Decidability of Identity-free Relational Kleene Lattices

Paul Brunet, Damien Pous

Decidability of Identity-free Relational Kleene Lattices

Paul Brunet & Damien Pous

Plume team – LIP, CNRS, ENS de Lyon, Inria, UCBL, Université de Lyon, UMR 5668

Abstract

Families of binary relations are important interpretations of regular expressions, and the equivalence of two regular expressions with respect to their relational interpretations is decidable: the problem reduces to the equality of the denoted regular languages.

Putting together a few results from the literature, we first make explicit a generalisation of this reduction, for regular expressions extended with converse and intersection: instead of considering sets of words (i.e., formal languages), one has to consider sets of directed and labelled graphs.

We then focus on identity-free regular expressions with intersection—a setting where the above graphs are acyclic—and we show that the corresponding equational theory is decidable. We achieve this by defining an automaton model, based on Petri Nets, to recognise these sets of acyclic graphs, and by providing an algorithm to compare such automata.

Introduction

Binary relations appear everywhere in mathematics and computer science, together with the operations of union ($\cup$), intersection ($\cap$), composition ($\circ$), converse ($.^\vee$), reflexive-transitive closure ($.^\star$), and the constants identity (Id) and empty relation ($\varnothing$). As such, an algorithm for deciding the equivalence of expressions built with these operators with respect to their relational interpretations is a very desirable goal. However such an algorithm has yet to be found.

Regular expressions [6], where only the operators $\cup, \circ, .^\star, \mathrm{Id}$, and $\varnothing$ are allowed, are the most famous example of a decidable fragment [9]. In this setting, it is now well-known that the equivalence of two expressions in all relational interpretations is equivalent to the equality of the regular languages denoted by these expressions in the usual sense (the letter x is interpreted as $\{x\}$). Several equational or semi-equational theories are known to be complete for this fragment [10–12].

The converse operation can also be added to regular expressions, and the resulting theory remains decidable (see [3, 7] or [4]). In this case, decidability is obtained by 1) reducing the problem of equivalence of two expressions to the equality of some regular sets of words over an extended alphabet, and 2) defining automata constructions to recognise these sets.

Freyd and Scedrov sketched an algorithm for representable allegories [8, page 208], that is, expressions with composition, intersection, converse, and identity, but without union or reflexive-transitive closure. Similar constructions were given independently by Andréka and Bredikhin [2], in a more comprehensive way. The key idea is the following: if we restrict ourselves to the above syntax (variables, composition, intersection, converse, identity), we get what is called *ground terms*. Such a term u can be represented as labelled directed graphs $G(u)$ with two distinguished vertices called the *input* and the *output*. A variable a corresponds to a graph with one edge labelled by a linking the input to the output. The identity is represented by the graph with a single vertex and no edges, The composition of two graphs with disjoints sets of vertices can be performed by identifying the output of the first graph and the input of the second one. The operation corresponding to the intersection

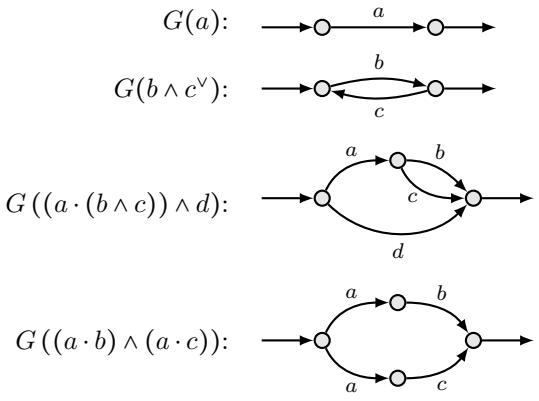


Figure 1: Graphs associated to some ground terms

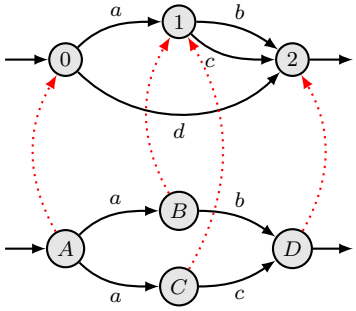


Figure 2: A graph homomorphism.

consists in merging the inputs of the two graphs, as well as their outputs. And finally, converse is obtained by swapping the inputs and the outputs. Some examples are given in Figure 1.

These graphs can be endowed with a preorder relation $G \blacktriangleleft F$, defined by the existence of a graph homomorphism from F to G (preserving inputs and outputs). For instance the graph corresponding to $(a \cdot (b \wedge c)) \wedge d$ is smaller than the graph of $(a \cdot b) \wedge (a \cdot c)$, thanks to the homomorphism depicted in Figure 2 using dotted arrows. Notice that this preorder has nothing to do with the respective sizes of the graphs: a graph may very well be smaller (in the sense of $\blacktriangleleft$) than another while having more vertices (and vice versa). The key result from Freyd and Scedrov [8, page 208], or Andréka and Bredikhin [2, Theorem 1], is that for any two ground terms u, v, u is contained in v under any relational interpretation if and only if $G(u) \blacktriangleleft G(v)$.

This is for ground terms; to handle the whole syntax, we need to add union and reflexive-transitive closure. It suffices for that to consider sets of graphs: to each expression e, one can associate a set of graphs $G(e)$. Writing $X^{\blacktriangleleft}$ for the downward closure of a set of graphs X by the relation $\blacktriangleleft$, we obtain the following generalisation of the above result: for any two expressions e and f, e is contained in f under any relational interpretation if and only if $G(e) \subseteq G(f)^{\blacktriangleleft}$. (Theorem 6 in the sequel—this result is almost there in the work by Andréka et al. [1], but this explicit formulation is new, to the best of our knowledge.)

This result encompasses the case of plain regular expressions, whose graphs are just words and for which the preorder $\blacktriangleleft$ reduces to isomorphism, but also the case of regular expressions with converse, whose graphs are words over a duplicated alphabet and for which the preorder $\blacktriangleleft$ can be reformulated in terms of the rewriting system proposed by Ésik et al. [3, 7].

Our main contribution is then to exploit this characterisation to obtain decidability for identity-free regular expressions with intersection, whose equational theory has been studied by Andréka et al. [1]. The reason why we need to exclude identity and converse is that in presence of intersection, they yield cyclic graphs, and we do not know how to handle such graphs. We hope to get rid of this assumption in future work.

The key concept which we introduce is a new kind of finite automaton, allowing us to recognise sets of graphs that are downward-closed w.r.t. the graph embedding relation $\blacktriangleleft$. To give some intuition about this automaton model, let us look at the example from Figure 2, and try to build sequentially a morphism h from $F = G((a \cdot b) \wedge (a \cdot c))$ to $G = G((a \cdot (b \wedge c)) \wedge d)$.

- We start by placing a token ⓐ on A. We know that for h to be a morphism, it has to preserve the input of the graph, so we map A to position 0 in G.

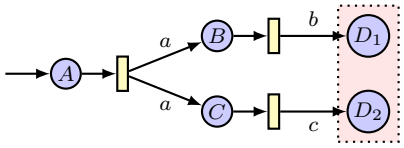


Figure 3: The automaton corresponding to the term $(a \cdot b) \wedge (a \cdot c)$.

- There are two outgoing edges from A, both labelled by a. We split token ⓐ into Ⓑ and Ⓒ, and move Ⓑ to position B and Ⓒ to position C. We then map the positions of both tokens to position 1 in G, which is consistent with h being a morphism, thanks to the arc $(0, a, 1)$.
- Now we try to move Ⓑ. B has one outgoing edge, labelled by b. We may move Ⓑ to D, and using the arc $(1, b, 2)$ in G we map D to position 2.
- Then we can look at Ⓒ. We have to move it to position D, and thanks to the arc $(1, c, 2)$ we can confirm the map of D to 2, and merge back Ⓑ and Ⓒ.

At the end, we have only one token, placed on the output of F, and during the procedure we have mapped all positions in F to positions in G, while preserving all labelled edges.

This kind of procedure is reminiscent of Petri nets [13–15]: at each step we relate tokens to positions in G, and fire transitions according to the edges of G. This is the basic idea behind the notion of Petri automata which we introduce in Section 2. For instance the Petri automaton we will construct for the term $(a \cdot b) \wedge (a \cdot c)$ is depicted in Figure 3, and the procedure sketched above can then be formally described as a reading of the graph G in this automaton.

Given an expression e, we show in Section 3 how to build a Petri automaton that recognises exactly the graphs in $G(e)^{\blacktriangleleft}$. We then show in Section 4 how to compare Petri automata. Several difficulties arise, that do not appear with classical word automata. Our solution nevertheless uses a standard coinductive approach, where we define an appropriate notion of simulation.

1. Expressions and languages

In this section we consider the full signature $\langle \wedge, \vee, \cdot, \cdot^\star, \cdot^\vee, \mathbb{0}, \mathbb{1} \rangle$ of Kleene lattices with conversion. We fix a set X of variables, and we denote by $\mathrm{Reg}_X^{\vee\wedge}$ the set of expressions build from variables in X with these connectives. These expressions are meant to be interpreted in relational models: $\cdot$ corresponds to the composition of relations; $\vee$ to the union; $\wedge$ to the intersection; R^* to the reflexive transitive closure of a relation R; and $R^\vee$ to the converse of R. The constants $\mathbb{0}$ and $\mathbb{1}$ are respectively interpreted as the empty relation and the identity relation. For any set S, we write $\mathcal{P}(S) \coloneqq \{P \mid P \subseteq S\}$ for the set of subsets of S. Let $A \to B$ be the functions from A to B and $A \dashrightarrow B$ the partial maps from A to B. $\mathrm{dom}(f)$ denotes the domain of a partial map f. If $\sigma : X \to \mathcal{P}(S \times S)$ is an interpretation of the alphabet X into some space of relations, we write $\widehat{\sigma}$ for the unique homomorphism extending σ from $\mathrm{Reg}_X^{\vee\wedge}$ to $\mathcal{P}(S \times S)$. We say that two expressions e and f are relationally equivalent, written $Rel \vDash e = f$, if for any relational interpretation σ we have $\widehat{\sigma}(e) = \widehat{\sigma}(f)$. Similarly, we write $Rel \vDash e \leqslant f$ if $\widehat{\sigma}(e) \subseteq \widehat{\sigma}(f)$ holds for any σ.

The *ground terms* are defined by the following sub-syntax:

$$u, v, w \in W_X \coloneqq x \in X \mid w \cdot w \mid w \wedge w \mid w^\vee \mid \mathbb{1} \ .$$

We let G range over 2-pointed labelled directed graphs, which we simply call *graphs* in the sequel. Those are tuples $\langle V, E, \iota, o \rangle$ with V a finite set of vertices, $E \subseteq V \times X \times V$ a set of edges labelled with X, and $\iota, o \in V$ the two distinguished vertices, respectively called *input* and *output*.

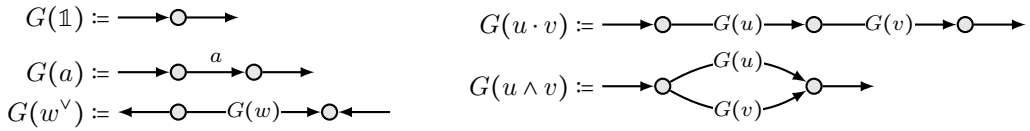


Figure 4: Graphs corresponding to ground terms.

To each ground term w, we associate such a graph $G(w)$. The graph for $\mathbb{1}$ has only one vertex, both input and output. The graph of a has one edge labelled by a linking its input to its output. The composition of two graphs with disjoint sets of vertices can be performed by identifying the output of the first graph and the input of the second one. The intersection on graphs consists in merging their inputs and merging their outputs. The converse consists simply in exchanging the input and the output of a graph. See Figure 4 for a graphical description of this construction. Those graphs were introduced independently by Freyd and Scedrov [8, page 208], and Andréka and Bredikhin [2].

Another useful notion is the notion of morphism between graphs:

Definition 1 (Graph morphism, preorder on ground terms)
A *graph morphism* from $\langle V_1, E_1, \iota_1, o_1\rangle$ to $\langle V_2, E_2, \iota_2, o_2\rangle$ is a map $h : V_1 \to V_2$ such that $h(\iota_1) = \iota_2$, $h(o_1) = o_2$, and $(p, x, q) \in E_1$ entails $(h(p), x, h(q)) \in E_2$. We denote by $\blacktriangleleft$ the relation on graphs defined by $G \blacktriangleleft G'$ if there exists a graph morphism from G' to G. This relation gives rise to a preorder on ground terms, written $\lhd$ and defined by $u \lhd v$ if $G(u) \blacktriangleleft G(v)$. *

Given a set S of graphs, we write $S^{\blacktriangleleft}$ for its downward closure w.r.t. $\blacktriangleleft$: $S^{\blacktriangleleft} \coloneqq \{G \mid G \blacktriangleleft G', G' \in S\}$. Similarly, we write $S^{\lhd}$ for the downward closure of a set of ground terms w.r.t. $\lhd$.

As explained in the introduction, the above preorder on ground terms precisely characterises inclusion under arbitrary relational interpretations:

Theorem 2 ([2, Theorem 1], or [8, page 208]). *For all ground terms $u, v \in W_X$, we have*

$$\mathit{Rel} \vDash u \leqslant v \Leftrightarrow u \lhd v \ .$$

To extend this result to the expressions we consider in this paper, we introduce the following generalisation of the language of a regular expression. Sets of words become sets of ground terms.

Definition 3 (Term language of an expression)
The *term language* denoted by an expression $e \in \mathrm{Reg}_X^{\vee\wedge}$, written $[\![e]\!]$, is the set of ground terms defined inductively as follows:

$$\begin{aligned}
[\![x]\!] &\coloneqq \{x\} & [\![e \cdot f]\!] &\coloneqq \{w \cdot w' \mid w \in [\![e]\!] \text{ and } w' \in [\![f]\!]\} \\
[\![e \vee f]\!] &\coloneqq [\![e]\!] \cup [\![f]\!] & [\![e \wedge f]\!] &\coloneqq \{w \wedge w' \mid w \in [\![e]\!] \text{ and } w' \in [\![f]\!]\} \\
[\![e^\star]\!] &\coloneqq \textstyle\bigcup_{n\in\mathbb{N}} \{w_1 \cdot \cdots \cdot w_n \mid \forall i, w_i \in [\![e]\!]\} & [\![e^\vee]\!] &\coloneqq \{w^\vee \mid w \in [\![e]\!]\} \\
[\![\mathbb{1}]\!] &\coloneqq \{\mathbb{1}\} & [\![\mathbb{0}]\!] &\coloneqq \varnothing \ .
\end{aligned}$$

*

We need a slight refinement of a lemma established by Andréka, Mikulás, and Németi [1]:

Lemma 4. *For all expression $e \in Reg_X^{\vee\wedge} X$, and all relational interpretations $\sigma : X \to \mathcal{P}(S \times S)$, we have*

$$\widehat{\sigma}(e) = \bigcup_{w\in[\![e]\!]} \widehat{\sigma}(w) = \bigcup_{w\in[\![e]\!]^{\lhd}} \widehat{\sigma}(w) \ .$$

Proof. The first equality is exactly [1, Lemma 2.1]; for the second one, we use the fact that $\widehat{\sigma}(w) \subseteq \widehat{\sigma}(u)$ whenever $w \lhd u$, thanks to Theorem 2 (i.e., [2, Theorem 1]). □

The above definitions make it possible to characterise inclusion under all relational interpretation in terms of downward-closed term languages. To obtain decidability, we need to go one step further, by considering graph languages.

Definition 5 (Graph language of an expression)
The *graph language* of an expression e, denoted by $G(e)$ is the set of graphs associated to the ground terms in $[\![e]\!]$: $G(e) \coloneqq \{G(w) \mid w \in [\![e]\!]\}$. *

We finally obtain the following characterisation, which allows us to reduce validity in *Rel* to an equality of graph languages.

Theorem 6. *The following properties are equivalent, for all expressions $e, f \in \mathrm{Reg}_X^{\vee\wedge}$:*

(i) $\mathit{Rel} \vDash e = f$,

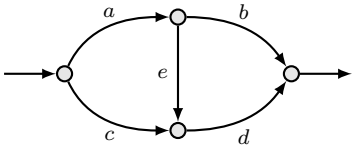
(ii) $[\![e]\!]^{\lhd} = [\![f]\!]^{\lhd}$,

(iii) $G(e)^{\blacktriangleleft} = G(f)^{\blacktriangleleft}$.

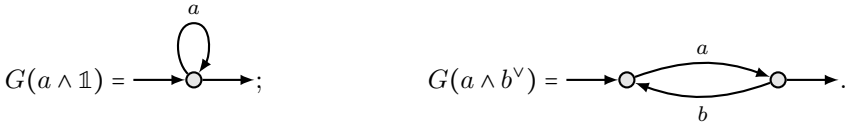
Proof. The implication (ii) ⇒ (i) follows easily from Lemma 4, and (iii) ⇒ (ii) is a matter of unfolding definitions. For (i) ⇒ (iii), we mainly use [2, Lemma 3]. □

The above statement can also be reformulated in terms of inclusions, to match the result announced in the introduction: $\mathit{Rel} \vDash e \leqslant v$ if and only if $[\![e]\!] \subseteq [\![f]\!]^{\lhd}$ if and only if $G(e) \subseteq G(f)^{\blacktriangleleft}$.

Also notice that while by definition $G(e)$ only contains graphs emanating from ground terms, this is not the case for its closure $G(e)^{\blacktriangleleft}$. For instance, $G((a \cdot b) \wedge (c \cdot d))^{\blacktriangleleft}$ contains the following graph, which is not the graph of any ground term.



The above result holds for the whole syntax of regular expressions with converse and intersection. However, in the remainder of the paper, we have to focus on expressions without converse and identity. This is because in combination with intersection, these two operations introduce cycles in the graphs associated to ground terms. Consider for instance the graphs for $a \wedge \mathbb{1}$ and $a \wedge b^{\vee}$:



Since reflexive-transitive closure $(\cdot^{\star})$ implicitly contains an occurrence of the identity, we also have to replace this operator with transitive closure $(\cdot^{+})$. We thus work we expressions from $\mathrm{Reg}_X^{\wedge-}$, defined with the following syntax: $e, f \in \mathrm{Reg}_X^{\wedge-} \coloneqq x \in X \mid e \wedge f \mid e \vee f \mid e \cdot f \mid e^{+} \mid \mathbb{0}$. Accordingly, ground terms are restricted to the following syntax: $u, v, w \in W_X^{-} \coloneqq x \in X \mid w \cdot w \mid w \wedge w$.

2. Petri Automata

Before getting to our definition of automata, we recall the standard notion of Petri net.

Definition 7 (Petri Net)
A *Petri Net* is a structure $N = \langle P, T, F, W, M_0 \rangle$ where:

- P and T are finite disjoints sets, respectively of *places* and *transitions*;
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs, called the *flow relation*;
- $W : (P \times T) \cup (T \times P) \to \mathbb{N}$ is a weight function, such that $W(f) = 0$ if $f \notin F$;
- $M_0 : P \to \mathbb{N}$ is the initial marking.

Given a marking M in a net N, a transition $\tau \in T$ is *enabled* if for any place p such that (p, τ) is in the flow relation F, we have $W(p, \tau) \leqslant M(p)$. In that case, τ can *fire*, and it results in a new marking M' such that $M'(p) = M(p) - W(p, \tau) + W(\tau, p)$. A marking is called *accessible* if it can be obtained by successively firing transitions starting from M_0. ∗

To present examples in a simple way, we use the standard graphical representation of Petri nets: they are represented as graphs, with round nodes for places and rectangular nodes for transitions. The flow relation is simply represented by arrows; the places appearing in the initial marking have an additional incoming arrow.

A Petri net is said to be *one-bounded* if in any accessible marking M, no place is marked with more than one token. Such markings can be seen as finite sets of places; we call them *configurations* in the sequel, and we let ξ, Ξ range over them. Bounded nets form an interesting class of nets, because many problems which are undecidable in the general case become decidable in this setting: whereas general Petri nets have an infinite set of accessible markings, they are finitely many in a bounded net.

Our notion of *Petri Automata* is defined below. The main difference with regular Petri nets is the labelling with letters from the alphabet X of all arcs coming out of transitions. Note that this slightly differs from the usual notion of *labelled Petri net*, where the labels are put on transitions.

Definition 8 (Petri Automaton)
A *Petri automaton* is a structure $\langle N, L, \mathscr{F} \rangle$ where:

- $N = \langle P, T, F, W, M_0 \rangle$ is a *one-bounded* Petri net such that:
  - the weight $W(f)$ of all arcs f that appear in F is equal to 1;
  - all transitions $\tau \in T$, have at least one incoming arc and one outgoing arc, meaning that there are places $p, q \in P$ such that $(p, \tau) \in F$ and $(\tau, q) \in F$;
  - there is an initial place ι such that M_0 contains only one token, placed in ι.
- $L : (T \times P) \cap F \to X$ is a labelling function;
- $\mathscr{F} \subseteq \mathcal{P}(P)$ is a set of final configurations. ∗

A transition τ in a Petri automaton can be alternatively described by a pair $[\tau] = (s, t)$ where:

- $s = \{ p \mid (p, \tau) \in F \}$ is the *input of* τ, often denoted by ${}^\bullet\tau$, and
- $t = \{ (x, q) \mid (\tau, q) \in F \text{ and } x = L(\tau, q) \}$ is the *output of* τ. Notice that this differs from the usual notion of output of a transition in a Petri net: in the usual setting $\tau^\bullet$ is just the set of places reachable from the τ. Here we add to each place the label of the arc reaching it.

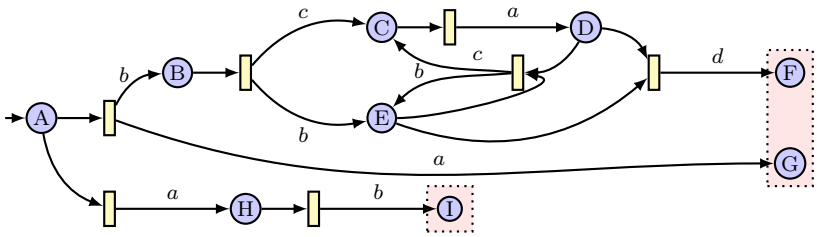


Figure 5: A Petri automaton. The initial state is A, and the final configurations are $\{I\}$ and $\{F, G\}$.

For commodity reasons, we will thus define Petri automata using quadruples $\mathscr{A} = \langle P, \mathscr{T}, \iota, \mathscr{F} \rangle$ with ι the initial place and $\mathscr{T} = \{(s,t) \mid \exists \tau \in T : [\tau] = (s,t)\}$.

Graphical representations of such automata are given in Figures 3 and 5. In these drawings a final configuration is represented by a dotted rectangle around the places contained in this configuration. Now we explain how to use Petri automata to define languages of graphs. We first describe what is a *run* of an automaton, and then how to use runs to read graphs.

Let $\mathscr{A} = \langle P, \mathscr{T}, \iota, \mathscr{F} \rangle$ be a Petri automaton. We write $\xi \xrightarrow{\tau}_{\mathscr{A}} \xi'$ when the configuration ξ' can be obtained by firing some transition τ in the configuration ξ. A set of transitions $T \subseteq \mathscr{T}$ is called *compatible* if their inputs are pairwise disjoint. If furthermore all transitions in T are enabled in a configuration ξ, one can observe that the configuration ξ' reached after firing them successively does not depend on the order in which they are fired. In that case we write $\xi \xrightarrow{T}_{\mathscr{A}} \xi'$.

Definition 9 (Run, accepting run, parallel run)
A *run* is a sequence $\xi = \langle (\xi_k)_{0 \leqslant k \leqslant n}, (\tau_k)_{0 \leqslant k < n} \rangle$ of configurations and transitions, such that $\xi_k \subseteq P$, $\tau_k \in \mathscr{T}$ and $\forall k < n,\ \xi_k \xrightarrow{\tau_k} \xi_{k+1}$. When $\xi_0 = \{\iota\}$ and $\xi_n \in \mathscr{F}$, we call ξ an *accepting run*.

A *parallel run* is defined similarly, as a sequence $\Xi = \langle (\Xi_k)_{0 \leqslant k \leqslant n}, (T_k)_{0 \leqslant k < n} \rangle$, where the $T_k \subseteq \mathscr{T}$ are compatible sets of transitions such that $\Xi_k \xrightarrow{T_k} \Xi_{k+1}$. ∗

(Note that a run ξ is uniquely determined by ξ_0 and the sequence (τ_k): all subsequent configurations can be computed deterministically.)

Example 10 (An accepting run in the automaton from Figure 5)
Consider the run $\xi = \langle (\xi_0, \xi_1, \xi_2, \xi_3, \xi_4, \xi_5, \xi_6), (\tau_0, \tau_1, \tau_2, \tau_3, \tau_4, \tau_5) \rangle$, with

$$\begin{array}{rl|rl} \xi_0 = & \{A\}, & \tau_0 = & (\{A\}, \{(b,B),(a,G)\}), \\ \xi_1 = & \{B,G\}, & \tau_1 = & (\{B\}, \{(c,C),(b,E)\}), \\ \xi_2 = \xi_4 = & \{C,E,G\}, & \tau_2 = \tau_4 = & (\{C\}, \{(a,D)\}), \\ \xi_3 = \xi_5 = & \{D,E,G\}, & \tau_3 = & (\{D,E\}, \{(c,C),(b,E)\}), \\ \xi_6 = & \{F,G\}. & \tau_5 = & (\{D,E\}, \{(d,F)\}). \end{array}$$

We can easily check, using the firing rules of a Petri net that:

$$\{A\} \xrightarrow{\tau_0} \{B,G\} \xrightarrow{\tau_1} \{C,E,G\} \xrightarrow{\tau_2} \{D,E,G\} \xrightarrow{\tau_3} \{C,E,G\} \xrightarrow{\tau_4} \{D,E,G\} \xrightarrow{\tau_5} \{F,G\}.$$

As $\{A\}$ is the initial configuration and $\{F,G\} \in \mathscr{F}$, this run is accepting. It can be represented graphically as in Figure 6. ■

As in finite-state automata, we now need to specify how to read a graph in an automaton. This is done by linking the intermediate configurations of a run to vertices in the graph, and by imposing conditions to match transitions with labelled edges of the graph.

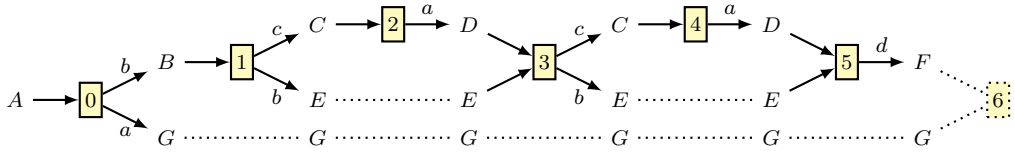


Figure 6: An accepting run in the automaton from Figure 5.

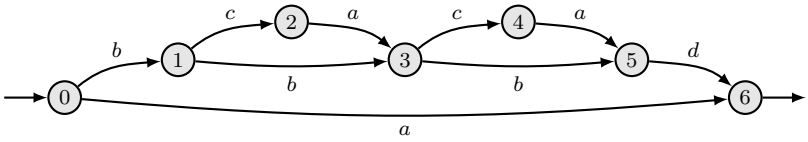


Figure 7: Trace of the run depicted in Figure 6.

Definition 11 (Reading along a run, parallel reading, language of a run)
A *reading* of $G = \langle V, E, \iota, o\rangle$ along a run $\xi = \langle (\xi_k)_{0\leqslant k\leqslant n}, (s_k, t_k)_{0\leqslant k<n}\rangle$ is a sequence $(\rho_k)_{0\leqslant k\leqslant n}$ such that for all k, ρ_k is a map from ξ_k to V_w, $\rho_0(\xi_0) = \{\iota\}$, $\rho_n(\xi_n) = \{o\}$, and $\forall k < n$, the following holds:

- all tokens in the input of the transition are mapped to the same vertex in the graph: $\forall p, q \in s_k,\ \rho_k(p) = \rho_k(q)$;
- the images of tokens in ξ_k that are not in the input of the transition are unchanged: $\forall p \in \xi_k \setminus s_k,\ \rho_k(p) = \rho_{k+1}(p)$,
- each pair in the output of the transition can be mapped to an edge of the graph with the same label: $\forall p \in s_k, \forall (x, q) \in t_k, (\rho_k(p), x, \rho_{k+1}(q)) \in E$.

Similarly, we define a *parallel reading* ρ along some parallel run $\Xi = \langle (\Xi_k)_{0\leqslant k\leqslant n}, (T_k)_{0\leqslant k<n}\rangle$ by requiring that: $\rho_0(\Xi_0) = \{\iota\}$, $\rho_n(\Xi_n) = \{o\}$, and $\forall k < n$ the following holds:

- $\forall p \in \Xi_k \setminus \bigcup_{(s,t)\in T_k} s,\ \rho_{k+1}(p) = \rho_k(p)$;
- $\forall (s, t) \in T_k, \forall p, q \in s, \rho_k(p) = \rho_k(q)$;
- $\forall (s, t) \in T_k, \forall p \in s, \forall (x, q) \in t, (\rho_k(p), x, \rho_{k+1}(q)) \in E$.

The *language of a run* ξ, denoted by $\mathscr{L}(\xi)$ is the set of graphs that can be read along ξ. ∗

The language of a Petri automaton is finally obtained by considering all accepting runs.

Definition 12 (Language recognised by a Petri automaton)
The language recognised by $\mathscr{A}$, written $\mathscr{L}(\mathscr{A})$, is the following set of graphs:

$$\mathscr{L}(\mathscr{A}) \coloneqq \bigcup_{\xi \text{ accepting in } \mathscr{A}} \mathscr{L}(\xi)\ . \qquad *$$

The language of a run ξ can be characterised using a single graph which we call the *trace* of ξ: graphs are accepted by ξ exactly when they are smaller than the trace of ξ, according to $\blacktriangleleft$ (Lemma 14 below). For instance the trace of the run presented in Figure 6 is shown in Figure 7.

This trace is constructed by creating a vertex k for each transition $\tau_k = (s_k, t_k)$ of the run, plus a final vertex n. We add an edge (k, x, l) whenever there is some place q such that $(x, q) \in t_k$, and τ_l is the first transition after τ_k in the run with q among its inputs, or $l = n$ if there is no such transition in the run. Formally:

Definition 13 (Trace of a run)
Let $\xi = \langle (\xi_k)_{0\leqslant k\leqslant n}, (s_k,t_k)_{0\leqslant k<n}\rangle$ be run. For an index $k \leqslant n$ and a place q, let $\nu(k,q)$ be either the smallest index l such that $k \leqslant l$ and $q \in s_l$, or n if there is no such index. The *trace of* ξ is the graph $\mathrm{Tr}(\xi) \coloneqq \langle \{0,\dots,n\}, E_\xi, 0, n\rangle$ with $E_\xi \coloneqq \{(k,x,\nu(k+1,q)) \mid (x,q) \in t_k\}$. We write $\mathrm{Tr}(\mathscr{A})$ for the set of traces associated to the accepting runs of $\mathscr{A}$. ∗

Lemma 14. *For any accepting run ξ, we have $G \in \mathscr{L}(\xi)$ if and only if $G \blacktriangleleft Tr(\xi)$.*

Proof. Suppose there exists a graph morphism h from $\mathrm{Tr}(\xi)$ to G. Then we can build a reading by defining $\rho_k(p) = h(\nu(k,p))$ for $0 \leqslant k \leqslant n$ and $p \in \xi_k$. On the other hand, if we have a reading $(\rho_k)_{0\leqslant k\leqslant n}$ of G, we can build a morphism h by letting $h(k) = \rho_k(p)$ for any $p \in s_k$. As $(\rho_k)_k$ is a reading, h is well defined. □

As a corollary, we obtain the following characterisation of the language of a Petri automaton.

Corollary 15. $\mathscr{L}(\mathscr{A}) = Tr(\mathscr{A})^{\blacktriangleleft}$.

The left-hand side language is defined through readings along accepting runs, which is a local and incremental notion and which allows us to define *simulations* in Section 4. By contrast, the right-hand side language is defined globally, which eases the following construction of an automaton recognising the language of an expression.

3. From expressions to automata

We now show how to associate to any expression $e \in \mathrm{Reg}^{\wedge-}_X$ an automaton $\mathscr{A}(e)$ that recognises the language $G(e)^{\blacktriangleleft}$. In fact the produced automaton has an even stronger connection with e: the graphs in $G(e)$ are exactly the traces of accepting runs in $\mathscr{A}(e)$.

Definition 16
To each expression $e \in \mathrm{Reg}^{\wedge-}_X$, we associate a Petri automaton $\mathscr{A}(e)$ defined inductively as follows:

- $\mathscr{A}(x) \coloneqq \langle \{0,1\}, \{(\{0\},\{(x,1)\})\}, 0, \{\{1\}\}\rangle$
- $\mathscr{A}(\mathbb{0}) \coloneqq \langle \{0\}, \varnothing, 0, \varnothing\rangle$
- $\mathscr{A}(e_1 \vee e_2) \coloneqq \langle P_1 \cup P_2, \mathscr{T}, \iota_1, \mathscr{F}_1 \cup \mathscr{F}_2\rangle$ with $\mathscr{T} \coloneqq \mathscr{T}_1 \cup \mathscr{T}_2 \cup \{(\{\iota_1\},t) \mid (\{\iota_2\},t) \in \mathscr{T}_2\}$.
- $\mathscr{A}(e_1 \cdot e_2) \coloneqq \langle P_1 \cup P_2, \mathscr{T}, \iota_1, \mathscr{F}_2\rangle$ with $\mathscr{T} \coloneqq \mathscr{T}_1 \cup \mathscr{T}_2 \cup \{(f,t) \mid f \in \mathscr{F}_1 \text{ and } (\{\iota_2\},t) \in \mathscr{T}_2\}$.
- $\mathscr{A}(e_1^+) \coloneqq \langle P_1, \mathscr{T}, \iota_1, \mathscr{F}_1\rangle$ with $\mathscr{T} \coloneqq \mathscr{T}_1 \cup \{(f,t) \mid f \in \mathscr{F}_1 \text{ and } (\{\iota_1\},t) \in \mathscr{T}_1\}$.
- $\mathscr{A}(e_1 \wedge e_2) \coloneqq \langle P_1 \cup P_2, \mathscr{T}, \iota_1, \mathscr{F}\rangle$ with $\mathscr{F} \coloneqq \{f_1 \cup f_2 \mid f_1 \in \mathscr{F}_1,\ f_2 \in \mathscr{F}_2\}$ and

$$\mathscr{T} \coloneqq \{(s,t) \mid i \in \{1,2\},\ (s,t) \in \mathscr{T}_i,\ \iota_i \notin s\} \cup \{(\{\iota_1\}, t_1 \cup t_2) \mid (\{\iota_1\},t_1) \in \mathscr{T}_1,\ (\{\iota_2\},t_2) \in \mathscr{T}_2\}$$

(In the inductive cases, we assume that $\mathscr{A}(e_i) = \langle P_i, \mathscr{T}_i, \iota_i, \mathscr{F}_i\rangle$ for $i \in \{1,2\}$, with $P_1 \cap P_2 = \varnothing$.) ∗

We prove by induction on e that $\mathscr{A}(e)$ is indeed a Petri automaton; for the one-boundedness assumption, we add to the induction hypothesis the fact that for any configuration ξ accessible in $\mathscr{A}(e)$, if there is a final configuration $f \in \mathscr{F}$ such that $f \subseteq \xi$, then $f = \xi$. Another invariant is that the initial place never appears in a final configuration, nor in the outputs of any transition. Note that the place ι_2 becomes unreachable by construction in the cases for union, composition and intersection, so that it could safely be removed, together with the associated transitions. One can also check that the number of places in the produced automaton is linear in the size of the input expression.

Theorem 17 (Correctness)**.** *For all expression* $e \in Reg_X^{\wedge -}$, $\mathscr{L}(\mathscr{A}(e)) = G(e)^{\blacktriangleleft}$.

Proof. As explained above, we prove a stronger result, namely $\mathrm{Tr}(\mathscr{A}(e)) = G(e)$ (up to graph isomorphisms). This allows us to conclude thanks to Corollary 15. □

Remark 18. If e is an expression without intersection, it can be shown that the transitions in $\mathscr{A}(e)$ are all of the form $(\{p\}, \{(x, q)\})$, with only one input and one output. In consequence, the accessible configurations are singletons, and the resulting Petri automaton has the structure of a Non-deterministic Finite-state Automaton (NFA). Actually, in that case, the construction we described above is just a variation on Thompson's construction [16], with inlined epsilon transition elimination.

Combined with Theorem 6 from Section 1, the above theorem allows us to reduce the problem of deciding whether $Rel \vDash e = f$ to the problem of checking whether $\mathscr{L}(\mathscr{A}(e_1)) = \mathscr{L}(\mathscr{A}(e_2))$. By symmetry, it then suffices to decide inclusion of Petri automata languages.

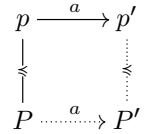
4. Comparing automata

We want to compare automata by testing if any graph accepted by $\mathscr{A}_1$ is also accepted by $\mathscr{A}_2$. Let us go back to standard non-deterministic finite-state automata (NFA), to find intuitions. In that setting an automaton over some alphabet Σ is defined by $\mathscr{A} = \langle Q, \iota, F, \Delta \rangle$ where Q is a finite set of states, ι is an initial state, F is a set of finite states F and Δ is a set of transitions of the form (p, a, q) where p and q are states and a is a letter. Consider two automata $\mathscr{A}_1 = \langle Q_1, \iota_1, F_1, \Delta_1 \rangle$ and $\mathscr{A}_2 = \langle Q_2, \iota_2, F_2, \Delta_2 \rangle$. $L(\mathscr{A}_1) \subseteq L(\mathscr{A}_2)$ means that for any word $w = a_1 \cdots a_n$ accepted by $\mathscr{A}_1$, w is also accepted by $\mathscr{A}_2$. Thus if there is an execution in $\mathscr{A}_1$ recognising w then there is an execution in $\mathscr{A}_2$ recognising w. One can then put them together side by side like so:

$$\begin{array}{lllllll} \iota_1 & \xrightarrow{a_1}_{\mathscr{A}_1} & p_1 & \xrightarrow{a_2}_{\mathscr{A}_1} & \cdots & \xrightarrow{a_n}_{\mathscr{A}_1} & p_n \in F_1 \\ \iota_2 & \xrightarrow{a_1}_{\mathscr{A}_2} & q_1 & \xrightarrow{a_2}_{\mathscr{A}_2} & \cdots & \xrightarrow{a_n}_{\mathscr{A}_2} & q_n \in F_2 \end{array}$$

Thus trying to prove that $L(\mathscr{A}_1) \subseteq L(\mathscr{A}_2)$ amounts to finding a method to build from any run in $\mathscr{A}_1$ a corresponding run in $\mathscr{A}_2$. This can be done by computing a *simulation* between the automaton $\mathscr{A}_1$ and the determinised automaton of $\mathscr{A}_2$. A simulation between these automata is then a relation $\preccurlyeq \subseteq Q_1 \times \mathcal{P}(Q_2)$ such that

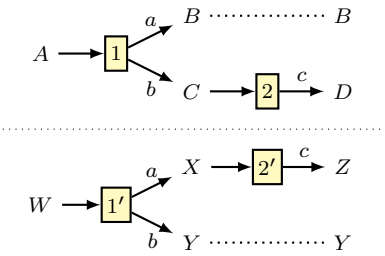
- $\iota_1 \preccurlyeq \{\iota_2\}$, and if $p \preccurlyeq P$ and $p \in F_1$, then $P \cap F_2 \neq \varnothing$;
- if $p \preccurlyeq P$ and $(p, a, p') \in \Delta_1$ then $p' \preccurlyeq P'$, where $P' \coloneqq \{p' \mid (p, a, p') \in \Delta_2,\ p \in P\}$.

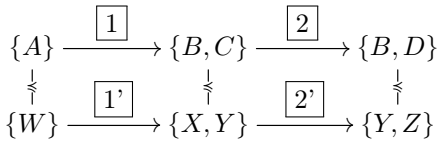


If such a relation can be found, then for any accepting execution in $\mathscr{A}_1$, we can use the relation to extract a corresponding execution in $\mathscr{A}_2$. It is also possible to prove that if the language of $\mathscr{A}_1$ is indeed included in the language of $\mathscr{A}_2$, then such a relation exists. This gives us an algorithm to decide the inclusion of languages, because the set of states of both automata being finite, there is only a finite number of candidates for $\preccurlyeq$. More realistically, one can define a coinductive algorithm that computes a simulation relation on-the-fly.

We follow a similar approach for Petri automata, but we need to make several important adjustments. Consider two automata $\mathscr{A}_1 = \langle P_1, \mathscr{T}_1, \iota_1, \mathscr{F}_1 \rangle$ and $\mathscr{A}_2 = \langle P_2, \mathscr{T}_2, \iota_2, \mathscr{F}_2 \rangle$, we try to show that for any graph G accepted by $\mathscr{A}_1$, G is recognised by $\mathscr{A}_2$. By Lemma 14, this amounts to proving that for any accepting run ξ in $\mathscr{A}_1$, $\mathrm{Tr}(\xi)$ is recognised by some accepting run ξ' in $\mathscr{A}_2$. Leaving non-determinism apart, the first idea that comes to mind is to find a relation between the configurations in $\mathscr{A}_1$ and the configurations in $\mathscr{A}_2$, that satisfy some conditions on the initial and final configurations, and such that if $\xi_k \preccurlyeq \xi'_k$ and $\xi_k \xrightarrow{\tau}_{\mathscr{A}_1} \xi_{k+1}$, then there is a configuration ξ'_{k+1} in $\mathscr{A}_2$ such that $\xi_{k+1} \preccurlyeq \xi'_{k+1}$,

$\xi'_k \xrightarrow{\tau'}_{\mathscr{A}_2} \xi'_{k+1}$, and these transition steps are compatible in some sense. However, such a definition will not give us the result we are looking for. Consider the two runs on the left-hand side:

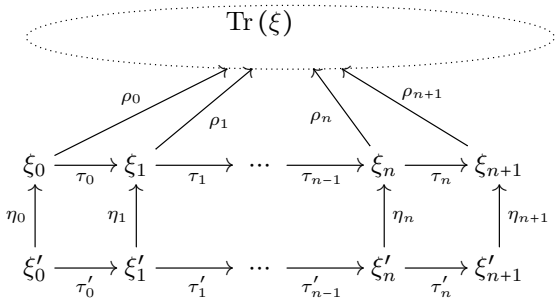




The trace of the first run corresponds to the ground term $a \wedge (b \cdot c)$, and the trace of the second one corresponds to $(a \cdot c) \wedge b$. These two terms are incomparable, but the relation $\preccurlyeq$ depicted on the right-hand side satisfies the previously stated conditions.

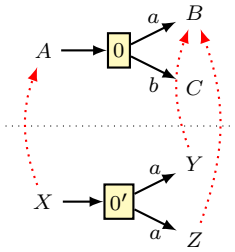
The problem here is that in Petri automata, runs are token firing games. To adequately compare two runs, we need to closely track the tokens. For this reason, we will relate a configuration ξ_k in $\mathscr{A}_1$ not only to a configuration ξ'_k in $\mathscr{A}_2$, but to a map η_k from ξ'_k to ξ_k. This will enable us to associate with each token situated on some place in P_2 another token placed on $\mathscr{A}_1$.

We want to find a reading of $\mathrm{Tr}\,(\xi)$ in $\mathscr{A}_2$, i.e. a run in $\mathscr{A}_2$ together with a sequence of maps associating places in $\mathscr{A}_2$ to positions in $\mathrm{Tr}\,(\xi)$. Consider the picture below. Since we already have a reading of $\mathrm{Tr}\,(\xi)$ along ξ (by defining $\rho_k(p) = \nu(k, p)$, as in the proof of Lemma 14), it suffices to find maps from the places in $\mathscr{A}_2$ to the places in $\mathscr{A}_1$ (the maps η_k): the reading of $\mathrm{Tr}\,(\xi)$ in $\mathscr{A}_2$ will be obtained by composing η_k with ρ_k.

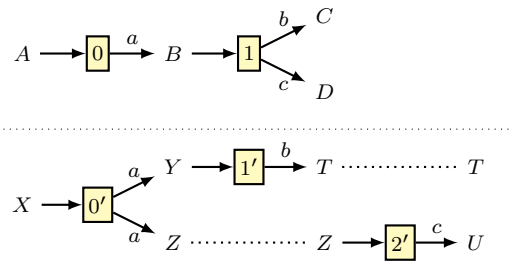


We need to impose some constraints on the maps (η_k) to ensure that $(\rho_k \circ \eta_k)_{0 \leqslant k \leqslant n}$ is indeed a correct reading in $\mathscr{A}_2$. First, we need to ascertain that a transition τ'_k in $\mathscr{A}_2$ may be fired from the reading state $\rho_k \circ \eta_k$ to reach the reading state $\rho_{k+1} \circ \eta_{k+1}$. Furthermore, as for NFA, we want transitions τ_k and τ'_k to be related: specifically, we require τ'_k to be included (via the morphisms η_k and η_{k+1}) in the transition τ_k. This is meaningful because transition τ_k contains a lot of information about the vertex k of $\mathrm{Tr}\,(\xi)$ and about ρ: the labels of the outgoing edges from k are the labels on the output of τ_k, and the only places that will ever be mapped to k in the reading ρ are exactly the places in the input of τ_k.

This already shows an important difference between the simulations for NFA and Petri automata. For NFA, we relate a transition $p \xrightarrow{a} p'$ to a transition $q \xrightarrow{a} q'$ with the same label a. Here the transitions $\xi_k \xrightarrow{\tau_k}_{\mathscr{A}_1} \xi_{k+1}$ may have different labels. Consider the step represented on the right, corresponding to a square in the above diagram. The output of $\boxed{0}$ has a label b that does not appear in $\boxed{0'}$, and $\boxed{0'}$ has two outputs labelled by a. Nevertheless this satisfies the conditions informally stated above, indeed, $a \wedge b \leqslant a \wedge a$ holds.



However this definition is not yet satisfactory. Consider the two runs below:



Their traces correspond respectively to the ground terms $a \cdot (b \wedge c)$ and $(a \cdot b) \wedge (a \cdot c)$. The problem is that $a \cdot (b \wedge c) \leqslant (a \cdot b) \wedge (a \cdot c)$, but with the previous definition, we cannot relate these runs: they do not have the same length. The solution here consists in grouping the transitions $\boxed{1'}$ and $\boxed{2'}$ together, and consider these two steps as a single step in a *parallel run*. This last modification gives us a notion of simulation we can really work with.

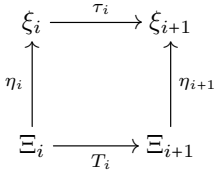
Definition 19 (Simulation)
A relation $\preccurlyeq \subseteq \mathcal{P}(P_1) \times \mathcal{P}(P_2 \rightharpoonup P_1)$ between the configurations of $\mathscr{A}_1$ and the partial maps from the places of $\mathscr{A}_2$ to the places of $\mathscr{A}_1$ is called a *simulation* between $\mathscr{A}_1$ and $\mathscr{A}_2$ if:

- $\{\iota_1\} \preccurlyeq \{[\iota_2 \mapsto \iota_1]\}$;
- if $\xi \preccurlyeq E$ and $\xi \xrightarrow{(s,t)}_{\mathscr{A}_1} \xi'$, then $\xi' \preccurlyeq E'$ where E' is the set of all η' such that there is some $\eta \in E$ and a compatible set of transitions $T \subseteq \mathscr{T}_2$ such that:
  - $\mathrm{dom}(\eta) \xrightarrow{T}_{\mathscr{A}_2} \mathrm{dom}(\eta')$;
  - $\forall (s', t') \in T, \eta(s') \subseteq s$ and $\forall (x, q) \in t', (x, \eta'(q)) \in t$;
  - $\forall p \in \mathrm{dom}(\eta), (\forall (s', t') \in T, p \notin s') \Rightarrow \eta(p) = \eta'(p)$.
- if $\xi \preccurlyeq E$ and $\xi \in \mathscr{F}_1$, then there must be some $\eta \in E$ such that $\mathrm{dom}(\eta) \in \mathscr{F}_2$. *

We will now prove that the language of $\mathscr{A}_1$ is contained in the language of $\mathscr{A}_2$ if and only if there exists such a simulation. We first introduce the following notion of embedding.

Definition 20 (Embedding)
Let $\xi = \langle (\xi_k)_{0 \leqslant k \leqslant n}, (\tau_k)_{0 \leqslant i < n} \rangle$ be a run in $\mathscr{A}_1$, and $\Xi = \langle (\Xi_k)_{0 \leqslant k \leqslant n}, (T_i)_{0 \leqslant i < n} \rangle$ a parallel run in $\mathscr{A}_2$. An *embedding* of Ξ into ξ is a sequence $(\eta_i)_{0 \leqslant i \leqslant n}$ of maps such that for any $i < n$, we have:

- η_i is a map from Ξ_i to ξ_i;
- the image of T_i by η_i is included in τ_i, meaning that for any $(s, t) \in T_i$, for any $p \in s$ and $(x, q) \in t$, $\eta_i(p)$ is contained in the input of τ_i and $(x, \eta_{i+1}(q))$ is in the output of τ_i;
- the image of the tokens in Ξ_i that do not appear in the input of T_i are preserved ($\eta_i(p) = \eta_{i+1}(p)$) and their image is not in the input of τ_i.



*

Figure 8 illustrates the embedding of some parallel run, with trace $((b \cdot c \cdot a \cdot b) \wedge (b \cdot b \cdot c \cdot a)) \cdot d$, into the run presented in Figure 6. Notice that is it necessary to have a parallel run instead of a simple one: to find something that matches transition $\boxed{1}$, we need to fire two transitions in parallel.

There is a close relationship between simulations and embeddings:

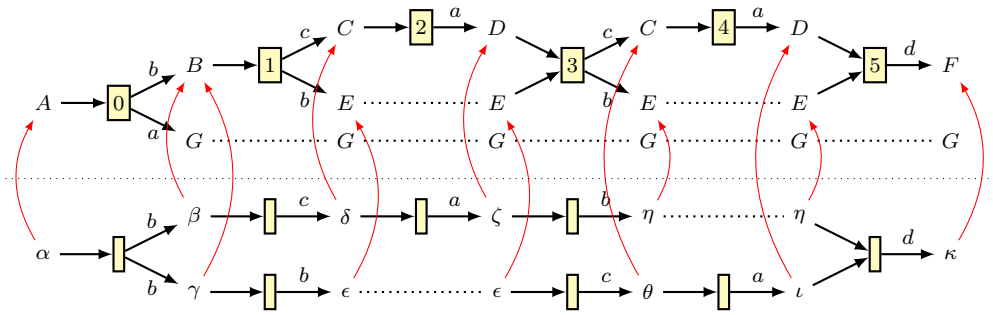


Figure 8: Embedding of a parallel run into the run from Figure 6.

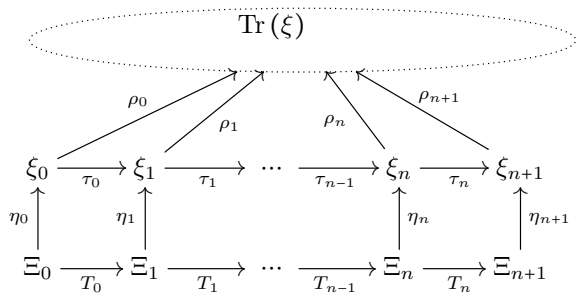
Lemma 21. *Let $\mathscr{A}_1$ and $\mathscr{A}_2$ be two Petri automata, the following are equivalent:*

1. *there exists a simulation $\preccurlyeq$ between $\mathscr{A}_1$ and $\mathscr{A}_2$;*
2. *for any accepting run ξ in $\mathscr{A}_1$, there is an accepting parallel run Ξ in $\mathscr{A}_2$ that can be embedded into ξ.*

Proof. If we have a simulation $\preccurlyeq$, let $\xi = \langle (\xi_k)_{0\leqslant k\leqslant n}, (\tau_k)_{0\leqslant k<n}\rangle$ be an accepting run in $\mathscr{A}_1$. By the definition of simulation, we can find a sequence of sets of maps $(E_k)_{0\leqslant k\leqslant n}$ such that $E_0 = \{[\iota_2 \mapsto \iota_1]\}$ and $\forall k, \xi_k \preccurlyeq E_k$. Furthermore, we can extract from this a sequence of maps $(\eta_k)_{0\leqslant k\leqslant n}$ and a sequence of parallel transitions $(T_k)_{0\leqslant k<n}$ such that (η_k) is an embedding of $\langle (\mathrm{dom}\,(\eta_k))_{0\leqslant k\leqslant n}, (T_k)_{0\leqslant k<n}\rangle$ (which is accepting) into ξ. This follows directly from the definitions of embedding and simulation.

On the other hand, if we have property *2.*, then we can define a relation $\preccurlyeq$ by saying that $\xi \preccurlyeq E$ if there is an accepting run $\xi' = \langle (\xi'_k)_{0\leqslant k\leqslant n}, (\tau_k)_{0\leqslant k<n}\rangle$ in $\mathscr{A}_1$ such that there is an index k_0: $\xi = \xi'_{k_0}$; and the following holds: $\eta \in E$ if there is an accepting parallel run $\Xi = \langle (\Xi_k)_{0\leqslant k\leqslant n}, (T_k)_{0\leqslant k<n}\rangle$ and $(\eta'_k)_{0\leqslant k\leqslant n}$ an embedding of Ξ into ξ such that $\eta = \eta'_{k_0}$. It is then immediate to check that $\preccurlyeq$ is indeed a simulation. □

If η is an embedding of Ξ into ξ, we can easily check that $(\rho_i \circ \eta_i)_{0\leqslant i\leqslant n}$ is a parallel reading of $\mathrm{Tr}\,(\xi)$ along Ξ in $\mathscr{A}_2$, as illustrated by this diagram:



Thus, it is clear that once we have such a run Ξ with the sequence of maps η, we have that $\mathrm{Tr}\,(\xi)$ is indeed in the language of $\mathscr{A}_2$. The more difficult question is the completeness of this approach: if $\mathrm{Tr}\,(\xi)$ is recognised by $\mathscr{A}_2$, is it always the case that we can find a run Ξ that may be embedded into ξ? The answer is affirmative, thanks to Lemma 23 below. If $(\rho_j)_{0\leqslant j\leqslant n}$ is a reading of G along $\xi = \langle (\xi_k)_{0\leqslant k\leqslant n}, (s_k, t_k)_{0\leqslant k<n}\rangle$, we write $\mathit{active}(j)$ for the only position in $\rho_j(s_j)$[1].

[1]Recall that if $(\rho_j)_{0\leqslant j\leqslant n}$ is a reading along ξ then $\forall p, q \in s_j, \rho_j(p) = \rho_j(q)$.

Definition 22 (Consistent ordering)
$\leqslant$ is a *consistent ordering* on $G = \langle V, E, \iota, o\rangle$ if $\langle V, \leqslant\rangle$ is a linear order and $(p, x, q) \in E$ entails $p \leqslant q$. ∗

Lemma 23. *Let $G \in \mathscr{L}(\mathscr{A}_2)$ and $\leqslant$ be any consistent ordering on G. Then there exists a run ξ and a reading $(\rho_j)_{0\leqslant j\leqslant n}$ of G along ξ such that $\forall k, active(k) \leqslant active(k+1)$.*

Proof. The proof of this result is achieved by taking any run ξ accepting G, and then exchanging transitions in ξ according to $\leqslant$, while preserving the existence of a reading. The details of this proof being a bit technical, we omit them here. □

Notice that if G contains cycles, this lemma cannot apply because of the lack of consistent ordering. This enables us to build an embedding from any reading of $\mathrm{Tr}(\xi)$ in $\mathscr{A}_2$.

Lemma 24. *Let ξ a accepting run of $\mathscr{A}_1$. Then $Tr(\xi)$ is in $\mathscr{L}(\mathscr{A}_2)$ if and only if there is an accepting parallel run in $\mathscr{A}_2$ that can be embedded into ξ.*

Proof. We do not include the details of this proof here for length reasons.

For the if direction, we build a parallel reading from the embedding, as explained above. For the other direction, we consider a reading of $\mathrm{Tr}(\xi)$ in $\mathscr{A}_2$ along some run ξ'. Notice that the natural ordering on $\mathbb{N}$ is consistent for $\mathrm{Tr}(\xi)$; we may thus change the order of the transitions in ξ' (using Lemma 23) and group them adequately to obtain a parallel reading Ξ that embeds in ξ. □

So we know that the existence of embeddings is equivalent to the inclusion of languages, and we previously established that it is also equivalent to the existence of a simulation relation. Hence, the following characterisation holds:

Theorem 25. *Let $\mathscr{A}_1$ and $\mathscr{A}_2$ be two Petri automata. $\mathscr{L}(\mathscr{A}_1) \subseteq \mathscr{L}(\mathscr{A}_2)$ if and only if there exists a simulation relation $\preccurlyeq$ between $\mathscr{A}_1$ and $\mathscr{A}_2$.*

Proof. By Lemmas 14, 21 and 24. □

As Petri automata are finite, there are finitely many relations in $\mathcal{P}(\mathcal{P}(P_1) \times \mathcal{P}(P_2 \dashrightarrow P_1))$. The existence of a simulation thus is decidable, allowing us to prove the main result:

Theorem 26. *Given two expressions $e, f \in Reg_X^{\wedge -}$, testing whether $Rel \vDash e = f$ is decidable.*

Proof. By Theorems 6, 17 and 25, and reasoning by double inclusion. □

In practice, we may build the simulation on-the-fly, starting from the pair $(\{\iota_1\}, \{[\iota_2 \mapsto \iota_1]\})$ and progressing from there. We have implemented this algorithm in OCaml: even though its theoretical worst case time complexity is huge[2], we get a result almost instantaneously on simple one-line examples.

Conclusions and directions for future work

By introducing Petri automata, we proved the decidability of the equivalence of identity-free regular expressions with intersection, with respect to their relational interpretations. Actually, this also holds for their language interpretations, because Andréka et al. showed in [1] that the classes of identity-free relational Kleene lattices and identity-free language Kleene lattices coincide. They differ however when we include the identity constant, or the converse operation.

[2] A quick analysis gives a $\mathcal{O}\left(2^{n+n^m}\right)$ complexity bound, where n and m are the numbers of places of the automata.

The construction and algorithm presented here were implemented in OCaml as an exercise. The resulting program is available online as an interactive applet [5].

By adding ϵ-transitions to Petri automata, we could partly cope with the identity, in the sense that we can build automata to recognise the graph languages of expressions over the signature $\langle \vee, \wedge, \cdot, ^\star, \mathbb{1}, \mathbb{0} \rangle$. However, we did not find a way of comparing ϵ-Petri automata: the notion of simulation we described here is not equivalent to the inclusion of languages for these automata.

Similarly, we could define a variant of Petri automata for recognising the graph languages associated to expressions with converse: it suffice to consider a duplicated alphabet. However, we did not find a proper way of extending our notion of simulation to capture language inclusion of such automata.

References

[1] H. Andréka, S. Mikulás, and I. Németi. The equational theory of Kleene lattices. *Theoretical Computer Science*, 412(52):7099–7108, 2011.

[2] H. Andréka and D. Bredikhin. The equational theory of union-free algebras of relations. *Algebra Universalis*, 33(4):516–532, 1995.

[3] S. L. Bloom, Z. Ésik, and G. Stefanescu. Notes on equational theories of relations. *Algebra Universalis*, 33(1):98–126, 1995.

[4] P. Brunet and D. Pous. Kleene algebra with converse. In *Proc. RAMiCS*, volume 8428 of *Lecture Notes in Computer Science*, pages 101–118. Springer Verlag, 2014.

[5] P. Brunet and D. Pous. Web appendix to this abstract, 2014. `http://perso.ens-lyon.fr/paul.brunet/rklm.html`.

[6] J. H. Conway. *Regular algebra and finite machines*. Chapman and Hall Mathematics Series, 1971.

[7] Z. Ésik and L. Bernátsky. Equational properties of Kleene algebras of relations with conversion. *Theoretical Computer Science*, 137(2):237–251, 1995.

[8] P. J. Freyd and A. Scedrov. *Categories, Allegories*. North Holland, 1990.

[9] S. C. Kleene. *Representation of Events in Nerve Nets and Finite Automata*. Memorandum. Rand Corporation, 1951.

[10] D. Kozen. On Kleene Algebras and closed semirings. In *Proc. MFCS*, volume 452 of *Lecture Notes in Computer Science*, pages 26–47. Springer Verlag, 1990.

[11] D. Kozen. A completeness theorem for Kleene Algebras and the algebra of regular events. In *Proc. LICS*, pages 214–225. IEEE Computer Society, 1991.

[12] D. Krob. A Complete System of B-Rational Identities. In *Proc. ICALP*, volume 443 of *Lecture Notes in Computer Science*, pages 60–73. Springer Verlag, 1990.

[13] T. Murata. Petri nets: Properties, analysis and applications. *Proc. of the IEEE*, 77(4):541–580, Apr 1989.

[14] C. A. Petri. Fundamentals of a theory of asynchronous information flow. In *IFIP Congress*, pages 386–390, 1962.

[15] C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, Darmstadt Univ. of Tech., 1962.

[16] K. Thompson. Regular expression search algorithm. *C. of the ACM*, 11:419–422, 1968.