06)*. Springer-Verlag, 17–30. http://dx.doi.org/10.1007/11817963_5

Andrzej Ehrenfeucht, David Haussler, and Grzegorz Rozenberg. 1983. On regularity of context-free languages. *Theoretical Computer Science* 27, 3 (1983), 311–332. https://doi.org/10.1016/0304-3975(82)90124-4

Tomáš Fiedor, Lukáš Holík, Ondřej Lengál, and Tomáš Vojnar. 2019. Nested antichains for WS1S. *Acta Informatica* 56, 3 (2019), 205–228.

Alain Finkel and Philippe Schnoebelen. 2001. Well-structured transition systems everywhere! *Theoretical Computer Science* 256, 1-2 (2001), 63–92. https://doi.org/10.1016/s0304-3975(00)00102-x

Pierre Ganty, Francesco Ranzato, and Pedro Valero. 2019. Language Inclusion Algorithms as Complete Abstract Interpretations. In *Proc. of the 26th International Static Analysis Symposium (SAS'19), LNCS vol. 11822*, Bor-Yuh Evan Chang (Ed.). Springer, 140–161.

Roberto Giacobazzi and Elisa Quintarelli. 2001. Incompleteness, Counterexamples, and Refinements in Abstract Model-Checking. In *Proceedings of the 8th Static Analysis Symposium (SAS'01), LNCS vol. 2126*. Springer, 356–373. https://doi.org/10.1007/3-540-47764-0_20

Roberto Giacobazzi, Francesco Ranzato, and Francesca Scozzari. 2000. Making Abstract Interpretations Complete. *J. ACM* 47, 2 (2000), 361–416. https://doi.org/10.1145/333979.333989

Seymour Ginsburg and H. Gordon Rice. 1962. Two Families of Languages Related to ALGOL. *J. ACM* 9, 3 (July 1962), 350–371. https://doi.org/10.1145/321127.321132

Piotr Hofman, Richard Mayr, and Patrick Totzke. 2013. Decidability of Weak Simulation on One-Counter Nets. In *Proceedings of the 28th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'13)*. IEEE Computer Society, 203–212. https://doi.org/10.1109/LICS.2013.26

Piotr Hofman and Patrick Totzke. 2018. Trace inclusion for one-counter nets revisited. *Theoretical Computer Science* 735 (July 2018), 50–63. https://doi.org/10.1016/j.tcs.2017.05.009

Martin Hofmann and Wei Chen. 2014. Abstract interpretation from Büchi automata. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL'14) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'14)*. ACM Press. https://doi.org/10.1145/2603088.2603127

Lukáš Holík and Roland Meyer. 2015. Antichains for the Verification of Recursive Programs. In *Networked Systems*. Springer International Publishing, 322–336. https://doi.org/10.1007/978-3-319-26850-7_22

John E. Hopcroft and Jeff D. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing Company.

Harry B. Hunt, Daniel J. Rosenkrantz, and Thomas G. Szymanski. 1976. On the equivalence, containment, and covering problems for the regular and context-free languages. *J. Comput. System Sci.* 12, 2 (1976), 222 – 268. https://doi.org/10.1016/S0022-0000(76)80038-4

Petr Jančar, Javier Esparza, and Faron Moller. 1999. Petri Nets and Regular Processes. *J. Comput. System Sci.* 59, 3 (1999), 476–503. https://doi.org/10.1006/jcss.1999.1643

Dexter Kozen. 1992. On the Myhill-Nerode Theorem for Trees. *Bulletin of the EATCS* 47 (1992), 170–173.

Antoine Miné. 2017. Tutorial on Static Inference of Numeric Invariants by Abstract Interpretation. *Foundations and Trends in Programming Languages* 4, 3-4 (2017), 120–372. https://doi.org/10.1561/2500000034

David Park. 1969. Fixpoint induction and proofs of program properties. *Machine Intelligence* 5 (1969).

Francesco Ranzato. 2013. Complete Abstractions Everywhere. In *Proceedings of the 14th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'13)*, Vol. 7737. LNCS Springer, 15–26. https://doi.org/10.1007/978-3-642-35873-9_3

Xavier Rival and Kwangkeun Yi. 2020. *Introduction to Static Analysis: An Abstract Interpretation Perspective*. The MIT Press.

Jacques Sakarovitch. 2009. *Elements of Automata Theory*. Cambridge University Press. https://doi.org/10.1017/CBO9781139195218

Marcel Paul Schützenberger. 1963. On Context-Free Languages and Push-Down Automata. *Information and Control* 6, 3 (1963), 246–264. https://doi.org/10.1016/S0019-9958(63)90306-1

William M. Waite and Gerhard Goos. 1984. *Compiler Construction*. Springer-Verlag, New York, USA.