Language Inclusion Algorithms as Complete Abstract Interpretations

Pierre Ganty[1], Francesco Ranzato[2], and Pedro Valero[1,3](✉)

[1] IMDEA Software Institute, Madrid, Spain
{pierre.ganty,pedro.valero}@imdea.org
[2] Dipartimento di Matematica, University of Padova, Padova, Italy
francesco.ranzato@unipd.it
[3] Universidad Politécnica de Madrid, Madrid, Spain

Abstract. We study the language inclusion problem $L_1 \subseteq L_2$ where L_1 is regular. Our approach relies on abstract interpretation and checks whether an overapproximating abstraction of L_1, obtained by successively overapproximating 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). Such overapproximating abstraction function on languages can be defined using quasiorder relations on words where the abstraction gives the language of all words "greater than or equal to" a given input word for that quasiorder. We put forward a range of 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. 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 greatest fixpoint language inclusion check which relies on quotients of languages and, to the best of our knowledge, was not previously known.

1 Introduction

Language inclusion is a fundamental and classical problem which consists in deciding, given two languages L_1 and L_2, whether $L_1 \subseteq L_2$ holds. We consider languages of finite words over a finite alphabet Σ.

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 [6,7] 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 [18], who used abstract interpretation to decide language inclusion between languages of infinite words.

Assuming that L_1 is specified as least fixpoint of an equation system on $\wp(\Sigma^*)$, an approximation of L_1 is obtained by applying an overapproximating

B.-Y. E. Chang (Ed.): SAS 2019, LNCS 11822, pp. 140–161, 2019.
https://doi.org/10.1007/978-3-030-32304-2_8

abstraction function for sets of words $\rho : \wp(\Sigma^*) \to \wp(\Sigma^*)$ at each step of the Kleene iterates converging to the least fixpoint. This ρ is an upper closure operator which is used in standard abstract interpretation for approximating an input language by adding words (possibly none) to it. This abstract interpretation-based approach provides an abstract inclusion check $\rho(L_1) \subseteq L_2$ which is always sound by construction. We then give conditions on ρ which ensure a *complete* abstract inclusion check, namely, the answer to $\rho(L_1) \subseteq L_2$ is always exact (no "false alarms" in abstract interpretation terminology): (i) $\rho(L_2) = L_2$; (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 [6,16,23]. This approach leads us to design in Sect. 4 an algorithmic framework for language inclusion problems which is parameterized by an underlying language abstraction (cf. Theorem 4.5).

We then focus on overapproximating abstractions ρ which are induced by a quasiorder relation $\leqslant$ on words in Σ^*. 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 having a complete abstract inclusion check in terms of the quasiorder $\leqslant$. Termination, which corresponds to having finitely many Kleene iterates in the fixpoint computations, is guaranteed by requiring that the relation $\leqslant$ is a well-quasiorder.

We define quasiorders satisfying the above conditions which are directly derived from the standard Nerode equivalence relations on words. These quasiorders have been first investigated by Ehrenfeucht et al. [11] and have been later generalized and extended by de Luca and Varricchio [8,9]. In particular, drawing from a result by de Luca and Varricchio [8], we show that the language abstractions induced by the Nerode quasiorders are the most general ones which fit in our algorithmic framework for checking language inclusion. While these quasiorder abstractions do not depend on some language representation (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 for the inclusion problem $L_1 \subseteq L_2$ when: (i) both L_1 and L_2 are regular and (ii) L_1 is regular and L_2 is the trace language of a one-counter net.

These decision procedures that we systematically derive here 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 the automaton-based representation of L_2, it turns out that we obtain the well-known "antichain algorithm" by De Wulf et al. [10]. Also, by including a simulation relation in the definition of the well-quasiorder we derive a decision procedure that partially matches the inclusion algorithm by Abdulla et al. [2], hence also that by Bonchi and Pous [4]. For the case in which L_1 is regular and L_2 is the set of traces of a one-counter net we derive an alternative proof for the decidability of the language inclusion problem [19].

Finally, we leverage a standard duality result in abstract fixpoint checking [5] and put forward a greatest fixpoint approach (instead of the above least fixpoint-based procedure) for the case where both L_1 and L_2 are regular languages. In this case, we exploit the properties of the overapproximating abstraction induced by the quasiorder in order to show that the Kleene iterates of this greatest fixpoint computation are finitely many. Interestingly, the Kleene iterates of the greatest fixpoint are finitely many whether you apply the overapproximating abstraction or not, which we show relying on forward complete abstract interpretations [15]. An extended version of this paper is available on-line [14].

2 Background

Order Theory Basics. If X is a subset of some universe set U then X^c denotes the complement of X with respect to U, and U is implicitly given by the context.

$\langle D, \leqslant\rangle$ is a *quasiordered set* (qoset) when $\leqslant$ is a quasiorder relation on D, i.e. a reflexive and transitive binary relation. 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 of a poset is *directed* if it is nonempty and every pair of elements has an upper bound in it. 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 (so that binary lubs are enough). A poset is a *complete lattice* if it has the lub of all its arbitrary (possibly empty) subsets (so that it also has the greatest lower bound (glb) of all its arbitrary subsets).

A qoset $\langle D, \leqslant\rangle$ is a *well-quasiordered set* (wqoset) 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 $x_i \leqslant x_j$. For every qoset $\langle D, \leqslant\rangle$ with $X, Y \subseteq D$, we define the following relation:

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

A *minor* of a subset $X \subseteq D$, denoted by $\lfloor X \rfloor$, is a subset of minimal elements of X w.r.t. $\leqslant$, i.e. $\lfloor X \rfloor \triangleq \{x \in X \mid \forall y \in X, y \leqslant x \text{ implies } y = x\}$. The *minor* of X satisfies the following properties: (i) $X \sqsubseteq \lfloor X \rfloor$ and (ii) $\lfloor X \rfloor$ is an *antichain*, that is, $x_1 \leqslant x_2$ for no distinct $x_1, x_2 \in \lfloor X \rfloor$. Let us recall that every subset of a wqoset $\langle D, \leqslant\rangle$ has at least one minor set, all minor sets are finite and if $\langle D, \leqslant\rangle$ is additionally a poset then there exists exactly one minor set. We denote the set of antichains of a qoset $\langle D, \leqslant\rangle$ by $\mathrm{AC}_{\langle D, \leqslant\rangle} \triangleq \{X \subseteq D \mid X \text{ is an antichain}\}$. It turns out that $\langle \mathrm{AC}_{\langle D, \leqslant\rangle}, \sqsubseteq\rangle$ is a qoset, it is ACC if $\langle D, \leqslant\rangle$ is a wqoset and it is a poset if $\langle D, \leqslant\rangle$ is a poset.

Kleene Iterates. Let $\langle X, \leqslant\rangle$ be a qoset, $f : X \to X$ be a function and $b \in X$. Then, the trace of values of the variable $x \in X$ computed by the following iterative procedure:

$$\texttt{Kleene}(f, b) \triangleq \begin{cases} x := b; \\ \textbf{while } f(x) \neq x \textbf{ do } x := f(x); \\ \textbf{return } x; \end{cases}$$

provides the possibly infinite sequence of so-called Kleene iterates of the function f starting from the basis b. When $\langle X, \leqslant \rangle$ is a ACC (resp. DCC) CPO, $b \leqslant f(b)$ (resp. $f(b) \leqslant b$) and f is monotonic then $\texttt{Kleene}(f, b)$ terminates and returns the least (resp. greatest) fixpoint of the function f which is greater (resp. less) than or equal to b.

Let us also recall that given a monotonic function $f\colon C \to C$ on a complete lattice C, its least and greatest fixpoints always exist, and we denote them, resp., by $\mathrm{lfp}(f)$ and $\mathrm{gfp}(f)$.

For the sake of clarity, we overload the notation and use the same symbol for an operator/relation and its componentwise (i.e. pointwise) extension on product domains. A vector $\overrightarrow{\boldsymbol{Y}}$ in some product domain $D^{|S|}$ might be also denoted by $\langle Y_i \rangle_{i \in S}$ and $\overrightarrow{\boldsymbol{Y}}_q$ denotes its component Y_q.

Language Theory Basics. Let Σ be an *alphabet* (that is, a finite nonempty set of symbols). Words are finite sequences of symbols where ϵ denote the empty sequence. Languages are sets of words where Σ^* is the set of all words. Concatenation in Σ^* is simply denoted by juxtaposition, both for concatenating words uv, languages L_1L_2 and words with languages such as uLv. We sometimes use the symbol $\cdot$ to refer explicitly to the concatenation operation.

A *finite automaton* (FA) is a tuple $\mathcal{A} = \langle Q, \delta, I, F, \Sigma \rangle$ where Σ is the *alphabet*, Q is the finite set of *states*, $I \subseteq Q$ are the *initial states*, $F \subseteq Q$ are the *final states*, and $\delta\colon Q \times \Sigma \to \wp(Q)$ is the *transition relation*. If $u \in \Sigma^*$ and $q, q' \in Q$ then $q \overset{u}{\rightsquigarrow} q'$ means that the state q' is reachable from q by following the string u. Therefore, $q \overset{\epsilon}{\rightsquigarrow} q'$ holds iff $q = q'$. The *language generated by a FA* $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) \triangleq \{u \in \Sigma^* \mid \exists q_i \in I, \exists q_f \in F,\ q_i \overset{u}{\rightsquigarrow} q_f\}$. Figure 1 depicts an example of FA.

3 Inclusion Check by Complete Abstractions

The *language inclusion problem* consists in checking whether $L_1 \subseteq L_2$ holds where L_1 and L_2 are two languages over an alphabet Σ. In this section, we show how (backward) complete abstractions ρ can be used to compute an overapproximation $\rho(L_1)$ of L_1 such that $\rho(L_1) \subseteq L_2 \Leftrightarrow L_1 \subseteq L_2$.

Let $\mathrm{uco}(C)$ denote the set of upper closure operators (or simply closure operators) on a poset $\langle C, \leq_C \rangle$, that is, the set of monotonic, idempotent (i.e., $\rho(x) = \rho(\rho(x))$) and increasing (i.e., $x \leq_C \rho(x)$) functions in $C \to C$. We often write $c \in \rho(C)$ (or simply $c \in \rho$ when C is clear from the context) to denote that there exists $c' \in C$ with $c = \rho(c')$, and let us recall that this happens iff $\rho(c) = c$.

Closure-based abstract interpretation [7] can be applied to solve a generic inclusion checking problem stated through least fixpoints as follows. Let $\rho \in \text{uco}(C)$ and $c_2 \in C$ such that $c_2 \in \rho$. Then, for all $c_1 \in C$, it turns out that

$$c_1 \leq_C c_2 \Leftrightarrow \rho(c_1) \leq_C \rho(c_2) \Leftrightarrow \rho(c_1) \leq_C c_2. \tag{1}$$

We apply here the notion of backward completeness in abstract interpretation [6,7,16,23]. In abstract interpretation, a closure operator $\rho \in \text{uco}(C)$ on a concrete domain C plays the role of abstraction function for objects of C. A closure $\rho \in \text{uco}(C)$ is called backward complete for a concrete monotonic function $f : C \to C$ when $\rho f = \rho f \rho$ holds. 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 are approximated by ρ. It is well-known [7] that in this case backward completeness implies completeness of least fixpoints, namely, $\rho(\text{lfp}(f)) = \text{lfp}(\rho f) = \text{lfp}(\rho f \rho)$ holds by assuming that these least fixpoints exist (this is the case, e.g., when C is a CPO). Theorem 3.1 states that in order to check an inclusion $c_1 \leq_C c_2$ for some $c_1 = \text{lfp}(f)$ and $c_2 \in \rho$, it is enough to perform an inclusion check $\text{lfp}(\rho f) \leq_C c_2$ which is defined on the abstraction $\rho(C)$.

Theorem 3.1. *If C is a CPO, $f : C \to C$ is monotonic, $\rho \in \text{uco}(C)$ is backward complete for f and $c_2 \in \rho$, then $\text{lfp}(f) \leq_C c_2 \Leftrightarrow \text{lfp}(\rho f) \leq_C c_2$. In particular, if $\langle \rho, \leq_C \rangle$ is ACC then the Kleene iterates of $\text{lfp}(\rho f)$ are finitely many.*

In the following sections we apply this general abstraction technique for a number of different language inclusion problems, by designing decision algorithms which rely on specific backward complete abstractions of $\wp(\Sigma^*)$.

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

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

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

$$\mathcal{L}(\mathcal{A}) = \textstyle\bigcup_{q \in I} W_{q,F}^{\mathcal{A}} = \bigcup_{q \in F} W_{I,q}^{\mathcal{A}} \tag{2}$$

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 [24]. Given a Boolean predicate $p(x)$ (typically a membership predicate) and two sets T and F, we define the following parametric choice function:

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

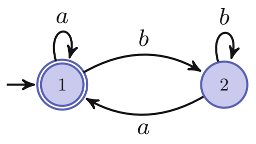

Fig. 1. A finite automaton $\mathcal{A}$ with $\mathcal{L}(\mathcal{A}) = (a + (b^+a))^*$.

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

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

Thus, the functions in the right-hand side of the equations in $\mathrm{Eqn}(\mathcal{A})$ have type $\wp(\Sigma^*)^{|Q|} \to \wp(\Sigma^*)$. Since $\langle \wp(\Sigma^*)^{|Q|}, \subseteq\rangle$ is a (product) complete lattice (as $\langle \wp(\Sigma^*), \subseteq\rangle$ is a complete lattice) and all the right-hand side functions in $\mathrm{Eqn}(\mathcal{A})$ are monotonic, the least solution $\langle Y_q\rangle_{q\in Q}$ of $\mathrm{Eqn}(\mathcal{A})$ does exist and it is easily seen that for every $q \in Q$, $Y_q = W_{q,F}^{\mathcal{A}}$ holds.

Note that, by using right (rather than left) concatenations, one could also define an equivalent set of equations whose least solution coincides with $\langle W_{I,q}^{\mathcal{A}}\rangle_{q\in Q}$ instead of $\langle W_{q,F}^{\mathcal{A}}\rangle_{q\in Q}$.

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

$$\mathrm{Eqn}(\mathcal{A}) = \begin{cases} X_1 = \{\epsilon\} \cup X_1a \cup X_2a \\ X_2 = \varnothing \cup X_1b \cup X_2b \end{cases} . \qquad \Diamond$$

Equivalently, the equations in $\mathrm{Eqn}(\mathcal{A})$ can be stated using an "initial" vector $\vec{\boldsymbol{\epsilon}}^F \in \wp(\Sigma^*)^{|Q|}$ and the function $\mathrm{Pre}_{\mathcal{A}} \colon \wp(\Sigma^*)^{|Q|} \to \wp(\Sigma^*)^{|Q|}$ defined as follows:

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

Since $\epsilon \in W_{q,F}^{\mathcal{A}}$ for all $q \in F$, the least fixpoint computation can start from the vector $\vec{\boldsymbol{\epsilon}}^F$ and iteratively apply $\mathrm{Pre}_{\mathcal{A}}$, that is, it turns out that

$$\langle W_{q,F}^{\mathcal{A}}\rangle_{q\in Q} = \mathrm{lfp}(\lambda \vec{\boldsymbol{X}}.\, \vec{\boldsymbol{\epsilon}}^F \cup \mathrm{Pre}_{\mathcal{A}}(\vec{\boldsymbol{X}})). \tag{3}$$

Together with Equation (2), it follows that $\mathcal{L}(\mathcal{A})$ equals the union of the component languages of the vector $\mathrm{lfp}(\lambda \vec{\boldsymbol{X}}.\, \vec{\boldsymbol{\epsilon}}^F \cup \mathrm{Pre}_{\mathcal{A}}(\vec{\boldsymbol{X}}))$ indexed by initial states in I.

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

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

Fixpoint-Based Inclusion Check. Consider the 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:

$$\overrightarrow{\boldsymbol{L_2}}^I \triangleq \langle \psi_{\Sigma^*}^{L_2}(q \in I) \rangle_{q \in Q}. \tag{4}$$

Using (2), (3) and (4), it is routine to prove that

$$\mathcal{L}(\mathcal{A}) \subseteq L_2 \Leftrightarrow \mathrm{lfp}(\lambda \overrightarrow{\boldsymbol{X}}.\, \overrightarrow{\boldsymbol{\epsilon}}^F \cup \mathrm{Pre}_{\mathcal{A}}(\overrightarrow{\boldsymbol{X}})) \subseteq \overrightarrow{\boldsymbol{L_2}}^I. \tag{5}$$

4.2 Abstract Inclusion Check Using Closures

In what follows we will use Theorem 3.1 for solving the language inclusion problem, where we will have that $C = \langle \wp(\Sigma^*)^{|Q|}, \subseteq \rangle$, $f = \lambda \overrightarrow{\boldsymbol{X}}.\, \overrightarrow{\boldsymbol{\epsilon}}^F \cup \mathrm{Pre}_{\mathcal{A}}(\overrightarrow{\boldsymbol{X}})$ and $\rho : \wp(\Sigma^*)^{|Q|} \to \wp(\Sigma^*)^{|Q|}$ is an upper closure operator.

Theorem 4.3. *Let Σ be an alphabet. 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 $\mathcal{A} = \langle Q, \delta, I, F, \Sigma \rangle$, the closure ρ is backward complete for $\mathrm{Pre}_{\mathcal{A}}$ and $\lambda \overrightarrow{\boldsymbol{X}}.\, \overrightarrow{\boldsymbol{\epsilon}}^F \cup \mathrm{Pre}_{\mathcal{A}}(\overrightarrow{\boldsymbol{X}})$.*

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 $\rho(\mathrm{lfp}(\lambda \overrightarrow{\boldsymbol{X}}.\, \overrightarrow{\boldsymbol{\epsilon}}^F \cup \mathrm{Pre}_{\mathcal{A}}(\overrightarrow{\boldsymbol{X}}))) = \mathrm{lfp}(\lambda \overrightarrow{\boldsymbol{X}}.\, \rho(\overrightarrow{\boldsymbol{\epsilon}}^F \cup \mathrm{Pre}_{\mathcal{A}}(\overrightarrow{\boldsymbol{X}})))$.*

Note that if ρ is backward complete for $\lambda X.aX$ for all $a \in \Sigma$ and $L_2 \in \rho$ then, as a consequence of Theorem 3.1 and Corollary 4.4, the equivalence (5) becomes

$$\mathcal{L}(\mathcal{A}) \subseteq L_2 \Leftrightarrow \mathrm{lfp}(\lambda \overrightarrow{\boldsymbol{X}}.\, \rho(\overrightarrow{\boldsymbol{\epsilon}}^F \cup \mathrm{Pre}_{\mathcal{A}}(\overrightarrow{\boldsymbol{X}}))) \subseteq \overrightarrow{\boldsymbol{L_2}}^I. \tag{6}$$

4.3 Abstract Inclusion Check Using Galois Connections

To solve a language inclusion problem $\mathcal{L}(\mathcal{A}) \subseteq L_2$ using equivalence (6) we must first compute the corresponding least fixpoint and then decide its inclusion in $\overrightarrow{\boldsymbol{L_2}}^I$. Since closure operators are fully isomorphic to Galois connections [7, Section 6], they allow us to conveniently define and reason on abstract domains independently of their representation. 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 $\langle C, \leq_C \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \leq_A \rangle$. In an adjunction, 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).

The next result shows that there exists an algorithm that solves the language inclusion problem $\mathcal{L}(\mathcal{A}) \subseteq L_2$ on an abstraction D of the concrete domain of languages $\langle \wp(\Sigma^*), \subseteq \rangle$ whenever D satisfies a list of requirements related to backward completeness and computability.

Theorem 4.5. *Let $\mathcal{A} = \langle Q, \delta, I, F, \Sigma\rangle$ be a FA and let L_2 be a language over Σ. Let $\langle \wp(\Sigma^*), \subseteq\rangle \xleftrightarrows[\alpha]{\gamma} \langle D, \leq_D\rangle$ be a GC where $\langle D, \leq_D\rangle$ is a poset. Assume that the following properties hold:*

(i) $L_2 \in \gamma(D)$ and for every $a \in \Sigma$ and $X \in \wp(\Sigma^*)$, $\alpha(aX) = \alpha(a\gamma\alpha(X))$.
(ii) $(D, \leq_D, \sqcup)$ is an effective domain, meaning that: $(D, \leq_D, \sqcup)$ is an ACC join-semilattice, every element of D has a finite representation, $\leq_D$ is decidable and $\sqcup$ is a computable binary lub.
(iii) There is an algorithm, say $\mathrm{Pre}^\sharp(\vec{\boldsymbol{X}})$, which computes $\alpha(\mathrm{Pre}_{\mathcal{A}}(\gamma(\vec{\boldsymbol{X}})))$, for all $\vec{\boldsymbol{X}} \in \wp(\Sigma^*)^{|Q|}$.
(iv) There is an algorithm, say $\epsilon^\sharp$, computing $\alpha(\vec{\boldsymbol{\epsilon}}^F)$.
(v) There is an algorithm, say $\mathrm{Incl}^\sharp(\vec{\boldsymbol{X}})$, deciding the abstract inclusion $\vec{\boldsymbol{X}} \leq_D \alpha(\vec{\boldsymbol{L_2}}^I)$, for every vector $\vec{\boldsymbol{X}} \in \alpha(\wp(\Sigma^*)^{|Q|})$.

Then, the following algorithm decides whether $\mathcal{L}(\mathcal{A}) \subseteq L_2$:

```
$\langle Y_q\rangle_{q\in Q} :=$ Kleene$(\lambda\vec{\boldsymbol{X}}.\, \epsilon^\sharp \sqcup \mathrm{Pre}^\sharp(\vec{\boldsymbol{X}}), \vec{\boldsymbol{\varnothing}})$;
return $\mathrm{Incl}^\sharp(\langle Y_q\rangle_{q\in Q})$;
```

Quasiorder Galois Connections. It turns out that Theorem 4.5 still holds for abstract domains which are mere qosets rather than posets.

Definition 4.6 (Quasiorder GC). A *quasiorder GC* (QGC) $\langle C, \leq_C\rangle \xleftrightarrows[\alpha]{\gamma} \langle D, \leq_D\rangle$ consists of: (a) two qosets $\langle C, \leq_C\rangle$ and $\langle D, \leq_D\rangle$ such that one of them is a poset; (b) two functions $\alpha\colon C \to D$ and $\gamma\colon D \to C$ such that $\alpha(c) \leq_D d \Leftrightarrow c \leq_C \gamma(d)$ holds for all $c \in C$ and $d \in D$. ■

Analogously to GCs, it is easily seen that in QGCs both α and γ are monotonic as well as $c \leq_C \gamma(\alpha(c))$ and $\alpha(\gamma(d)) \leq_D d$ always hold. Observe that if C is a poset and $d \leq_D d' \leq_D d$ with $d \neq d'$ then $\gamma(d) = \gamma(d')$, because γ is monotonic, and conversely, if D is a poset and $c \leq_C c' \leq_C c$ with $c \neq c'$ then $\alpha(c) = \alpha(c')$ holds. Similarly to GCs, if C is a poset then $\gamma \circ \alpha \in \mathrm{uco}(\langle C, \leq_C\rangle)$ holds for QGCs.

In the following, we apply all the standard order-theoretic notions used for posets also to a qoset $\langle D, \leq_D\rangle$ by implicitly referring to the quotient poset $\langle D_{/\cong_D}, \leq_{D/\cong_D}\rangle$ where $\cong_D \triangleq \leq_D \cap \leq_D^{-1}$.

For example:

- $\langle D, \leq_D\rangle$ is ACC (CPO) means that the poset $\langle D_{/\cong_D}, \leq_{D/\cong_D}\rangle$ is ACC (CPO).
- $\langle D, \leq_D\rangle$ is a join-semilattice means that $\langle D_{/\cong_D}, \leq_{D/\cong_D}\rangle$ is a join-semilattice; a binary lub for D (one could have several binary lubs) is a map $\lambda\langle d, d'\rangle. d \sqcup d'$ such that $\lambda\langle [d]_{\cong_D}, [d']_{\cong_D}\rangle.[d \sqcup d']_{\cong_D}$ is the lub in the poset $\langle D_{/\cong_D}, \leq_{D/\cong_D}\rangle$.

Corollary 4.7. *Theorem 4.5 still holds for a QGC $\langle \wp(\Sigma^*), \subseteq\rangle \xleftrightarrows[\alpha]{\gamma} \langle D, \leq_D\rangle$ where $\langle D, \leq_D\rangle$ is a qoset.*

5 Instantiating the Framework

In this section we focus on a particular class of closures on sets of words: those induced by quasiorder relations on words. Then, we provide a list of conditions on quasiorders such that the induced closures fit our framework. In addition, we study some instances of such quasiorders and compare them.

5.1 Word-Based Abstractions

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

$$\rho_{\leqslant}(X) \triangleq \{v \in \Sigma^* \mid \exists u \in X,\, u \leqslant v\}. \tag{7}$$

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 $\langle \wp(\Sigma^*), \subseteq \rangle$.

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

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

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

Lemma 5.2. *Let L be a language over Σ and $\leqslant_L$ be a left (resp. right) L-consistent quasiorder on Σ^*. Then,*

(a) $\rho_{\leqslant_L}(L) = L$.
(b) $\rho_{\leqslant_L}$ is backward complete for $\lambda X.\, aX$ (resp. $\lambda X.\, Xa$) for all $a \in \Sigma$.

Moreover, we show that the $\leqslant$-upward closure $\rho_{\leqslant}$ in (7) can be equivalently defined through the qoset of antichains. In fact, the qoset of antichains $\langle \mathrm{AC}_{\langle \Sigma^*, \leqslant \rangle}, \sqsubseteq \rangle$ can be viewed as a language abstraction through the minor abstraction map. Let $\alpha_{\leqslant} : \wp(\Sigma^*) \to \mathrm{AC}_{\langle \Sigma^*, \leqslant \rangle}$ and $\gamma_{\leqslant} : \mathrm{AC}_{\langle \Sigma^*, \leqslant \rangle} \to \wp(\Sigma^*)$ be defined as follows:

$$\alpha_{\leqslant}(X) \triangleq \lfloor X \rfloor, \qquad \gamma_{\leqslant}(Y) \triangleq \rho_{\leqslant}(Y). \tag{8}$$

Theorem 5.3. *Let $\langle \Sigma^*, \leqslant \rangle$ be a qoset.*

(a) $\langle \wp(\Sigma^*), \subseteq \rangle \underset{\alpha_{\leqslant}}{\overset{\gamma_{\leqslant}}{\leftrightarrows}} \langle \mathrm{AC}_{\langle \Sigma^*, \leqslant \rangle}, \sqsubseteq \rangle$ *is a QGC.*
(b) $\gamma_{\leqslant} \circ \alpha_{\leqslant} = \rho_{\leqslant}$.

The QGC $\langle \wp(\Sigma^*), \subseteq \rangle \xleftrightarrow[\alpha_{\leqslant}]{\gamma_{\leqslant}} \langle \mathrm{AC}_{\langle \Sigma^*, \leqslant \rangle}, \sqsubseteq \rangle$ allows us to represent and manipulate $\leqslant$-upward closed sets in $\wp(\Sigma^*)$ using finite subsets, as already shown by Abdulla et al. [1].

We are now in position to show that, given a language L_2 whose membership decision problem is decidable, for every decidable L_2-consistent wqo relation $\leqslant_{L_2}$, the QGC $\langle \wp(\Sigma^*), \subseteq \rangle \xleftrightarrow[\alpha_{\leqslant_{L_2}}]{\gamma_{\leqslant_{L_2}}} \langle \mathrm{AC}_{\langle \Sigma^*, \leqslant_{L_2} \rangle}, \sqsubseteq \rangle$ of Theorem 5.3 (a) yields an algorithm for deciding the inclusion $\mathcal{L}(\mathcal{A}) \subseteq L_2$ where $\mathcal{A}$ is a FA. In particular, for a left L_2-consistent wqo $\leqslant^l_{L_2}$, the algorithm FAIncW solves this inclusion problem. FAIncW is called "word-based" algorithm because the vector $\langle Y_q \rangle_{q \in Q}$ used by FAIncW consists of finite sets of words.

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

Data: FA $\mathcal{A} = \langle Q, \delta, I, F, \Sigma \rangle$; a decision procedure
      for membership in $L_2$; a decidable left $L_2$-consistent wqo $\leqslant^l_{L_2}$.
1 $\langle Y_q \rangle_{q \in Q} :=$ Kleene$(\lambda \vec{X}. \lfloor \vec{\epsilon}^F \rfloor \sqcup \lfloor \mathrm{Pre}_{\mathcal{A}}(\vec{X}) \rfloor, \vec{\varnothing})$;
2 forall the $q \in I$ do
3     forall the $u \in Y_q$ do
4         if $u \notin L_2$ then return false
5 return true;
```

Theorem 5.4. *Let $\mathcal{A}$ be a FA and let L_2 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, the algorithm* FAIncW *decides the inclusion $\mathcal{L}(\mathcal{A}) \subseteq L_2$.*

A symmetric version of algorithm FAIncW (and of Theorem 5.4) for *right* L_2-consistent wqos, which relies on equations concatenating to the right (instead of to the left as in $\mathrm{Eqn}(\mathcal{A})$), can be found in the extended version of this paper [14].

In what follows, we will consider different quasiorders and we will show that they fulfill the requirements of Theorem 5.4 (or its symmetric version for right quasiorders), so that they yield algorithms for solving the language inclusion problem.

5.2 Nerode Quasiorders

Given $w \in \Sigma^*$ and $X \in \wp(\Sigma^*)$, left and right quotients are defined as usual: $w^{-1}X \triangleq \{u \in \Sigma^* \mid wu \in X\}$ and $Xw^{-1} \triangleq \{u \in \Sigma^* \mid uw \in X\}$. Given a language $L \subseteq \Sigma^*$, let us define the following quasiorder relations on Σ^*:

$$u \leqq^l_L v \overset{\triangle}{\Longleftrightarrow} Lu^{-1} \subseteq Lv^{-1}, \qquad u \leqq^r_L v \overset{\triangle}{\Longleftrightarrow} u^{-1}L \subseteq v^{-1}L.$$

De Luca and Varricchio [8] call them, resp., the *left* ($\leqq^l_L$) and *right* ($\leqq^r_L$) *Nerode quasiorders relative to* L. The following result shows that Nerode quasiorders are the most general (i.e., greatest for set inclusion) 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 5.5. *Let $L \subseteq \Sigma^*$ be a language.*

(a) $\leqq^l_L$ and $\leqq^r_L$ are, resp., left and right L-consistent quasiorders. If L is regular then $\leqq^l_L$ and $\leqq^r_L$ are, additionally, decidable wqos.
(b) Let $\leqslant$ be a quasiorder on Σ^*. If $\leqslant$ is left (resp. right) L-consistent then $\rho_{\leqq^l_L} \subseteq \rho_{\leqslant}$ (resp. $\rho_{\leqq^r_L} \subseteq \rho_{\leqslant}$).

Let us now consider a first instantiation of Theorem 5.4. Because membership is decidable for regular languages, Lemma 5.5 (a) for $\leqq^l_{L_2}$ implies that the hypotheses (i) and (ii) of Theorem 5.4 are satisfied, so that the algorithm `FAIncW` decides the inclusion $\mathcal{L}(\mathcal{A}) \subseteq L_2$. Furthermore, under these hypotheses, Lemma 5.5 (b) shows that $\leqq^l_{L_2}$ is the most general (i.e., greatest for set inclusion) left L_2-consistent quasiorder relation on Σ^* for which the algorithm `FAIncW` can be instantiated for deciding an inclusion $\mathcal{L}(\mathcal{A}) \subseteq L_2$.

Remark 5.6 (On the Complexity of Nerode quasiorders). For the inclusion problem between languages generated by finite automata, deciding the (left or right) Nerode quasiorder can be easily shown[1] to be as hard as the language inclusion problem, which is PSPACE-complete. For the inclusion problem of a language generated by an automaton within the trace set of a one-counter net (cf. Sect. 5.3) the right Nerode quasiorder is a right language-consistent well-quasiorder but it turns out to be undecidable (cf. Lemma 5.12). More details can be found in the extended version of this paper [14]. ■

5.3 State-Based Quasiorders

Consider the 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 $\mathcal{A}_2$, called state-based quasiorders. This is a strict subclass of Nerode quasiorders defined in Sect. 5.2 and sidesteps the untractability or undecidability of Nerode quasiorders (cf. Remark 5.6) yet allowing to define an algorithm solving the language inclusion problem.

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:

$$u \leq^l_{\mathcal{A}} v \overset{\triangle}{\Longleftrightarrow} \mathrm{pre}^{\mathcal{A}}_u(F) \subseteq \mathrm{pre}^{\mathcal{A}}_v(F), \qquad u \leq^r_{\mathcal{A}} v \overset{\triangle}{\Longleftrightarrow} \mathrm{post}^{\mathcal{A}}_u(I) \subseteq \mathrm{post}^{\mathcal{A}}_v(I), \tag{9}$$

where, for all $X \subseteq Q$ and $u \in \Sigma^*$, $\mathrm{pre}^{\mathcal{A}}_u(X) \triangleq \{q \in Q \mid u \in W^{\mathcal{A}}_{q,X}\}$ and $\mathrm{post}^{\mathcal{A}}_u(X) \triangleq \{q' \in Q \mid u \in W^{\mathcal{A}}_{X,q'}\}$.

Lemma 5.7. *Let $\mathcal{A}$ be a FA. Then, $\leq^l_{\mathcal{A}}$ and $\leq^r_{\mathcal{A}}$ are, resp., decidable left and right $\mathcal{L}(\mathcal{A})$-consistent wqos.*

[1] Sketch: Given $\mathcal{A}_1 = (Q_1, \delta_1, I_1, F_1, \Sigma)$ and $\mathcal{A}_2 = (Q_2, \delta_2, I_2, F_2, \Sigma)$ define $\mathcal{A}_3 = (Q_1 \cup Q_2 \cup \{q^\dagger\}, \delta_3, \{q^\dagger\}, F_1 \cup F_2)$ where δ_3 maps $(q^\dagger, a)$ to I_1, $(q^\dagger, b)$ to I_2 and like δ_1 or δ_2 elsewhere. Then, it turns out that $a \leqq^r_{\mathcal{L}(\mathcal{A}_3)} b \Leftrightarrow a^{-1}\mathcal{L}(\mathcal{A}_3) \subseteq b^{-1}\mathcal{L}(\mathcal{A}_3) \Leftrightarrow \mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$.

It follows from Lemma 5.7 that Theorem 5.4 applies to $\leq^l_{\mathcal{A}_2}$ (and $\leq^r_{\mathcal{A}_2}$), so that one can instantiate the algorithm FAIncW with the wqo $\leq^l_{\mathcal{A}_2}$ for deciding $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$. Turning back to the left Nerode wqo $\leqq^l_{\mathcal{L}(\mathcal{A}_2)}$ we find that:

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

Since $\mathrm{pre}^{\mathcal{A}_2}_u(F) \subseteq \mathrm{pre}^{\mathcal{A}_2}_v(F) \Rightarrow W_{I,\mathrm{pre}^{\mathcal{A}_2}_u(F)} \subseteq W_{I,\mathrm{pre}^{\mathcal{A}_2}_v(F)}$, it follows that $u \leq^l_{\mathcal{A}_2} v \Rightarrow u \leqq^l_{\mathcal{L}(\mathcal{A}_2)} v$. Moreover, by Lemmas 5.5 (b) and 5.7, we have that $\rho_{\leqq^l_{\mathcal{L}(\mathcal{A}_2)}} \subseteq \rho_{\leq^l_{\mathcal{A}_2}}$.

Simulation-Based Quasiorders. Recall that, given a FA $\mathcal{A} = \langle Q, \delta, I, F, \Sigma\rangle$, a *simulation* on $\mathcal{A}$ is a binary relation $\preceq \subseteq Q \times Q$ such that if $p \preceq q$ then: (i) $p \in F$ implies $q \in F$ and (ii) for every transition $p \xrightarrow{a} p'$, there exists a transition $q \xrightarrow{a} q'$ such that $p' \preceq q'$. It is well-known that simulation implies language inclusion, i.e., if $\preceq$ is a simulation on $\mathcal{A}$ then

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

We lift a quasiorder $\preceq$ on Q to a quasiorder $\preceq^{\forall\exists}$ on $\wp(Q)$ as follows:

$$X \preceq^{\forall\exists} Y \overset{\triangle}{\iff} \forall x \in X, \exists y \in Y,\ x \preceq y$$

so that $X \preceq^{\forall\exists} Y \Rightarrow W^{\mathcal{A}}_{X,F} \subseteq W^{\mathcal{A}}_{Y,F}$ holds. Therefore, we define the *right simulation-based quasiorder* $\preceq^r_{\mathcal{A}}$ on Σ^* as follows:

$$u \preceq^r_{\mathcal{A}} v \overset{\triangle}{\iff} \mathrm{post}^{\mathcal{A}}_u(I) \preceq^{\forall\exists} \mathrm{post}^{\mathcal{A}}_v(I). \tag{10}$$

Lemma 5.8. *Given a simulation relation $\preceq$ on $\mathcal{A}$, the right simulation-based quasiorder $\preceq^r_{\mathcal{A}}$ is a decidable right $\mathcal{L}(\mathcal{A})$-consistent wqo.*

Thus, once again, Theorem 5.4 applies to $\preceq^r_{\mathcal{A}_2}$ and this allows us to instantiate the algorithm FAIncW to $\preceq^r_{\mathcal{A}_2}$ for deciding $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$.

Observe that $u \preceq^r_{\mathcal{A}_2} v$ 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 $u \leqq^r_{\mathcal{L}(\mathcal{A}_2)} v$, so that $u \preceq^r_{\mathcal{A}_2} v \Rightarrow u \leqq^r_{\mathcal{L}(\mathcal{A}_2)} v$. Moreover, $u \leq^r_{\mathcal{A}_2} v \Rightarrow u \preceq^r_{\mathcal{A}_2} v$ trivially holds. Summing up, the following containments relate (the right versions of) state-based, simulation-based and Nerode quasiorders:

$$\leq^r_{\mathcal{A}_2} \subseteq \preceq^r_{\mathcal{A}_2} \subseteq \leqq^r_{\mathcal{L}(\mathcal{A}_2)}.$$

All these quasiorders are decidable $\mathcal{L}(\mathcal{A}_2)$-consistent wqos so that the algorithm FAIncW can be instantiated for each of them for deciding $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$.

Inclusion in Traces of One-Counter Nets. We show that our framework can be instantiated to systematically derive an algorithm for deciding the inclusion

$\mathcal{L}(\mathcal{A}) \subseteq L_2$ where L_2 is the trace set of a one-counter net. This is accomplished by defining a decidable L_2-consistent quasiorder so that Theorem 5.4 can be applied.

Intuitively, a one-counter net is a FA equipped with a nonnegative integer counter. Formally, a One-Counter Net (OCN) [17] is a tuple $\mathcal{O} = \langle Q, \Sigma, \delta \rangle$ 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 $\mathcal{O}$ 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 and $q'n'$ 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:

$$T(qn) \triangleq \{u \in \Sigma^* \mid Z_u^{qn} \neq \varnothing\} \quad \text{where}$$
$$Z_u^{qn} \triangleq \{q_k n_k \mid qn \xrightarrow{a_1} q_1 n_1 \xrightarrow{a_2} \cdots \xrightarrow{a_k} q_k n_k,\ a_1 \cdots a_k = u\}.$$

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

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

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

where $\max \varnothing \triangleq \perp$. Define the following quasiorder on Σ^*:

$$u \leq_{qn}^r v \stackrel{\triangle}{\iff} \forall q' \in Q,\ M_{Z_u^{qn}}(q') \leq_{\mathbb{N}_\perp} M_{Z_v^{qn}}(q').$$

Lemma 5.9. *Given a OCN $\mathcal{O}$ together with a configuration qn, $\leq_{qn}^r$ is a right $T(qn)$-consistent decidable wqo.*

Thus, as a consequence of Theorem 5.4, Lemma 5.9 and the decidability of membership in $T(qn)$, we derive the following known decidability result [19, Theorem 3.2] by resorting to our framework.

Theorem 5.10. *Given a FA $\mathcal{A}$ and a OCN $\mathcal{O}$ together with a configuration qn, the problem $\mathcal{L}(\mathcal{A}) \subseteq T(qn)$ is decidable.*

Moreover, the following result closes a conjecture made by de Luca and Varricchio [8, Section 6].

Lemma 5.11. *The right Nerode quasiorder $\leqq_{T(qn)}^r$ is a well-quasiorder.*

It is worth remarking that, by Lemma 5.5 (a), the left and right Nerode quasiorders relative to $T(qn)$ are $T(qn)$-consistent. However, the left Nerode quasiorder does not need to be a wqo for otherwise $T(qn)$ would be regular.

Lemma 5.12. *The right Nerode quasiorder for the trace set of OCN is undecidable.*

We conjecture that, using our framework, Theorem 5.10 can be extended to traces of Petri Nets, which is already known to be true [19].

6 A Novel Perspective on the Antichain Algorithm

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 left $\mathcal{L}(\mathcal{A}_2)$-consistent wqo $\leq^l_{\mathcal{A}_2}$ defined in (9). Theorem 5.4 shows that the algorithm `FAIncW` solves the inclusion problem $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$ by computing on the qoset abstraction $\langle \mathrm{AC}_{\langle \Sigma^*, \leq^l_{\mathcal{A}_2}\rangle}, \sqsubseteq\rangle$ of antichains of $\langle \Sigma^*, \leq^l_{\mathcal{A}_2}\rangle$.

Since $u \leq^l_{\mathcal{A}_2} v \Leftrightarrow \mathrm{pre}^{\mathcal{A}_2}_u(F_2) \subseteq \mathrm{pre}^{\mathcal{A}_2}_v(F_2)$ holds, we can equivalently consider the set of states $\mathrm{pre}^{\mathcal{A}_2}_u(F_2)$ rather than a word $u \in \Sigma^*$. This leads us to design an algorithm analogous to `FAIncW` but computing on the poset $\langle \mathrm{AC}_{\langle \wp(Q_2), \subseteq\rangle}, \sqsubseteq\rangle$ of antichains of sets of states of $\langle \wp(Q_2), \subseteq\rangle$. In order to do this, the poset $\langle \mathrm{AC}_{\langle \wp(Q_2), \subseteq\rangle}, \sqsubseteq\rangle$ is viewed as an abstraction of the qoset $\langle \mathrm{AC}_{\langle \Sigma^*, \leq^l_{\mathcal{A}_2}\rangle}, \sqsubseteq'\rangle$ (where $\sqsubseteq'$ is used for distinguishing the two ordering relations on antichains) through the abstraction and concretization maps $\alpha_{\mathcal{A}_2} : \mathrm{AC}_{\langle \Sigma^*, \leq^l_{\mathcal{A}_2}\rangle} \to \mathrm{AC}_{\langle \wp(Q_2), \subseteq\rangle}$ and $\gamma_{\mathcal{A}_2} : \mathrm{AC}_{\langle \wp(Q_2), \subseteq\rangle} \to \mathrm{AC}_{\langle \Sigma^*, \leq^l_{\mathcal{A}_2}\rangle}$ defined as follows:

$$\alpha_{\mathcal{A}_2}(X) \triangleq \{\mathrm{pre}^{\mathcal{A}_2}_u(F_2) \in \wp(Q_2) \mid u \in X\},$$
$$\gamma_{\mathcal{A}_2}(Y) \triangleq \lfloor \{u \in \Sigma^* \mid \mathrm{pre}^{\mathcal{A}_2}_u(F_2) \in Y\}\rfloor.$$

Lemma 6.1. $\langle \mathrm{AC}_{\langle \Sigma^*, \leq^l_{\mathcal{A}_2}\rangle}, \sqsubseteq'\rangle \underset{\alpha_{\mathcal{A}_2}}{\overset{\gamma_{\mathcal{A}_2}}{\leftrightarrows}} \langle \mathrm{AC}_{\langle \wp(Q_2), \subseteq\rangle}, \sqsubseteq\rangle$ *is a QGC.*

Combining the word-based algorithm `FAIncW` with $\alpha_{\mathcal{A}_2}$ and $\gamma_{\mathcal{A}_2}$ we are able to systematically derive a new algorithm which solves the inclusion $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$ using the abstract domain $\langle \mathrm{AC}_{\langle \wp(Q_2), \subseteq\rangle}, \sqsubseteq\rangle$ which is viewed as an abstraction of $\langle \wp(\Sigma^*), \subseteq\rangle$ by composing the following two QGCs:

$$\langle \wp(\Sigma^*), \subseteq\rangle \underset{\alpha_{\leq^l_{\mathcal{A}_2}}}{\overset{\gamma_{\leq^l_{\mathcal{A}_2}}}{\leftrightarrows}} \langle \mathrm{AC}_{\langle \Sigma^*, \leq^l_{\mathcal{A}_2}\rangle}, \sqsubseteq'\rangle, \qquad \text{[by Theorem 5.3(a)]}$$
$$\langle \mathrm{AC}_{\langle \Sigma^*, \leq^l_{\mathcal{A}_2}\rangle}, \sqsubseteq'\rangle \underset{\alpha_{\mathcal{A}_2}}{\overset{\gamma_{\mathcal{A}_2}}{\leftrightarrows}} \langle \mathrm{AC}_{\langle \wp(Q_2), \subseteq\rangle}, \sqsubseteq\rangle. \qquad \text{[by Lemma 6.1]}$$

Let $\alpha \colon \wp(\Sigma^*) \to \mathrm{AC}_{\langle \wp(Q_2), \subseteq\rangle}$, $\gamma \colon \mathrm{AC}_{\langle \wp(Q_2), \subseteq\rangle} \to \wp(\Sigma^*)$ and $\mathrm{Pre}^{\mathcal{A}_2}_{\mathcal{A}_1} : \wp(Q_2)^{|Q_1|} \to \wp(Q_2)^{|Q_1|}$ be defined as follows:

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

Lemma 6.2. *The following properties hold:*

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

(e) For all $\vec{\boldsymbol{X}} \in \alpha(\wp(\Sigma^*)^{|Q_1|}), \mathrm{Pre}^{\mathcal{A}_2}_{\mathcal{A}_1}(\vec{\boldsymbol{X}}) = \alpha_{\mathcal{A}_2} \circ \alpha_{\leq^l_{\mathcal{A}_2}} \circ \mathrm{Pre}_{\mathcal{A}_1} \circ \gamma_{\leq^l_{\mathcal{A}_2}} \circ \gamma_{\mathcal{A}_2}(\vec{\boldsymbol{X}})$

It follows from Lemma 6.2 that the GC $\langle \wp(\Sigma^*), \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathrm{AC}_{\langle \wp(Q_2), \subseteq\rangle}, \sqsubseteq\rangle$ together with the abstract function $\mathrm{Pre}^{\mathcal{A}_2}_{\mathcal{A}_1}$ satisfy the hypotheses (i)–(iv) of Theorem 4.5. Thus, in order to obtain an algorithm for deciding $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$ it remains to show that requirement (v) of Theorem 4.5 holds, i.e., there is an algorithm to decide whether $\vec{\boldsymbol{Y}} \sqsubseteq \alpha(\vec{\boldsymbol{L_2}}^{I_2})$ for every $\vec{\boldsymbol{Y}} \in \alpha(\wp(\Sigma^*))^{|Q_1|}$.

Let us notice that the Kleene iterates of the function $\lambda\vec{\boldsymbol{X}}.\, \alpha(\vec{\boldsymbol{\epsilon}}^{F_1}) \sqcup \mathrm{Pre}^{\mathcal{A}_2}_{\mathcal{A}_1}(\vec{\boldsymbol{X}})$ of Theorem 4.5 are vectors of antichains in $\langle \mathrm{AC}_{\langle \wp(Q_2), \subseteq\rangle}, \sqsubseteq\rangle$, where each component indexed by some $q \in Q_1$ represents (through its minor set) a set of sets of states that are predecessors of F_2 in $\mathcal{A}_2$ by a word generated by $\mathcal{A}_1$ from that state q (i.e., $\mathrm{pre}^{\mathcal{A}_2}_u(F_2)$ with $u \in W^{\mathcal{A}_1}_{q,F_1}$). Since $\epsilon \in W^{\mathcal{A}_1}_{q,F_1}$ for all $q \in F_1$ and $\mathrm{pre}^{\mathcal{A}_2}_\epsilon(F_2) = F_2$ the iterations of the procedure `Kleene` begin with $\alpha(\vec{\boldsymbol{\epsilon}}^{F_1}) = \langle \psi^{F_2}_{\varnothing}(q \in F_1)\rangle_{q \in Q_1}$. By taking the minor of each vector component, we are considering smaller sets which still preserve the relation $\sqsubseteq$ (because $A \sqsubseteq B \Leftrightarrow \lfloor A \rfloor \sqsubseteq B \Leftrightarrow A \sqsubseteq \lfloor B \rfloor \Leftrightarrow \lfloor A \rfloor \sqsubseteq \lfloor B \rfloor$). Let $\vec{\boldsymbol{Y}}$ be the fixpoint computed by the `Kleene` procedure. We have that, for each component $q \in Q_1$, $\vec{\boldsymbol{Y}}_q = \lfloor \{\mathrm{pre}^{\mathcal{A}_2}_u(F_2) \mid u \in W^{\mathcal{A}_1}_{q,F_1}\}\rfloor$. Whenever $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$ holds, all the sets of states in $\vec{\boldsymbol{Y}}_q$ for $q \in I_1$ are predecessors of F_2 in $\mathcal{A}_2$ by words in $\mathcal{L}(\mathcal{A}_2)$, so that they all contain at least one initial state in I_2. As a result, we obtain the following algorithm `FAIncS`, a "state-based" inclusion algorithm for deciding $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$.

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

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

De Wulf et al. [10] introduced two antichain algorithms, called forward and backward, for deciding the universality of the language generated by a FA, i.e., whether the language is Σ^* or not. Then, they extended the backward algorithm in order to decide language inclusion. In what follows we show that `FAIncS` is equivalent to the corresponding extension of the forward algorithm and, therefore, dual to the antichain algorithm of De Wulf et al. [10].

To do that, 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 the language inclusion between two languages holds iff it holds between the reverse languages generated by the reverse automata, we conclude that the algorithm FAIncS is equivalent to the forward antichain algorithm.

Let us consider the following poset of antichains $\langle \mathrm{AC}_{\langle \wp(Q_2),\subseteq\rangle}, \widetilde{\sqsubseteq}\rangle$ where

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

It is easy to see that $\widetilde{\sqsubseteq}$ and $\sqsubseteq^{-1}$ coincides. As observed by De Wulf et al. [10], it turns out that $\langle \mathrm{AC}_{\langle \wp(Q_2),\subseteq\rangle}, \widetilde{\sqsubseteq}\rangle$ is a finite lattice, where $\widetilde{\sqcap}$ and $\widetilde{\sqcup}$ denote, resp., glb and lub of antichains. The 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 universality. The following result extends this forward algorithm in order to decide language inclusion.

Theorem 6.4 *bf ([10,* **Theorems 3 and 6]).** *Let* $\overrightarrow{\boldsymbol{\mathcal{FP}}} \triangleq \widetilde{\bigsqcap}\{\overrightarrow{\boldsymbol{X}} \in (\mathrm{AC}_{\langle \wp(Q_2),\subseteq\rangle})^{|Q_1|} \mid \overrightarrow{\boldsymbol{X}} = \mathrm{Post}^{\mathcal{A}_2}_{\mathcal{A}_1}(\overrightarrow{\boldsymbol{X}}) \,\widetilde{\sqcap}\, \langle \psi_{\varnothing}^{\{I_2\}}(q \in I_1)\rangle_{q\in Q_1}\}$ *with* $\mathrm{Post}^{\mathcal{A}_2}_{\mathcal{A}_1}(\langle X_q\rangle_{q\in Q_1}) \triangleq \langle \lfloor\{\mathrm{post}^{\mathcal{A}_2}_a(X) \mid \exists a \in \Sigma, q' \in Q_1, X \in X_{q'},\, q \in \delta_1(q',a)\}\rfloor\rangle_{q\in Q_1}$. *Then,* $\mathcal{L}(\mathcal{A}_1) \nsubseteq \mathcal{L}(\mathcal{A}_2)$ *iff* $\exists q \in F_1, \overrightarrow{\boldsymbol{\mathcal{FP}}}_q \mathrel{\widetilde{\sqsubseteq}} \{F_2^c\}$.

Let $\mathcal{A}^R$ denote the reverse 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, it is straightforward to check that $\mathrm{Post}^{\mathcal{A}_2}_{\mathcal{A}_1} = \mathrm{Pre}^{\mathcal{A}_2^R}_{\mathcal{A}_1^R}$. We therefore obtain the following result.

Theorem 6.5. *Let* $\overrightarrow{\boldsymbol{\mathcal{FP}}} \triangleq \widetilde{\bigsqcap}\{\overrightarrow{\boldsymbol{X}} \in (\mathrm{AC}_{\langle \wp(Q_2),\subseteq\rangle})^{|Q_1|} \mid \overrightarrow{\boldsymbol{X}} = \mathrm{Pre}^{\mathcal{A}_2}_{\mathcal{A}_1}(\overrightarrow{\boldsymbol{X}}) \,\widetilde{\sqcap}\, \langle \psi_{\varnothing}^{\{F_2\}}(q \in F_1)\rangle_{q\in Q_1}\}$. *Then,* $\mathcal{L}(\mathcal{A}_1) \nsubseteq \mathcal{L}(\mathcal{A}_2)$ *iff* $\exists q \in I_1, \overrightarrow{\boldsymbol{\mathcal{FP}}}_q \mathrel{\widetilde{\sqsubseteq}} \{I_2^c\}$.

Since $\widetilde{\sqsubseteq} = \sqsubseteq^{-1}$, we have that $\widetilde{\sqcap} = \sqcup$. Moreover, by definition of α we have that $\langle \psi_{\varnothing}^{\{F_2\}}(q \in F_1)\rangle_{q\in Q_1} = \alpha(\overrightarrow{\boldsymbol{\epsilon}}^{F_1})$. Therefore, we can rewrite the vector $\overrightarrow{\boldsymbol{\mathcal{FP}}}$ of Theorem 6.5 as $\overrightarrow{\boldsymbol{\mathcal{FP}}} = \bigsqcup\{\overrightarrow{\boldsymbol{X}} \mid \overrightarrow{\boldsymbol{X}} = \mathrm{Pre}^{\mathcal{A}_2}_{\mathcal{A}_1}(\overrightarrow{\boldsymbol{X}}) \sqcup \alpha(\overrightarrow{\boldsymbol{\epsilon}}^{F_1})\}$, which is the least fixpoint for $\sqsubseteq$ of $\mathrm{Pre}^{\mathcal{A}_2}_{\mathcal{A}_1}$ above $\alpha(\overrightarrow{\boldsymbol{\epsilon}}^{F_1})$. It turns out that the Kleene iterates of this least fixpoint computation that converge to $\overrightarrow{\boldsymbol{\mathcal{FP}}}$ exactly coincide with the iterates computed by the `Kleene` procedure of the state-based algorithm FAIncS. In particular, if $\overrightarrow{\boldsymbol{Y}}$ is the output vector of the call to `Kleene` in FAIncS then $\overrightarrow{\boldsymbol{Y}} = \overrightarrow{\boldsymbol{\mathcal{FP}}}$. Furthermore, $\overrightarrow{\boldsymbol{\mathcal{FP}}}_q \mathrel{\widetilde{\sqsubseteq}} \{I_2^c\} \Leftrightarrow \exists S \in \overrightarrow{\boldsymbol{\mathcal{FP}}}_q,\, S \cap I_2 = \varnothing$. Summing up, the $\sqsubseteq$-lfp algorithm FAIncS coincides with the $\widetilde{\sqsubseteq}$-gfp antichain algorithm of Theorem 6.5.

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

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

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

where $\mathrm{cpre}^{\mathcal{A}_2}_u(F_2^c) = (\mathrm{pre}^{\mathcal{A}_2}_u(F_2))^c$. When using these functions, we obtain a lfp algorithm computing on the domain $\langle \mathrm{AC}_{\langle\wp(Q_2),\supseteq\rangle}, \sqsubseteq\rangle$. Indeed, it turns out that $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$ iff $\texttt{Kleene}(\lambda\vec{\boldsymbol{X}}.\, \alpha^c(\vec{\boldsymbol{\epsilon}}^{F_1}) \sqcup \mathrm{CPre}^{\mathcal{A}_2}_{\mathcal{A}_1}(\vec{\boldsymbol{X}}), \vec{\boldsymbol{\varnothing}}) \sqsubseteq \alpha^c(\overrightarrow{\boldsymbol{L_2}}^{I_1})$. It is straightforward to check that this algorithm coincides with the backward antichain algorithm defined by De Wulf et al. [10, Algorithm 1, Theorem 6] since both compute on the same domain, $\lfloor X\rfloor$ 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. [10] can be systematically derived by instantiating our framework. The original antichain algorithms were later improved by Abdulla et al. [2] and, subsequently, by Bonchi and Pous [4]. 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 pairs in the relation. When that binary relation is a simulation relation, our framework allows to partially match their results by using the quasiorder $\preceq^r_{\mathcal{A}}$ defined in Sect. 5.3. However, this quasiorder relation $\preceq^r_{\mathcal{A}}$ does not consider pairs of states $Q_2 \times Q_2$ whereas the aforementioned works do.

7 An Equivalent Greatest Fixpoint Algorithm

Let us recall a result from Cousot [5, Theorem 4] that if $g\colon C \to C$ 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., $g(c) \leq c' \Leftrightarrow c \leq \widetilde{g}(c')$ holds) then the following equivalence holds: for all $c, c' \in C$,

$$\mathrm{lfp}(\lambda x.\, c \vee g(x)) \leq c' \;\Leftrightarrow\; c \leq \mathrm{gfp}(\lambda y.\, c' \wedge \widetilde{g}(y)). \tag{11}$$

This property has been exploited to derive equivalent least/greatest fixpoint-based invariance proof methods for programs [5]. In the following, we use (11) 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 provides an adjunction between concatenation and quotients of sets of words.

Lemma 7.1. *For all $X, Y \subseteq \Sigma^*$ and $w \in \Sigma^*$, $wY \subseteq Z \Leftrightarrow Y \subseteq w^{-1}Z$ and $Yw \subseteq Z \Leftrightarrow Y \subseteq Zw^{-1}$.*

Given the set of equations induced by a FA $\mathcal{A} = \langle Q, \delta, I, F, \Sigma\rangle$, we define the function $\widetilde{\mathrm{Pre}}_{\mathcal{A}} : \wp(\Sigma^*)^{|Q|} \to \wp(\Sigma^*)^{|Q|}$ as follows:

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

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

Lemma 7.2. *For all* $\vec{\boldsymbol{X}}, \vec{\boldsymbol{Y}} \in \wp(\Sigma^*)^{|Q|}$, $\mathrm{Pre}_{\mathcal{A}}(\vec{\boldsymbol{X}}) \subseteq \vec{\boldsymbol{Y}} \Leftrightarrow \vec{\boldsymbol{X}} \subseteq \widetilde{\mathrm{Pre}}_{\mathcal{A}}(\vec{\boldsymbol{Y}})$.

Hence, from equivalences (5) and (11) we obtain:

$$\mathcal{L}(\mathcal{A}_1) \subseteq L_2 \;\Leftrightarrow\; \vec{\boldsymbol{\epsilon}}^{F_1} \subseteq \mathrm{gfp}(\lambda \vec{\boldsymbol{X}}.\, \vec{\boldsymbol{L_2}}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\vec{\boldsymbol{X}})). \tag{12}$$

The following algorithm `FAIncGfp` decides the inclusion $\mathcal{L}(\mathcal{A}_1) \subseteq L_2$ by implementing the greatest fixpoint computation in equivalence (12). The intuition behind algorithm `FAIncGfp` is that

$$L_1 \subseteq L_2 \Leftrightarrow \forall w \in L_1,\ (\epsilon \in w^{-1}L_2 \Leftrightarrow \epsilon \in \textstyle\bigcap_{w \in L_1} w^{-1}L_2),$$

where $L_1 = \mathcal{L}(\mathcal{A}_1)$. Therefore, `FAIncGfp` computes the set $\bigcap_{w \in L_1} w^{-1}L_2$ by using the automaton $\mathcal{A}_1$ and by considering prefixes of L_1 of increasing lengths. This means that after n iterations of `Kleene`, the algorithm `FAIncGfp` has computed $\bigcap_{wu \in L_1, |w| \leq n, q_0 \in I_1, q_0 \overset{w}{\rightsquigarrow} q} w^{-1}L_2$, for every state $q \in Q_1$.

`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 $L_2$.
1 $\langle Y_q \rangle_{q \in Q}$ := Kleene($\lambda \vec{\boldsymbol{X}}.\, \vec{\boldsymbol{L_2}}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\vec{\boldsymbol{X}}), \vec{\boldsymbol{\Sigma^*}}$);
2 forall the $q \in F_1$ do
3     if $\epsilon \notin Y_q$ then return false;
4 return true;
```

The regularity of L_2 clanguages of being closed under intersections and quotients show that each iterate computed by `Kleene`$(\lambda \vec{\boldsymbol{X}}.\, \vec{\boldsymbol{L_2}}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\vec{\boldsymbol{X}}), \vec{\boldsymbol{\Sigma^*}})$ is a (computable) regular language. To the best of our knowledge, this language inclusion algorithm `FAIncGfp` has never been described in the literature before.

Next, we discharge the fundamental assumption on which the correctness of this algorithm `FAIncGfp` depends on: the Kleene iterates computed by `FAIncGfp` are finitely many. In order to do that, we consider an abstract version of the greatest fixpoint computation exploiting a closure operator which guarantees 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 $\lambda \vec{\boldsymbol{X}}.\, \vec{\boldsymbol{L_2}}^{I_1} \cap \widetilde{\mathrm{Pre}}_{\mathcal{A}_1}(\vec{\boldsymbol{X}})$ used by `FAIncGfp`.

Forward completeness of abstract interpretations [15] is different from and orthogonal to backward completeness introduced in Sect. 3 and used in Sects. 4 and 5. 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 Kleene iterates of $\operatorname{gfp}(\overrightarrow{\boldsymbol{X}}.\, \overrightarrow{\boldsymbol{L_2}}^{I_1} \cap \widetilde{\operatorname{Pre}}_{\mathcal{A}_1}(\overrightarrow{\boldsymbol{X}}))$ 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 [1,13].

Let us now describe in detail this abstraction. A closure $\rho \in \operatorname{uco}(C)$ 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, f maps elements of ρ into elements of ρ. Dually to the case of backward completeness, forward completeness implies that $\operatorname{gfp}(f) = \operatorname{gfp}(f\rho) = \operatorname{gfp}(\rho f \rho)$ holds, when these greatest fixpoints exist (this is the case, e.g., when C is a complete lattice). It turns out that forward and backward completeness are related by the following duality on the function f.

Lemma 7.3 ([15, Corollary 1]). *Let $\langle C, \leq_C \rangle$ be a complete lattice and assume that $f : C \to C$ admits the right-adjoint $\widetilde{f} : C \to C$, i.e., $f(c) \leq_C c' \Leftrightarrow c \leq_C \widetilde{f}(c')$ holds. Then, ρ is backward complete for f iff ρ is forward complete for $\widetilde{f}$.*

Thus, by Lemma 7.3, in the following result instead of assuming the hypotheses implying that a closure ρ is forward complete for the right-adjoint $\widetilde{\operatorname{Pre}}_{\mathcal{A}_1}$ we state some hypotheses which guarantee that ρ is backward complete for its left-adjoint $\operatorname{Pre}_{\mathcal{A}_1}$.

Theorem 7.4. *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 $\rho \in \operatorname{uco}(\wp(\Sigma^*))$. 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 $\overrightarrow{\boldsymbol{\epsilon}}^{F_1} \subseteq \operatorname{gfp}(\overrightarrow{\boldsymbol{X}}.\, \rho(\overrightarrow{\boldsymbol{L_2}}^{I_1} \cap \widetilde{\operatorname{Pre}}_{\mathcal{A}_1}(\overrightarrow{\boldsymbol{X}})))$. Moreover, the Kleene iterates computed by $\operatorname{gfp}(\overrightarrow{\boldsymbol{X}}.\, \rho(\overrightarrow{\boldsymbol{L_2}}^{I_1} \cap \widetilde{\operatorname{Pre}}_{\mathcal{A}_1}(\overrightarrow{\boldsymbol{X}})))$ coincide in lockstep with those of $\operatorname{gfp}(\overrightarrow{\boldsymbol{X}}.\, \overrightarrow{\boldsymbol{L_2}}^{I_1} \cap \widetilde{\operatorname{Pre}}_{\mathcal{A}_1}(\overrightarrow{\boldsymbol{X}}))$.

We can now establish that the sequence of Kleene iterates computed by $\operatorname{gfp}(\overrightarrow{\boldsymbol{X}}.\, \overrightarrow{\boldsymbol{L_2}}^{I_1} \cap \widetilde{\operatorname{Pre}}_{\mathcal{A}_1}(\overrightarrow{\boldsymbol{X}}))$ is finite. 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 (9). Lemma 5.7 tells us that $\leq^l_{\mathcal{A}_2}$ 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, $\langle \rho_{\leq^l_{\mathcal{A}_2}}, \subseteq \rangle$ is a poset which is both ACC and DCC. In particular, the definition of $\leq^l_{\mathcal{A}_2}$ implies that every chain in $\langle \rho_{\leq^l_{\mathcal{A}_2}}, \subseteq \rangle$ has at most $2^{|Q_2|}$ elements, so that if we compute $2^{|Q_2|}$ Kleene iterates then we have necessarily computed the greatest fixpoint. Moreover, as a consequence of the DCC property we have that the Kleene iterates of $\operatorname{gfp}(\lambda\overrightarrow{\boldsymbol{X}}.\, \rho_{\leq_{\mathcal{A}_2}}(\overrightarrow{\boldsymbol{L_2}}^{I_1} \cap \widetilde{\operatorname{Pre}}_{\mathcal{A}_1}(\overrightarrow{\boldsymbol{X}})))$ are finitely many, hence so are the iterates of $\operatorname{gfp}(\lambda\overrightarrow{\boldsymbol{X}}.\, \overrightarrow{\boldsymbol{L_2}}^{I_1} \cap \widetilde{\operatorname{Pre}}_{\mathcal{A}_1}(\overrightarrow{\boldsymbol{X}}))$ because they go in lockstep as stated by Theorem 7.4.

Corollary 7.5. *Let $\mathcal{A}_1$ be a FA and let L_2 be a regular language. Then, the algorithm* `FAIncGfp` *decides the inclusion $\mathcal{L}(\mathcal{A}_1) \subseteq L_2$*

Finally, it is worth citing that Fiedor et al. [12] 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 [22]: $\mathrm{lfp}(\lambda X.\, f(X)) = (\mathrm{gfp}(\lambda X.\, (f(X^c)^c))^c$. Their approach differs from ours since we use the equivalence (11) 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.

8 Conclusion and 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 within our framework well-known algorithms, such as the antichain algorithms by De Wulf et al. [10], as well as novel algorithms, such as `FAIncGfp`, for deciding the inclusion of regular languages. Due to lack of space, we deliberately omitted from this paper the study of the inclusion problem $\mathcal{L}(\mathcal{G}) \subseteq L$ where $\mathcal{G}$ is a context-free grammar, in exchange for a deeper understanding of the $\mathcal{L}(\mathcal{A}) \subseteq L$ case. The case $\mathcal{L}(\mathcal{G}) \subseteq \mathcal{L}(\mathcal{A})$ is covered in the extended version [14].

Future directions include leveraging well-quasiorders for infinite words [21] to shed new light on the inclusion problem between ω languages. Our results could also be extended to inclusion of tree languages by relying on the extensions of Myhill-Nerode theorems for tree languages [20]. 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. [2] and Bonchi and Pous [4]. Finally, let us mention that the correspondence between least and greatest fixpoint-based inclusion checks assuming complete abstractions was studied by Bonchi et al. [3] 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 point of view of sound up-to techniques.

Acknowledgements. We would like to thank the reviewers for their insightful feedback that allowed us to find a simpler connection between our work and the antichain algorithms. 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, SID2018 project "Analysis of STatic Analyses (ASTA)", and by the Italian Ministry of Research MIUR, project No. 201784YSZ5 "AnalysiS of PRogram Analyses (ASPRA)".

References

1. Abdulla, P.A., Cerans, K., Jonsson, B., Tsay, Y.-K.: General decidability theorems for infinite-state systems. In: Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS 1996), pp. 313–321. IEEE Computer Society (1996)
2. Abdulla, P.A., Chen, Y.-F., Holík, L., Mayr, R., Vojnar, T.: When simulation meets antichains. In: Esparza, J., Majumdar, R. (eds.) TACAS 2010. LNCS, vol. 6015, pp. 158–174. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12002-2_14
3. Bonchi, F., Ganty, P., Giacobazzi, R., Pavlovic, D.: Sound up-to techniques and complete abstract domains. In: Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2018). ACM Press (2018)
4. Bonchi, F., Pous, D.: Checking NFA equivalence with bisimulations up to congruence. In: Proceedings of the 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2013), pp. 457–468. ACM Press (2013)
5. Cousot, P.: Partial completeness of abstract fixpoint checking. In: Choueiry, B.Y., Walsh, T. (eds.) SARA 2000. LNCS (LNAI), vol. 1864, pp. 1–25. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44914-0_1
6. Cousot, P., Cousot, R.: 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 1977), pp. 238–252. ACM Press (1977)
7. Cousot, P., Cousot, R.: Systematic design of program analysis frameworks. In: Proceedings of the 6th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL 1979), pp. 269–282. ACM, New York (1979)
8. de Luca, A., Varricchio, S.: Well quasi-orders and regular languages. Acta Informatica **31**(6), 539–557 (1994)
9. de Luca, A., Varricchio, S.: Finiteness and Regularity in Semigroups and Formal Languages. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-59849-4
10. De Wulf, M., Doyen, L., Henzinger, T.A., Raskin, J.-F.: Antichains: a new algorithm for checking universality of finite automata. In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 17–30. Springer, Heidelberg (2006). https://doi.org/10.1007/11817963_5
11. Ehrenfeucht, A., Haussler, D., Rozenberg, G.: On regularity of context-free languages. Theor. Comput. Sci. **27**(3), 311–332 (1983)
12. Fiedor, T., Holík, L., Lengál, O., Vojnar, T.: Nested antichains for WS1S. Acta Informatica **56**(3), 205–228 (2019)
13. Finkel, A., Schnoebelen, P.: Well-structured transition systems everywhere!. Theor. Comput. Sci. **256**(1–2), 63–92 (2001)
14. Ganty, P., Ranzato, F., Valero, P.: Complete abstractions for checking language inclusion. arXiv e-prints, arXiv:1904.01388, April 2019
15. Giacobazzi, R., Quintarelli, E.: Incompleteness, counterexamples, and refinements in abstract model-checking. In: Cousot, P. (ed.) SAS 2001. LNCS, vol. 2126, pp. 356–373. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-47764-0_20
16. Giacobazzi, R., Ranzato, F., Scozzari, F.: Making abstract interpretations complete. J. ACM **47**(2), 361–416 (2000)

17. Hofman, P., Totzke, P.: Trace inclusion for one-counter nets revisited. Theor. Comput. Sci. **735**, 50–63 (2018)
18. Hofmann, M., Chen, W.: Abstract interpretation from Büchi automata. In: Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL 2014) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2014). ACM Press (2014)
19. Jančar, P., Esparza, J., Moller, F.: Petri nets and regular processes. J. Comput. Syst. Sci. **59**(3), 476–503 (1999)
20. Kozen, D.: On the Myhill-Nerode theorem for trees. Bull. EATCS **47**, 170–173 (1992)
21. Ogawa, M.: Well-quasi-orders and regular ω-languages. Theor. Comput. Sci. **324**(1), 55–60 (2004)
22. Park, D.: Fixpoint induction and proofs of program properties. Mach. Intell. **5**, 59–78 (1969)
23. Ranzato, F.: Complete abstractions everywhere. In: Giacobazzi, R., Berdine, J., Mastroeni, I. (eds.) VMCAI 2013. LNCS, vol. 7737, pp. 15–26. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35873-9_3
24. Schützenberger, M.P.: On context-free languages and push-down automata. Inf. Control **6**(3), 246–264 (1963)