Kleene Algebra with Converse

Paul Brunet, Damien Pous

Kleene Algebra with Converse

Paul Brunet and Damien Pous *

LIP, CNRS, ENS Lyon, INRIA, Université de Lyon, UMR 5668

Abstract The equational theory generated by all algebras of binary relations with operations of union, composition, converse and reflexive transitive closure was studied by Bernátsky, Bloom, Ésik, and Stefanescu in 1995. We reformulate some of their proofs in syntactic and elementary terms, and we provide a new algorithm to decide the corresponding theory. This algorithm is both simpler and more efficient; it relies on an alternative automata construction, that allows us to prove that the considered equational theory lies in the complexity class PSpace.
Specific regular languages appear at various places in the proofs. Those proofs were made tractable by considering appropriate automata recognising those languages, and exploiting symmetries in those automata.

Introduction

In many contexts in computer science and mathematics operations of union, sequence or product and iteration appear naturally. *Kleene Algebra*, introduced by John H. Conway under the name *regular algebra* [Con71], provides an algebraic framework allowing to express properties of these operators, by studying the equivalence of expressions built with these connectives. It is well known that the corresponding equational theory is decidable [Kle51], and that it is complete for language and relation models.

As expressive as it may be, one may wish to integrate other usual operations in such a setting. Theories obtained this way, by addition of a finite set of equations to the axioms of Kleene Algebra, are called *Extensions of Kleene Algebra*. We shall focus here on one of these extensions, where an operation of *converse* is added to Kleene Algebra. The converse of a word is its mirror image (the word obtained by reversing the order of the letters), and the converse $R^\vee$ of a relation R is its reciprocal ($xR^\vee y \triangleq yRx$). This natural operation can be expressed simply as a set of equations that we add to Kleene Algebra's axioms.

The question that arises once this theory is built is its decidability: given two formal expressions built with the connectives product, sum, iteration and converse, can one decide automatically if they are equivalent, meaning that their equality can be proven using the axioms of the theory? Bloom, Ésik,

* Work partially funded by the french projects PiCoq (ANR-09-BLAN-0169-01) and PACE (ANR-12IS02001).

Stefanescu and Bernátsky gave an affirmative answer to that question in two articles, [BÉS95] and [ÉB95], in 1995.

However, although the algorithm they define proves the decidability result, it is too complicated to be used in actual applications. In this paper, beside some simplifications of the proofs given in [BÉS95], we give a new and more efficient algorithm to decide this problem, which we place in the complexity class PSpace.

The equational theory of Kleene algebra cannot be finitely axiomatised [Red64]. Krob presented the first purely axiomatic (but infinite) presentation [Kro90]. Several finite quasi-equational characterisations have been proposed [Sal66,Bof90, Kro90,Koz91,Bof95]; here we follow the one from Kozen [Koz91].

A Kleene Algebra is an algebraic structure $\langle K, +, \cdot, ^\star, \mathbb{0}, \mathbb{1}\rangle$ such that $\langle K, +, \cdot, \mathbb{0}, \mathbb{1}\rangle$ is an idempotent semi-ring, and the operation * satisfies the following properties

$$\mathbb{1} + aa^\star \leqslant a^\star \tag{1a}$$
$$\mathbb{1} + a^\star a \leqslant a^\star \tag{1b}$$
$$b + ax \leqslant x \Rightarrow a^\star b \leqslant x \tag{1c}$$
$$b + xa \leqslant x \Rightarrow ba^\star \leqslant x \tag{1d}$$

(Here $a \leqslant b$ is a shorthand for $a + b = b$.)

The quasi-variety KA consists in the axioms of an idempotent semi-ring together with axioms and inference rules (1a) to (1d). Kleene Algebras are thus *models* of KA. We shall call *regular expressions over* X, written Reg_X, the expressions built from letters of X, the binary connectives $+$ and $\cdot$, the unary connective * and the two constants $\mathbb{0}$ and $\mathbb{1}$.

Two families of such algebras are of particular interest: languages (sets of finite words over a finite alphabet, with union as sum and concatenation as product) and relations (binary relations over an arbitrary set with union and composition). KA is complete for both these models [Kro90, Koz91], meaning that for any $e, f \in \mathrm{Reg}_X$, $\mathrm{KA} \vdash e = f$ if and only if e and f coincide under any language (resp. relational) interpretation. This last property will be written $e \equiv_{\mathrm{Lang}} f$ (resp. $e \equiv_{\mathrm{Rel}} f$).

More remarkably, if we denote by $[\![e]\!]$ the language denoted by an expression e, we have that for any $e, f \in \mathrm{Reg}_X$, $\mathrm{KA} \vdash e = f$ if and only if $[\![e]\!] = [\![f]\!]$. By Kleene's theorem (see [Kle51]) the equality of two regular languages can be reduced to the equivalence of two finite automata, which is easy to compute. Hence, the theory KA is decidable.

Now let us add a unary operation of converse to regular expressions. We shall denote by $\mathrm{Reg}^\vee_X$ the set of regular expressions with converse over a finite alphabet X. While doing so, several questions arise:

1. Can the converse on languages and on relations be encoded in the same theory?
2. What axioms do we need to add to KA to model these operations?

3. Are the resulting theories complete for languages and relations?
4. Are these theories decidable?

There is a simple answer to the first question: no. Indeed the equation $a \leqslant a \cdot a^\vee \cdot a$ is valid for any relation a (because if $(x,y) \in a$, then $(x,y) \in a$, $(y,x) \in a^\vee$, and $(x,y) \in a$, so that $(x,y) \in a \circ a^\vee \circ a$). But this equation is not satisfied for all languages a (for instance, with the language $a = \{x\}$, $a \cdot a^\vee \cdot a = \{xxx\}$ and $x \notin \{xxx\}$). This means that there are two distinct theories corresponding to these two families of models. Let us begin by considering the case of languages.

Theorem 1 (Completeness of KAC$^-$ [BÉS95])**.** *A complete axiomatisation of the variety Lang$^\vee$ of languages generated by concatenation, union, star, and converse consists of the axioms of KA together with axioms (2a) to (2d).*

$$(a+b)^\vee = a^\vee + b^\vee \tag{2a}$$
$$(a \cdot b)^\vee = b^\vee \cdot a^\vee \tag{2b}$$
$$(a^\star)^\vee = (a^\vee)^\star \tag{2c}$$
$$a^{\vee\vee} = a. \tag{2d}$$

We call this theory KAC$^-$; it is decidable.

As for relations, we write $e \equiv_{\mathrm{Lang}^\vee} f$ if e and f have the same language interpretations (for a formal definition, see the "Notation" subsection below). To prove this result, one first associates to any expression $e \in \mathrm{Reg}_X^\vee$ an expression $\mathbf{e} \in \mathrm{Reg}_{\mathbf{X}}$, where $\mathbf{X}$ is an alphabet obtained by adding to X a disjoint copy of itself. Then, one proves that the following implications hold.

$$e \equiv_{\mathrm{Lang}^\vee} f \quad \Rightarrow \quad [\![\mathbf{e}]\!] = [\![\mathbf{f}]\!] \tag{3}$$
$$[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!] \quad \Rightarrow \quad \mathrm{KAC}^- \vdash e = f \tag{4}$$

(That KAC$^- \vdash e = f$ entails $e \equiv_{\mathrm{Lang}^\vee} f$ is obvious; decidability comes from that of regular languages equivalence.) We reformulate Bloom et al.'s proofs of these implications in elementary terms in Section 1.1.

As stated before, the equation $a \leqslant a \cdot a^\vee \cdot a$ provides a difference between languages with converse and relations with converse. It turns out that it is the only difference, in the sense that the following theorem holds:

Theorem 2 (Completeness of KAC [BÉS95, ÉB95])**.** *A complete axiomatisation of the variety Rel$^\vee$ of relations generated by composition, union, star, and converse consists of the axioms of KAC$^-$ together with the axiom (5).*

$$a \leqslant a \cdot a^\vee \cdot a. \tag{5}$$

We call this theory KAC; it is decidable.

The proof of this result also relies on a translation into regular languages. Ésik et al. define a notion of *closure*, written $\mathcal{cl}()$, for languages over $\mathbf{X}$, and they prove the following implications:

$$e \equiv_{\mathrm{Rel}^\vee} f \quad \Rightarrow \quad \mathcal{cl}(\llbracket \mathbf{e} \rrbracket) = \mathcal{cl}(\llbracket \mathbf{f} \rrbracket) \tag{6}$$

$$\mathcal{cl}(\llbracket \mathbf{e} \rrbracket) = \mathcal{cl}(\llbracket \mathbf{f} \rrbracket) \quad \Rightarrow \quad \mathrm{KAC} \vdash e = f \tag{7}$$

(Again, that $\mathrm{KAC} \vdash e = f$ entails $e \equiv_{\mathrm{Rel}^\vee} f$ is obvious.) The first implication (6) was proven in [BÉS95]; we give a new formulation of this proof in Section 1.2. The second one (7) was proven in [ÉB95].

The last consideration is the decidability of KAC. To this end, Bloom et al. propose a construction to obtain an automaton recognising $\mathcal{cl}(L)$, when given an automaton recognising L. Decidability follows: to decide whether $\mathrm{KAC} \vdash e = f$ one can build two automata recognising $\mathcal{cl}(\llbracket \mathbf{e} \rrbracket)$ and $\mathcal{cl}(\llbracket \mathbf{f} \rrbracket)$ and check if they are equivalent. Unfortunately, their construction tends to produce huge automata, which makes it useless for practical application. We propose a new and simpler one in Section 2; by analysing this construction, we show in Section 3 how it leads to a proof that the problem of equivalence in KAC is PSpace.

Notation

For any word w, $|w|$ is the size of w, meaning its number of letters; for any $1 \leqslant i \leqslant |w|$, we'll write $w(i)$ for the i^{th} letter of w and $w|_i \triangleq w(1)w(2)\cdots w(i)$ for its prefix of size i. Also, $\mathrm{suffixes}(w) \triangleq \{v \mid \exists u : uv = w\}$ is the set of all suffixes of w. A deterministic automaton is a tuple $\langle Q, \Sigma, q_0, T, \delta\rangle$; with Q a set of states, Σ an alphabet, $q_0 \in Q$ an initial state, $T \subseteq Q$ a set of final states and $\delta : Q \times \Sigma \to Q$ a transition function. A non-deterministic automaton is a tuple $\langle Q, \Sigma, I, T, \Delta\rangle$; with Q, Σ and T same as before, $I \subseteq Q$ a set of initial states and $\Delta \subseteq Q \times \Sigma \times Q$ a set of transitions. We write $L(\mathscr{A})$ for the *language recognised by the automaton* $\mathscr{A}$. For any $a \in \Sigma$, we write $\Delta(a)$ for $\{(p,q) \mid (p,a,q) \in \Delta\}$. We also use the compact notation $p \xrightarrow{w}_{\mathscr{A}} q$ to denote that there is in the automaton $\mathscr{A}$ a path labelled by w from the state p to the state q. For a set $E \subseteq Q$ and a relation R over Q, we write $E \cdot R$ for the set $\{y \mid \exists x \in E : xRy\}$.

Given a map σ from a set X to the languages on an alphabet Σ (resp. the relations on a set S), there is a unique extension of σ into a homomorphism from Reg_X to Lang_Σ (resp. Rel_S), which we denote by $\widehat{\sigma}$. The same thing can be done with regular expressions with converse, and we will use the same notation for it. We finally denote by $\equiv_V$ the equality in a variety V (Lang, Rel, $\mathrm{Lang}^\vee$ or $\mathrm{Rel}^\vee$): $e \equiv_V f \triangleq \forall K, \forall \sigma : X \to V_K, \widehat{\sigma}(e) = \widehat{\sigma}(f)$.

1 Preliminary material

1.1 Languages with converse: theory $\mathbf{KAC}^-$

We consider regular expressions with converse over a finite alphabet X. The alphabet $\mathbf{X}$ is defined as $X \cup X'$, where $X' \triangleq \{x' \mid x \in X\}$ is a disjoint copy

of X. As a shorthand, we use $'$ as an internal operation on $\mathbf{X}$ going from X to X' and from X' to X such that if $x \in X$, $x' \triangleq x' \in X'$ and $(x')' \triangleq x \in X$. An important operation in the following is the translation of an expression $e \in \mathrm{Reg}_X^\vee$ to an expression $\mathbf{e} \in \mathrm{Reg}_{\mathbf{X}}$. We proceed to its definition in two steps.

Let $\tau(e)$ denote the normal form of an expression $e \in \mathrm{Reg}_X^\vee$ in the following convergent term rewriting system:

$$\begin{array}{lll} (a+b)^\vee \to a^\vee + b^\vee & \mathbb{0}^\vee \to \mathbb{0} & (a^\star)^\vee \to (a^\vee)^\star \\ (a \cdot b)^\vee \to b^\vee \cdot a^\vee & \mathbb{1}^\vee \to \mathbb{1} & a^{\vee\vee} \to a \end{array}$$

The corresponding equations being derivable in KAC^-, one easily obtain that

$$\forall e \in \mathrm{Reg}_X^\vee,\ \mathrm{KAC}^- \vdash \tau(e) = e \tag{8}$$

We finally denote by $\mathbf{e}$ the expression obtained by further applying the substitution $\nu \triangleq [x^\vee \mapsto x', (\forall x \in \mathbf{X})]$, i.e., $\mathbf{e} \triangleq \nu(\tau(e))$. (Note that $\mathbf{e} \in \mathrm{Reg}_{\mathbf{X}}$: it is regular, all occurrences of the converse operation have been eliminated.) As explained in the introduction, Bloom et al.'s proof [BÉS95] amounts to proving the implications (3) and (4). We include a syntactic and elementary presentation of this proof, for the sake of completeness.

Lemma 3. *For all $e, f \in Reg_X^\vee$, $e \equiv_{Lang^\vee} f$ entails $[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!]$.*

Proof. For any $e \in \mathrm{Reg}_X^\vee$, we have $\tau(e) \equiv_{\mathrm{Lang}^\vee} e$ (†) as an immediate consequence of (8). Let us write $X_\bullet \triangleq X \uplus \{\bullet\}$ and consider the following interpretations (which appear in [BÉS95, proof of Proposition 4.3]):

$$\begin{array}{rl} \mu : X \longrightarrow & \mathcal{P}\left(X_\bullet^\star\right) \\ x \longmapsto & \{x \cdot \bullet\} \end{array} \qquad \begin{array}{rl} \eta : \mathbf{X} \longrightarrow & \mathcal{P}\left(X_\bullet^\star\right) \\ x \in X \longmapsto & \{x \cdot \bullet\} \\ x' \in X' \longmapsto & \{\bullet \cdot x\} \end{array}$$

One can check (see Appendix A.1)that $\widehat{\eta}$ is injective modulo equality of denoted languages, in the sense that for any expression $e \in \mathrm{Reg}_{\mathbf{X}}$, we have

$$\widehat{\eta}(e) = \widehat{\eta}(f) \text{ implies that } [\![e]\!] = [\![f]\!]\ . \tag{9}$$

By a simple induction on e, we get $\widehat{\mu}(\tau(e)) = \widehat{\eta}(\nu(\tau(e))) = \widehat{\eta}(\mathbf{e})$. Combined with (†), we deduce that $\widehat{\mu}(e) = \widehat{\eta}(\mathbf{e})$. All in all, we obtain: $e \equiv_{\mathrm{Lang}^\vee} f \Rightarrow \widehat{\mu}(e) = \widehat{\mu}(f) \Rightarrow \widehat{\eta}(\mathbf{e}) = \widehat{\eta}(\mathbf{f}) \Rightarrow [\![\mathbf{e}]\!] = [\![\mathbf{f}]\!]$. □

The second implication is even more immediate, using KA completeness.

Lemma 4. *For all $e, f \in Reg_X^\vee$, if $[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!]$ then $KAC^- \vdash e = f$.*

Proof. By completeness of KA [Kro90, Koz91], if $[\![\mathbf{e}]\!] = [\![\mathbf{f}]\!]$, then we know that there is a proof $\pi_1 : \mathrm{KA} \vdash \mathbf{e} = \mathbf{f}$. As KA is contained in KAC^-, the same proof can be seen as $\pi_1 : \mathrm{KAC}^- \vdash \mathbf{e} = \mathbf{f}$. By substituting x' by $(x^\vee)$ everywhere in this proof, we get a new proof $\pi_2 : \mathrm{KAC}^- \vdash \tau(e) = \tau(f)$. By (8) and transitivity we thus get $\mathrm{KAC}^- \vdash e = f$. □

We finally deduce that $e \equiv_{\mathrm{Lang}^\vee} f \Leftrightarrow [\![\mathbf{e}]\!] = [\![\mathbf{f}]\!] \Leftrightarrow \mathrm{KAC}^- \vdash e = f$. Since the regular expressions $\mathbf{e}$ and $\mathbf{f}$ can be easily computed from e and f, the problem of equivalence in KAC^- thus reduces to an equality of regular languages, which makes it decidable.

1.2 Relations with converse: theory KAC

We now move to the equational theory generated by relational models. It turns out that this theory will be characterised using "closed" languages on the extended alphabet $\mathbf{X}$. To define this closure operation, we first define a mirror operation $\overline{w}$ on words over $\mathbf{X}$, such that $\overline{\epsilon} \triangleq \epsilon$ and for any $x, w \in \mathbf{X} \times \mathbf{X}^\star$, $\overline{wx} = x'\overline{w}$. Accordingly with the axiom (5) of KAC we define a reduction relation $\rightsquigarrow$ on words over $\mathbf{X}$, using the following word rewriting rule.

$$w\overline{w}w \rightsquigarrow w \quad .$$

We call $w\overline{w}w$ a *pattern* of *root* w. The last two thirds of the pattern are $\overline{w}w$. Following [BÉS95, ÉB95], we extend this relation into a closure operation on languages.

Definition 5. The *closure* of a language $L \subseteq \mathbf{X}^\star$ is the smallest language containing L that is downward-closed with respect to $\rightsquigarrow$:

$$\mathcal{cl}(L) \triangleq \{v \mid \exists u \in L : u \rightsquigarrow^\star v\} \quad .$$

Example 6. If $X = \{a, b, c, d\}$, then $\mathbf{X} = \{a, b, c, d, a', b', c', d'\}$, and $\overline{ab'} = ba'$. We have the reduction $cab'ba'ab'd' \rightsquigarrow cab'd'$, by triggering a pattern of root ab'. For $L = \{aa'a, b, cab'ba'ab'd'\}$, we have $\mathcal{cl}(L) = L \cup \{a, cab'd'\}$.

Now we define a family of languages which play a prominent role in the sequel.

Definition 7. For any word $w \in \mathbf{X}^\star$, we define a regular language $\Gamma(w)$ by:

$$\Gamma(\epsilon) \triangleq \{\epsilon\}$$
$$\forall x \in \mathbf{X}, \forall w \in \mathbf{X}^\star, \quad \Gamma(wx) \triangleq (x'\Gamma(w)x)^\star \quad .$$

An equivalent operator called G is used in [BÉS95]: we actually have $\Gamma(w) = G(\overline{w})$, and our recursive definition directly corresponds to [BÉS95, Proposition 5.11.(2)]. By using such a simple recursive definition, we avoid the need for the notion of *admissible maps*, which is extensively used in [BÉS95].

Instead, we just have the following property to establish, which illustrates why these languages are of interest: words in $\Gamma(w)$ reduce into the last two thirds of a pattern compatible with w. Therefore, in the context of recognition by an automaton, $\Gamma(w)$ contains all the words that could potentially be skipped after reading w, in a closure automaton.

Proposition 8. *For all words u and v, $u \in \Gamma(v) \Leftrightarrow \exists t \in \mathit{suffixes}(v) : u \rightsquigarrow^\star \overline{t}t$.*

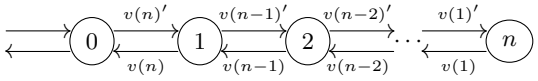


Fig. 1: Automaton $\mathscr{G}(v)$ recognising $\Gamma(v)$, with $|v| = n$.

Proof. The proof of the implication from left to right is routine but a bit lengthy,so that we put it in Appendix A.2.

For the converse implication, we first define the following language: $\Gamma'(v) \triangleq \{\bar{t}t \mid t \in \text{suffixes}(v)\}$. We thus have to show that the upward closure of $\Gamma'(v)$ is contained in $\Gamma(v)$. We first check that this language satisfies $\Gamma'(\epsilon) = \epsilon$ and $\Gamma'(vx) = \epsilon + x'\Gamma'(v)x$, which allows us to deduce that $\Gamma'(v) \subseteq \Gamma(v)$ by a straightforward induction.

It thus suffices to show that $\Gamma(v)$ is upward-closed with respect to $\rightsquigarrow$. For this, we introduce the family of automata $\mathscr{G}(v)$ depicted in Figure 1. One can check that $\mathscr{G}(v)$ recognises $\Gamma(v)$ by a simple induction on v. One can moreover notice that in this automaton, if $p \xrightarrow{x}_{\mathscr{G}(v)} q$, then $q \xrightarrow{x'}_{\mathscr{G}(v)} p$. More generally, for any word u, if $p \xrightarrow{u}_{\mathscr{G}(v)} q$, then $q \xrightarrow{\overline{u}}_{\mathscr{G}(v)} p$. So if $u_1 w u_2 \in \Gamma(v)$, then by definition of the automaton we have $0 \xrightarrow{u_1}_{\mathscr{G}(v)} q_1 \xrightarrow{w}_{\mathscr{G}(v)} q_2 \xrightarrow{u_2}_{\mathscr{G}(v)} 0$, and thus, by the previous remark:

$$0 \xrightarrow[\mathscr{G}(v)]{u_1} q_1 \xrightarrow[\mathscr{G}(v)]{w} q_2 \xrightarrow[\mathscr{G}(v)]{\overline{w}} q_1 \xrightarrow[\mathscr{G}(v)]{w} q_2 \xrightarrow[\mathscr{G}(v)]{u_2} 0 \quad ,$$

i.e., $u_1 w \overline{w} w u_2 \in \Gamma(v)$. In other words, for any words v and w and any $u \in \Gamma(v)$, if $w \rightsquigarrow u$ then w is also in $\Gamma(v)$, meaning exactly that $\Gamma(v)$ is upward-closed with respect to $\rightsquigarrow$.

Since $\Gamma'(v) \subseteq \Gamma(v)$, we deduce that $\Gamma(v)$ contains the upward closure of $\Gamma'(v)$, as expected. □

We now have enough material to embark in the proof of the implication (6) from the introduction, stating that if two expressions $e, f \in \text{Reg}_X^{\vee}$ are equal for all interpretations in all relational models, then $\mathcal{cl}(\mathbf{e}) = \mathcal{cl}(\mathbf{f})$.

Proof. Bloom et al. [BÉS95] consider specific relational interpretations: for any word $u \in \mathbf{X}^\star$ and for any letter $x \in \mathbf{X}$, they define

$$\phi_u(x) \triangleq \{(i-1, i) \mid u(i) = x\} \cup \{(i, i-1) \mid u(i) = x'\} \subseteq \{0, \dots, n\}^2 \quad ,$$

where $n \triangleq |u|$. The key property of those interpretations is the following:

$$(0, n) \in \widehat{\phi_u}(v) \Leftrightarrow v \rightsquigarrow^\star u \ . \tag{10}$$

We give a new proof of this property, by using the automaton $\Phi(u)$ depicted in Figure 2. By definition of $\Phi(u)$ and ϕ_u, we have that

$$(i, j) \in \phi_u(x) \Leftrightarrow i \xrightarrow{x}_{\Phi(u)} j \ .$$

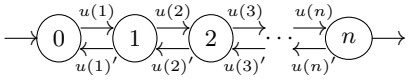


Fig. 2: Automaton $\Phi(u)$, with $|u| = n$.

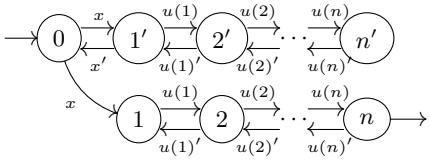


Fig. 3: Automaton $\Phi'(xu)$, with $|u| = n$, language equivalent to $\Phi(xu)$.

Therefore, proving (10) amounts to proving

$$v \in L(\Phi(u)) \Leftrightarrow v \rightsquigarrow^\star u \ . \tag{11}$$

First notice that $i \xrightarrow{x}_{\Phi(u)} j \Leftrightarrow j \xrightarrow{x'}_{\Phi(u)} i$. We can extend this to paths (as in the proof of Proposition 8) and then prove that if $s \rightsquigarrow t$ and $i \xrightarrow{t}_{\Phi(u)} j$ then $i \xrightarrow{s}_{\Phi(u)} j$. As u is clearly in $L(\Phi(u))$, any v such that $v \rightsquigarrow^\star u$ is also in $L(\Phi(u))$.

We proceed by induction on u for the other implication. The case $u = \epsilon$ being trivial, we consider $v \in L(\Phi(xu))$. We introduce a second automaton $\Phi'(xu)$ given in Figure 3, that recognises the same language as $\Phi(xu)$. The upper part of this automaton is actually the automaton $\mathscr{G}(\overline{xu})$ (as given in Figure 1), recognising the language $\Gamma(\overline{xu})$. Moreover, the lower part starting from state 1 is the automaton $\Phi(u)$. This allows us to obtain that $L(\Phi(xu)) = \Gamma(\overline{xu})xL(\Phi(u))$. Hence, for any $v \in L(\Phi(xu))$, there are $v_1 \in \Gamma(\overline{xu})$ and $v_2 \in L(\Phi(u))$ such that $v = v_1 x v_2$. By induction, we get $v_2 \rightsquigarrow^\star u$, and by Proposition 8 we know that $v_1 \rightsquigarrow^\star \overline{w}w$, with $w \in \text{suffixes}(\overline{xu})$. That means that $\overline{xu} = tw$, for some word t, so $xu = \overline{tw} = \overline{w}\ \overline{t}$. If we put everything back together:

$$v = v_1 x v_2 \rightsquigarrow^\star v_1 xu \rightsquigarrow^\star \overline{w}wxu = \overline{w}w\overline{w}\ \overline{t} \rightsquigarrow \overline{w}\ \overline{t} = xu \ .$$

This concludes the proof of (11), and thus (10).

We follow Bloom et al.'s proof [BÉS95] to deduce that the implication (6) from the introduction holds: we first prove that for all $e \in \text{Reg}_{\mathbf{X}}$, we have

$$\begin{aligned} u \in \mathcal{cl}(\llbracket e \rrbracket) &\Leftrightarrow \exists v \in \llbracket e \rrbracket, v \rightsquigarrow^\star u && \text{(by definition)} \\ &\Leftrightarrow \exists v \in \llbracket e \rrbracket, (0,n) \in \widehat{\phi_u}(v) && \text{(by (10))} \\ &\Leftrightarrow (0,n) \in \widehat{\phi_u}(e) \ . \end{aligned}$$

(For the last line, we use the fact that for any relational interpretation ϕ, we have $\widehat{\phi}(e) = \bigcup_{w \in [\![e]\!]} \widehat{\phi}(w)$.)

Furthermore, as $\phi_u(x') = \phi_u(x)^\vee$, we can prove that $\widehat{\phi_u}(e) = \widehat{\phi_u}(\mathbf{e})$ (see Appendix A.3). Therefore, for all expressions $e, f \in \mathrm{Reg}_X^\vee$ such that $e \equiv_{\mathrm{Rel}^\vee} f$, we have $\widehat{\phi_u}(\mathbf{e}) = \widehat{\phi_u}(e) = \widehat{\phi_u}(f) = \widehat{\phi_u}(\mathbf{f})$, and we deduce that $\mathcal{cl}([\![\mathbf{e}]\!]) = \mathcal{cl}([\![\mathbf{f}]\!])$ thanks to the above characterisation. □

2 Closure of an automaton

The problem here is the following: given two regular expressions $e, f \in \mathrm{Reg}_X^\vee$, how to decide $\mathcal{cl}([\![\mathbf{e}]\!]) = \mathcal{cl}([\![\mathbf{f}]\!])$? We follow the approach proposed by Bloom et al.: given an automaton recognising a language L, we show how to construct an automaton recognising $\mathcal{cl}(L)$. To solve the initial problem, it then suffices to build two automata recognising $[\![\mathbf{e}]\!]$ and $[\![\mathbf{f}]\!]$, to apply a construction to obtain two automata for $\mathcal{cl}([\![\mathbf{e}]\!])$ and $\mathcal{cl}([\![\mathbf{f}]\!])$, and to check those for language equivalence.

As a starting point, we first recall the construction proposed in [BÉS95].

2.1 The original construction

This construction uses the transition monoid of the input automaton:

Definition 9 (Transition monoid)**.** Let $\mathscr{A} = \langle Q, \Sigma, q_0, T, \delta \rangle$ be a deterministic automaton. Each word $u \in \Sigma^\star$ induces a function $u_\mathscr{A} : Q \to Q$ which associates to a state p the state q obtained by following the unique path from p labelled by u. The *transition monoid* of $\mathscr{A}$, written $M_\mathscr{A}$, is the set of functions $Q \to Q$ induced by words of $\Sigma^\star$, equipped with the composition of functions and the identity function.

This monoid is finite, and its subsets form a Kleene Algebra. Bloom et al. then proceed to define the closure automaton in the following way:

Theorem 10 (Closure automaton of [BÉS95])**.** *Let $L \subseteq \mathbf{X}^\star$ be a regular language, recognised by the deterministic automaton $\mathscr{A} = \langle Q, \mathbf{X}, q_0, Q_f, \delta \rangle$. Let $M_\mathscr{A}$ be the transition monoid of $\mathscr{A}$. Then the following deterministic automaton recognises $\mathcal{cl}(L)$:*

$$\mathscr{B} \triangleq \langle \mathcal{P}(M_\mathscr{A}) \times \mathcal{P}(M_\mathscr{A}), \mathbf{X}, (\{\epsilon_\mathscr{A}\}, \{\epsilon_\mathscr{A}\}), T, \delta_1 \rangle$$
$$\text{with } T \triangleq \{(F, G) \mid \exists u_\mathscr{A} \in F : u_\mathscr{A}(q_0) \in Q_f\},$$
$$\text{and } \delta_1((F, G), x) \triangleq \left(F \cdot \{x_\mathscr{A}\} \cdot \left((\{x'_\mathscr{A}\} \cdot G \cdot \{x_\mathscr{A}\})^\star\right),\ (\{x'_\mathscr{A}\} \cdot G \cdot \{x_\mathscr{A}\})^\star\right).$$

An important idea in this construction, that inspired our own, is the transition rule for the second component above. Let us write $\delta_2(G, x)$ for the expression $(\{x'_\mathscr{A}\} \cdot G \cdot \{x_\mathscr{A}\})^\star$, so that the definition of δ_1 can be reformulated as

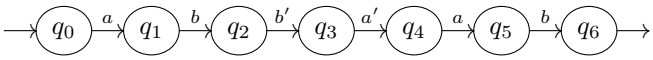
$$\delta_1((F, G), x) = (F \cdot \{x_\mathscr{A}\} \cdot \delta_2(G, x),\ \delta_2(G, x)).$$

With that in mind, one can see the second component as some kind of *history*, that runs on its own, and is used at each step to enrich the first component. At this point, it might be interesting to notice that the formula for $\delta_2(G, x)$ closely resembles the one for $\Gamma(wx) = (x'\Gamma(w)x)^\star$, which we defined in Section 1.2.

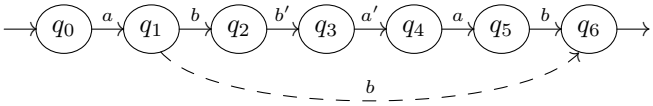
2.2 Intuitions

Let us forget the above construction, and try to build a closure automaton. One way would be to simply add transitions to the initial automaton. This idea comes naturally when one realises that if $u \rightsquigarrow^\star v$, then v is obtained by erasing some subwords from u: at each reduction step $u_1 w \overline{w} w u_2 \rightsquigarrow u_1 w u_2$ we just erase $\overline{w}w$. To "erase" such subwords using an automaton, it suffices to allow one to jump along certain paths.

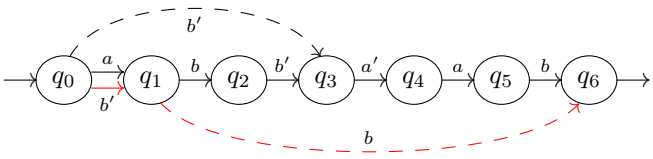
Suppose for instance that we start from the following automaton:



We can detect the pattern $ab\overline{ab}ab$, and allow one to "jump" over it when reading the last letter of the root of the pattern, in this case the b in second position. Our automaton thus becomes:



However, this approach is too naive, and it quickly leads to errors. If for instance we slightly modify the above example by adding a transition labelled by b' between q_0 and q_1, the same method leads to the following automaton, by detecting the patterns $b'bb'$ between q_0 and q_3 and $abb'a'ab$ between q_0 and q_6.



The problem is that the word $b'b$ is now wrongly recognised in the produced automaton. What happens here is that we can use the jump from q_1 to q_6, even though we didn't read the prerequisite for doing so, in this case the a constituting the beginning of the root ab of pattern $ab\overline{ab}ab$. (Note that the dual idea, consisting in enabling a jump when reading the first letter of the root of the pattern, would lead to similar problems.)

A way to prevent that, which was implicitly introduced in the original construction, consists in using a notion of *history*. The states of the closure automaton will be pairs of a state in the initial automaton and a history. That will allow us to distinguish between the state q_1 after reading a and the state q_1 after reading b', and to specify which jumps are possible considering what has been

previously read. In the construction given in [BÉS95], the history is given by an element of $\mathcal{P}(M_{\mathscr{A}})$, in the second component of the states (the "G" part). We will define a history as a set of words allowing for the same jumps, using $\Gamma(w)$.

2.3 Our construction

We have shown in Section 1.2 that $\forall u \in \Gamma(w), \exists v \in \text{suffixes}(w) : u \rightsquigarrow^\star \overline{v}v$, so we do have a characterisation of the words "allowing jumps" after having read some word w. The problem is that we want a finite number of possible histories, and there are infinitely many $\Gamma(w)$ (for instance, all the $\Gamma(a^n)$ are different). To get that, we will project $\Gamma(w)$ on the automaton. Let us consider a non-deterministic automaton $\mathscr{A} = \langle Q, \mathbf{X}, I, T, \Delta \rangle$ recognising a language L.

Definition 11. For any word $w \in \mathbf{X}^\star$ we define the relation $\gamma(w)$ between states of $\mathscr{A}$ by $\gamma(\epsilon) \triangleq \mathrm{Id}_Q$ and $\gamma(wx) = (\Delta(x') \circ \gamma(w) \circ \Delta(x))^\star$.

One can notice right away the strong relationship between γ and Γ:

Proposition 12. $\forall w, q_1, q_2,\ \ (q_1, q_2) \in \gamma(w) \ \Leftrightarrow\ \exists u \in \Gamma(w) : q_1 \xrightarrow{u}_{\mathscr{A}} q_2$.

This result is straightforward once one realises that $\gamma(w) = \widehat{\sigma}(\Gamma(w))$ with $\sigma(x) = \Delta(x)$. By composing Propositions 8 and 12 we eventually obtain that $((q_1, q_2) \in \gamma(w))$ iff $\exists u : q_1 \xrightarrow{u}_{\mathscr{A}} q_2$ and $u \rightsquigarrow^\star \overline{v}v$, with v a suffix of w.

The set Q being finite, γ has a finite index and one can define a finite set of histories as follows:

Definition 13. Let $\sim_\gamma$ be the kernel of γ: $u \sim_\gamma v$ iff $\gamma(u) = \gamma(v)$. We define the set G as the quotient of $\mathbf{X}^\star$ by $\sim_\gamma$. We denote by $[w]$ the elements of G, in such a way that $[u] = [v] \Leftrightarrow u \sim_\gamma v \Leftrightarrow \gamma(u) = \gamma(v)$.

We now have all the tools required for our construction of the closure of $\mathscr{A}$:

Theorem 14 (Closure Automaton)**.** *The closure of the language L is recognised by the automaton $\mathscr{A}' \triangleq \langle Q \times G, \mathbf{X}, I \times \{[\epsilon]\}, T \times G, \Delta' \rangle$ with:*

$$\Delta' = \{((q_1, [w]), x, (q_2, [wx])) \mid (q_1, q_2) \in \Delta(x) \circ \gamma(wx)\}.$$

We shall write L' for the language recognised by $\mathscr{A}'$. One can read the set of transitions as "from a state q_1 with an history w, perform a step x in the automaton $\mathscr{A}$, and then a jump compatible with wx, which becomes the new history". One can see, from the definition of Δ' and Proposition 12 that :

$$(q_1, [u]) \xrightarrow{x}_{\mathscr{A}'} (q_2, [ux]) \ \Leftrightarrow\ \exists (q_3, v) \in Q \times \Gamma(ux) : q_1 \xrightarrow{x}_{\mathscr{A}} q_3 \xrightarrow{v}_{\mathscr{A}} q_2. \quad (12)$$

Now we prove the correctness of this construction. First recall the notion of *simulation* [Mil89]:

Definition 15 (Simulation)**.** A relation R between the states of two automata $\mathscr{A}$ and $\mathscr{B}$ is a *simulation* if for all $(p, q) \in R$ we have (a) if $p \xrightarrow{x}_{\mathscr{A}} p'$, then there exists q' such that $q \xrightarrow{x}_{\mathscr{B}} q'$ and $(p', q') \in R$, and (b) if $p \in T_{\mathscr{A}}$ then $q \in T_{\mathscr{B}}$.

We say that $\mathscr{A}$ *is simulated by* $\mathscr{B}$ if there is a simulation R such that for any $p_0 \in I_{\mathscr{A}}$, there is $q_0 \in I_{\mathscr{B}}$ such that $p_0\ R\ q_0$.

The following property of γ is proved by exhibiting such a simulation:

Proposition 16. *For all words $u, v \in \mathbf{X}^\star$ such that $u \rightsquigarrow v$, we have $\gamma(u) \subseteq \gamma(v)$.*

Proof. First, notice that $\Gamma(u) \subseteq \Gamma(v) \Rightarrow \gamma(u) \subseteq \gamma(v)$, using Proposition 12. It thus suffices to prove $u \rightsquigarrow v \Rightarrow \Gamma(u) \subseteq \Gamma(v)$, which can be rewritten as $\Gamma(u_1 w \overline{w} w u_2) \subseteq \Gamma(u_1 w u_2)$. We can drop u_2 (it is clear that $\Gamma(w_1) \subseteq \Gamma(w_2) \Rightarrow \forall x \in \mathbf{X}, \Gamma(w_1 x) \subseteq \Gamma(w_2 x)$, from the definition of Γ): we now have to prove that $\Gamma(u_1 w \overline{w} w) \subseteq \Gamma(u_1 w)$. The proof of this inclusion relies on the fact that the automaton $\mathscr{G}(u_1 w \overline{w} w)$ is simulated by the automaton $\mathscr{G}(u_1 w)$, see Appendix A.4 for a formal definition of this simulation. □

We define an order relation $\preccurlyeq$ on the states of the produced automaton $(Q \times G)$, by $(p, [u]) \preccurlyeq (q, [v]) \triangleq p = q \wedge \gamma(u) \subseteq \gamma(v)$.

Proposition 17. *The relation $\preccurlyeq$ is a simulation for the automaton $\mathscr{A}'$.*

Proof. Suppose that $(p, [u]) \preccurlyeq (q, [v])$ and $(p, [u]) \xrightarrow{x}_{\mathscr{A}'} (p', [ux])$, i.e., $(p, p') \in \Delta_x \circ \gamma(ux)$. We have $p = q$ and $\gamma(u) \subseteq \gamma(v)$, hence $\gamma(ux) \subseteq \gamma(vx)$, and thus $(p, p') \in \Delta_x \circ \gamma(vx)$ meaning that $(p, [v]) \xrightarrow{x}_{\mathscr{A}'} (p', [vx])$. It remains to check that $(p', [ux]) \preccurlyeq (p', [vx])$, i.e., $\gamma(ux) \subseteq \gamma(vx)$, which we just proved. □

We may now prove that $L' = \mathcal{cl}(L)$.

Lemma 18. $L' \subseteq \mathcal{cl}(L)$

Proof. We prove by induction on u that for all q_0, q such that $(q_0, [\epsilon]) \xrightarrow{u}_{\mathscr{A}'} (q, [u])$, there exists v such that $v \rightsquigarrow^\star u$ and $q_0 \xrightarrow{v}_{\mathscr{A}} q$. The case $u = \epsilon$ is trivial.

If $(q_0, [\epsilon]) \xrightarrow{u}_{\mathscr{A}'} (q_1, [u]) \xrightarrow{x}_{\mathscr{A}'} (q, [ux])$, by induction one can find v_1 such that $q_0 \xrightarrow{v_1}_{\mathscr{A}} q_1$ and $v_1 \rightsquigarrow^\star u$. We also know (by (12) and Proposition 8) that there are some q_2, v_2 and $v_3 \in \mathrm{suffixes}(ux)$ such that $q_1 \xrightarrow{x}_{\mathscr{A}} q_2$, $v_2 \rightsquigarrow^\star \overline{v}_3 v_3$ and $q_2 \xrightarrow{v_2}_{\mathscr{A}} q$. We thus get

$$q_0 \xrightarrow{v_1}_{\mathscr{A}} q_1 \xrightarrow{x}_{\mathscr{A}} q_2 \xrightarrow{v_2}_{\mathscr{A}} q \text{ and } v_1 x v_2 \rightsquigarrow^\star u x v_2 \rightsquigarrow^\star u x \overline{v}_3 v_3 \rightsquigarrow u x.$$

By choosing $q \in T$, we obtain the desired result. □

Lemma 19. $L \subseteq L'$

Proof. This is actually very simple. First notice that for all u, $\gamma(u)$ is a reflexive relation, hence $q_1 \xrightarrow{x}_{\mathscr{A}} q_2$ entails $\forall u, (q_1, [u]) \xrightarrow{x}_{\mathscr{A}'} (q_2, [ux])$. This means that the relation R defined by $p \; R \; (q, [w]) \Leftrightarrow p = q$ is a simulation between $\mathscr{A}$ and $\mathscr{A}'$, and thus $L = L(\mathscr{A}) \subseteq L(\mathscr{A}') = L'$. □

Lemma 20. *L' is downward-closed for $\rightsquigarrow$.*

A technical lemma is required to establish this closure property:

Lemma 21. *If $(q_1, [uw]) \xrightarrow{x}_{\mathscr{A}'} (q_2, [uwx]) \xrightarrow{\overline{wx}\; wx}_{\mathscr{A}'} (q_3, [uwx\ \overline{wx}\ wx])$, then $(q_1, [uw]) \xrightarrow{x}_{\mathscr{A}'} (q_3, [uwx])$.*

Proof sketch. The proof being quite verbose and dry, we shall only give a sketch of it here, referring to Appendix A.5 for a detailed one. If $|w| = n$ and $|u| = m$, the premise can be equivalently stated:

$$(q_1, [(uw)|_{m+n-1}]) \xrightarrow{w(n)}_{\mathscr{A}'} (q_2, [uw]) \xrightarrow{\overline{w}w}_{\mathscr{A}'} (q_3, [uw\overline{w}w]).$$

(Recall that $u|_i$ denotes the prefix of length i of a word u.) Let us write $\Gamma_i = \Gamma((uw\overline{w}w)|_{m+n+i}) = \Gamma(uw(\overline{w}w)|_i)$ and $x_i = (uw\overline{w}w)(n+m+i)$ for $0 \leqslant i \leqslant 2n$. By Proposition 12 and the definition of $\mathscr{A}'$, we can show that there are $v_i \in \Gamma_i$ such that the execution above can be lifted into an execution in $\mathscr{A}$:

$$q_1 \xrightarrow{x_0 v_0 x_1 v_1 \cdots x_i v_i \cdots x_{2n} v_{2n}}_{\mathscr{A}} q_3.$$

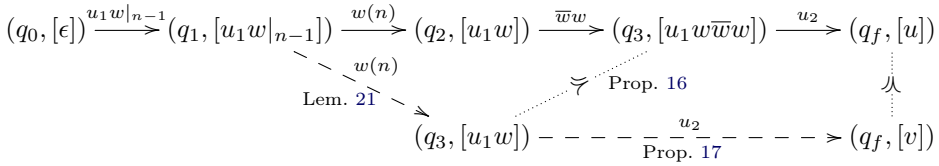
Then one can prove by recurrence on i and using Proposition 8 that:

$$\forall i, \exists t_i \in \Gamma(uw) : (\overline{w}w)|_i v_i \rightsquigarrow^\star t_i(\overline{w}w)|_i. \tag{13}$$

We deduce that $v_0 x_1 v_1 \cdots x_i v_i \cdots x_{2n} v_{2n} \rightsquigarrow^\star t_0 t_1 \cdots t_{2n} \overline{w}w \in \Gamma(uw)^{2n+2} \subseteq \Gamma(uw)$. By Proposition 8, this means that $v_0 x_1 v_1 \cdots x_i v_i \cdots x_{2n} v_{2n}$ is in $\Gamma(uw)$, so that $(q_1, q_3) \in \Delta(w(n)) \circ \gamma(uw)$, and $(q_1, [uw|_{n-1}]) \xrightarrow{w(n)}_{\mathscr{A}'} (q_2, [uw])$. □

With this intermediate lemma, one can obtain a succinct proof of Lemma 20:

Proof. The statement of the lemma is equivalent to saying that if $u \rightsquigarrow v$ with $u \in L'$ then v is also in L'. Consider $u = u_1 w \cdot \overline{w} \cdot w u_2$ and $v = u_1 w u_2$ with $|w| = n \geqslant 1$ (the case where $w = \epsilon$ doesn't hold any interest since it implies that $u = v$). By combining Lemma 21 and Proposition 17 we can build the following diagram:



□

Lemmas 19 and 20 tell us that L' is closed and contains L, so by definition of the closure of a language, we get $\mathcal{cl}(L) \subseteq L'$. Lemma 18 gives us the other inclusion, thus proving Theorem 14.

3 Analysis and consequences

3.1 Relationship with [BÉS95]'s construction

As suggested by an anonymous referee, one can also formally relate our construction to the one from [BÉS95]: we give below an explicit and rather natural

bisimulation relation between the automata produced by both these methods. This results in an alternative correctness proof of our construction, by reducing it to the correctness of the one from [BÉS95].

We first make the two constructions comparable: the original construction, because it considers the transition monoid, takes as input a deterministic automaton. It returns a deterministic automaton. Instead, our construction does not require determinism in its input, but produces a non-deterministic automaton. We thus have to ask of both methods to accept as their input a *non-deterministic* automaton, and to return a *deterministic* automaton.

For our construction, the straightforward thing to do would be to determinise the automaton afterwards. We can actually do better, by noticing that from a state $(p, [u])$, reading some x, there may be a lot of accessible states, but all of their histories (second components) will be equal to $[ux]$. So in order to get a deterministic automaton, one only has to perform the power-set construction on the first component of the automaton. This way, we get an automaton $\mathscr{A}_1$ with states in $\mathcal{P}(Q) \times G$ and a transition function

$$\delta_1((P, [u]), x) = (P \cdot (\Delta(x) \circ \gamma(ux)), [ux]) .$$

The original construction can also be adjusted very easily: first build a deterministic automaton $\mathcal{D}$ with the usual powerset construction, then apply the construction as described in Theorem 10 to get an automaton which we call $\mathscr{A}_2$. An important thing here is to understand the shape of the resulting transition monoid $M_{\mathcal{D}}$: its elements are functions over sets of states (because of the power-set construction) induced by words; more precisely, they are sup-semilattice homomorphisms, and they are in bijection with binary relations on states.

Define the following KA-homomorphism from $\mathcal{P}(M_{\mathcal{D}})$ to $\mathcal{P}(Q^2)$:

$$i(F) = \{(p, q) \mid \exists u_{\mathcal{D}} \in F : q \in u_{\mathcal{D}}(\{p\})\} .$$

(That i is a KA-homomorphism comes from the fact that the elements of $M_{\mathcal{D}}$ are themselves sup-semilattice homomorphisms on $\mathcal{P}(Q)$.) We can check that for all $x \in \mathbf{X}$, we have

$$\begin{aligned} i(\{x_{\mathcal{D}}\}) &= \{(p, q) \mid q \in x_{\mathcal{D}}(\{p\})\} = \{(p, q) \mid q \in \delta(\{p\}, x)\} \\ &= \left\{(p, q) \;\middle|\; p \xrightarrow{x}_{\mathscr{A}} q\right\} = \Delta(x) , \end{aligned}$$

It follows that the following relation is a bisimulation between $\mathscr{A}_1$ and $\mathscr{A}_2$.

$$\{((Q, [u]),\ (F, G)) \mid Q = I \cdot i(F) \text{ and } \gamma(u) = i(G)\}$$

In Appendix A.6 we give a detailed proof of this.

3.2 Complexity

Because we are speaking about algorithms rather than actual programs, it is a bit difficult to give accurate complexity bounds, considering the many possible

data structures appearing during the computation. However, one may think that a relevant complexity measure of the final algorithm (for deciding equality in KAC) could be the size of the produced automata. In the following the *size* of an automaton is its number of states. In order to give a fair comparison, we will consider the generic algorithms given in the previous subsection, taking as their input a *non-deterministic* automaton, and returning a *deterministic* automaton.

Let us begin by evaluating the size of the automaton produced by the method in [BÉS95], given a non-deterministic automaton of size n. As explained above, the states of the constructed transition monoid ($M_{\mathcal{D}}$) are in bijection with the binary relations on Q. There are thus at most 2^{n^2} elements in this monoid. We deduce that the final automaton, whose states are pairs of subsets of $M_{\mathcal{D}}$ has at most $2^{2^{n^2}} \times 2^{2^{n^2}} = 2^{2^{n^2+1}}$ states.

Now with the deterministic version of our construction, the states are in the set $\mathcal{P}(Q) \times G$. Since G is the set of equivalence classes of $\sim_\gamma$ and γ has values in the reflexive binary relations over Q, we know that $\sim_\gamma$ has less than $2^{n\times(n-1)}$ elements. Hence we can see that $|\mathcal{P}(Q) \times G| \leqslant 2^n \times 2^{n\times(n-1)} = 2^{n^2}$, which is significantly smaller than the $2^{2^{n^2+1}}$ states we get with the other construction.

3.3 A polynomial-space algorithm

The above upper-bound on the number of states of the automata produced by our construction allows us to show that the problem of equivalence in KAC is in PSpace (the problem was already known to be PSpace-hard since KAC is conservative over KA, which is PSpace-complete [MS73]).

Recall that the equivalence of two deterministic automata $\mathscr{A}$ and $\mathscr{B}$ is in LogSpace. The algorithm to show that relies on the fact that $\mathscr{A}$ and $\mathscr{B}$ are different if and only if there is a word w in the difference of $L(\mathscr{A})$ and $L(\mathscr{B})$ such that $|w| \leqslant |\mathscr{A}| \times |\mathscr{B}|$. With that in mind, we can give a non-deterministic algorithm, by simulating a computation in both automata with a letter chosen non-deterministically at each step, with a counter to stop us at size $|\mathscr{A}| \times |\mathscr{B}|$. The resulting algorithm will only have to store the counter of size $\log(|\mathscr{A}| \times |\mathscr{B}|)$ and the two current states.

For our problem, the first step is to compute $\mathbf{e}$ and $\mathbf{f}$ from the regular expressions with converse e and f. It is obvious that such a transformation can be done in linear time and space, by a single sweep of both e and f. Then we have to build automata for $\mathbf{e}$ and $\mathbf{f}$. Once again this is a very light operation: if one considers for instance the position automaton (also called Glushkov's construction [Glu61]), we obtain automata of respective sizes $n = |\mathbf{e}| + 1 = |e| + 1$ and $m = |\mathbf{f}| + 1 = |f| + 1$, where $|\cdot|$ denotes the number of variable leaves of a regular expression (possibly with converse).

Our construction then produces closed automata of size at most 2^{n^2} and 2^{m^2}, so that the non-deterministic algorithm to check their equivalence needs to scan all words of size smaller than by $2^{n^2} \times 2^{m^2} = 2^{n^2+m^2}$. The counter used to bound the recursion depth can thus be stored in polynomial space (n^2+m^2). It is worth

input : Two regular expressions with converse $e, f \in \mathrm{Reg}^{\vee}_X$
output: A Boolean, saying whether or not $\mathrm{KAC} \vdash e = f$.

$\mathscr{A}_1 = \langle Q_1, \mathbf{X}, I_1, T_1, \Delta_1 \rangle \leftarrow$ Glushkov' automaton recognising $[\![\mathbf{e}]\!]$;
$\mathscr{A}_2 = \langle Q_2, \mathbf{X}, I_2, T_2, \Delta_2 \rangle \leftarrow$ Glushkov' automaton recognising $[\![\mathbf{f}]\!]$;
$N \leftarrow (2^{(|e|+1)^2} \times 2^{(|f|+1)^2})$; /* N gets a value $\geqslant |\mathcal{L}(\mathscr{A}_1)| \cdot |\mathcal{L}(\mathscr{A}_2)|$ */
$((P_1, R_1), (P_2, R_2)) \leftarrow ((I_1, \mathrm{Id}_{Q_1}), (I_2, \mathrm{Id}_{Q_1}))$;
while $N > 0$ **do**
  $N \leftarrow N - 1$; /* N bounds the recursion depth */
  $f_1 \leftarrow$ is_empty$(P_1 \cap T_1)$;
  $f_2 \leftarrow$ is_empty$(P_2 \cap T_2)$;
  if $f_1 = f_2$ **then**
    $x \leftarrow$random(**X**); /* Non-deterministic choice */
    $(R_1, R_2) \leftarrow ((\Delta_1(x') \circ R_1 \circ \Delta_1(x))^\star, (\Delta_2(x') \circ R_2 \circ \Delta_2(x))^\star)$;
    $(P_1, P_2) \leftarrow (P_1 \cdot (\Delta_1(x) \circ R_1), P_2 \cdot (\Delta_2(x) \circ R_2))$;
  else
    return false; /* A difference appeared for some word, $e \neq f$ */
  end
end
return true; /* There was no difference, KAC $\vdash e = f$ */

Algorithm 1: A PSpace algorithm for KAC

mentionning here that with the automata constructed in [BÉS95], the counter would have size $2^{n^2+1} + 2^{m^2+1}$ which is not a polynomial.

Now the last two important things to worry about are the representation of the states of the closure automata, in particular their "history" component, and the way to compute their transition function. Let us focus on the automaton for e and let Q be the set of states of the Glushkov automaton built out of it.

- For the state representation, one needs to represent an equivalence class $[u] \in G$ by its image under γ: while the smallest word $w \in [u]$ may be quite long, $\gamma(u)$ is just a binary relation on Q. We shall thus represent the states in the determinised closure automaton as pairs of a set of states in Q and a binary relation (set of pairs) over Q. Such a pair can be stored in polynomial space (recall that $|Q| = n = |e| + 1$).
- For computing the transition function, the image of a pair $(\{q_1, \cdots, q_k\}, R)$ (with $R \subseteq Q^2$) by a letter $x \in \mathbf{X}$ is done in two steps: first the relation becomes $R' = (\Delta(x') \circ R \circ \Delta(x))^\star$, then the set of states becomes $\{q \mid \exists i, 1 \leqslant i \leqslant k : (q_i, q) \in \Delta(x) \circ R'\}$. Those computations take place in PSpace. (The composition of two relations in Q^2 can be performed in space $\mathcal{O}\left(|Q|^2\right)$, and the same holds for the reflexive and transitive closure of a relation R by building the powers $(R + \mathrm{Id}_Q)^{2^k}$ and keeping a copy of the previous iteration to stop when the fixed-point is reached.)

Summing up, we obtain Algorithm 1, which is PSpace.

Conclusion

Starting from the works of Bernátsky, Bloom, Ésik and Stefanescu, we gave a new and more efficient algorithm to decide the theory KAC. This algorithm relies on a new construction for the closure of an automaton, which allowed us to show that the problem was in fact in the complexity class PSpace.

To prove the correctness of our construction, we used the family of regular languages $\Gamma(w)$ ($G(w^\vee)$ in [BÉS95]), and we establish its main properties using a proper finite automata characterisation. Moreover, this function allowed us to reformulate the proof of the completeness of the reduction from equality in $\mathrm{Rel}^\vee$ to equivalence of closed automata (implication (6) from the introduction).

As an exercise, we have implemented and tested the various constructions and algorithms in an OCaml program which is available online[1].

To continue this work, we would like to implement our algorithm in the proof assistant Coq, as a tactic to automatically prove the equalities in KAC—as it has already been done for the theories KA and KAT. The simplifications we propose in this paper give us hope that such a task is feasible. The main difficulty certainly lies in the formalisation of the completeness proof of KAC (implication (7) from the introduction): the proof given in [ÉB95] uses yet another automaton construction for the closure, which is much more complicated than the one used in [BÉS95], and which seems quite difficult to formalise in Coq. We hope to find an alternative completeness proof, by exploiting the simplicity of the presented construction.

Acknowledgements. We are grateful to the anonymous referees who suggested us the alternative proof of correctness which we provide in Section 3.1, and who helped us to improve this paper.

References

[BÉS95] Bloom, S. L., Ésik, Z., Stefanescu, G.: Notes on equational theories of relations. Algebra Universalis 33, 98–126 (1995)

[Bof90] Boffa, M.: Une remarque sur les systèmes complets d'identités rationnelles. Informatique Théorique et Applications 24, 419–428 (1990)

[Bof95] Boffa, M.: Une condition impliquant toutes les identités rationnelles. Informatique Théorique et Applications 29, 515–518 (1995)

[Con71] Conway, J. H.: Regular algebra and finite machines. Chapman and Hall Mathematics Series (1971)

[ÉB95] Ésik, Z., Bernátsky, L.: Equational properties of Kleene algebras of relations with conversion. Theoretical Computer Science 137, 237–251 (1995)

[Glu61] Glushkov, V. M.: The abstract theory of automata. Russian Mathematical Surveys 16, 1 (1961)

[Kle51] Kleene, S. C.: Representation of Events in Nerve Nets and Finite Automata. Memorandum. Rand Corporation (1951)

[1] http://perso.ens-lyon.fr/paul.brunet/cka.html

[Koz91] Kozen, D.: A Completeness Theorem for Kleene Algebras and the Algebra of Regular Events. In: LICS, pp. 214–225. IEEE Computer Society (1991)
[Kro90] Krob, D.: A Complete System of B-Rational Identities. In: ICALP, Lecture Notes in Computer Science, vol. 443, pp. 60–73. Springer (1990)
[MS73] Meyer, A., Stockmeyer., L. J.: Word problems requiring exponential time. In: Proc. ACM symposium on Theory of computing, pp. 1–9. ACM (1973)
[Mil89] Milner, R.: Communication and Concurrency. Prentice Hall (1989)
[Red64] Redko, V. N.: On defining relations for the algebra of regular events. Ukrainskii Matematicheskii Zhurnal pp. 120–126 (1964)
[Sal66] Salomaa, A.: Two Complete Axiom Systems for the Algebra of Regular Events. J. ACM 13, 158–169 (1966)

A Omitted proofs

A.1 Proof of Equation (9)

We will show here that $\widehat{\eta}(e) = \widehat{\eta}(f)$ implies that $[\![e]\!] = [\![f]\!]$, for e and f regular expressions over $\mathbf{X}$.

It is well known that for any expression $e \in \mathrm{Reg}_{\mathbf{X}}$, for any $\sigma : \mathbf{X} \longrightarrow \mathcal{P}(\Sigma^\star)$,

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

Consider the following partial function: $i : \begin{array}{rcl} X_\bullet^\star & \longrightarrow & \mathbf{X}^\star \\ \epsilon & \longmapsto & \epsilon \\ x \bullet w & \longmapsto & x \cdot i(w) \\ \bullet x w & \longmapsto & x' \cdot i(w). \end{array}$

We will write $\widehat{i}$ the function $[W \mapsto \{i(w) \mid w \in W\}]$. We will show by induction on $w \in \mathbf{X}^\star$ that $\widehat{i} \circ \widehat{\eta}(w) = \{w\}$:

- $\widehat{i} \circ \widehat{\eta}(\epsilon) = \widehat{i}(\{\epsilon\}) = \{\epsilon\}$;
- if $x \in X$, then

$$\begin{aligned} \widehat{i} \circ \widehat{\eta}(xw) &= \widehat{i}(\eta(x) \cdot \widehat{\eta}(w)) && (\widehat{\eta} \text{ is a morphism}) \\ &= \widehat{i}(\{x\bullet\} \cdot \widehat{\eta}(w)) && (\text{definition of } \eta) \\ &= \{x\} \cdot (\widehat{i} \circ \widehat{\eta}(w)) && (\text{definition of } i) \\ &= \{xw\}; && (\text{induction hypothesis}) \end{aligned}$$

- and similarly if $x' \in X'$, then $\widehat{i} \circ \widehat{\eta}(x'w) = \widehat{i}(\{\bullet x\} \cdot \widehat{\eta}(w)) = \{x'\} \cdot (\widehat{i} \circ \widehat{\eta}(w)) = \{x'w\}$.

Thus, we get that:

$$[\![e]\!] = \bigcup_{w \in [\![e]\!]} \{w\} = \bigcup_{w \in [\![e]\!]} \widehat{i} \circ \widehat{\eta}(w) = \widehat{i}\left(\bigcup_{w \in [\![e]\!]} \widehat{\eta}(w) \right) = \widehat{i}(\widehat{\eta}(e)).$$

Thus we get $[\![e]\!] = \widehat{i}(\widehat{\eta}(e)) = \widehat{i}(\widehat{\eta}(f)) = [\![f]\!]$.

A.2 Proof of Proposition 8

Let us prove the first implication of Proposition 8:

$$\forall \mathbf{w} \in \mathbf{X}^\star, \forall u \in \Gamma(\mathbf{w}), \exists v \in \mathrm{suffixes}(\mathbf{w}) : u \rightsquigarrow^\star \overline{v} v.$$

We will proceed by induction on $\mathbf{w}$:

1. If $\mathbf{w} = \epsilon$, then $u \in \Gamma(\epsilon) = \{\epsilon\}$. So $u = \epsilon \rightsquigarrow^0 \overline{\epsilon}\epsilon$ and obviously $\epsilon \in \mathrm{suffixes}(\epsilon)$.

2. Otherwise $\mathbf{w} = wx$, and $u \in \Gamma(wx) = (x'\Gamma(w)x)^\star$. Thus we know that for some $n \in \mathbb{N}$, $u \in (x'\Gamma(w)x)^n$. We now will prove by recurrence on n that $u \in (x'\Gamma(w)x)^n \Rightarrow \exists v \in \text{suffixes}(wx) : u \rightsquigarrow^\star \overline{v}v$:
   (a) If $n = 0$ then $u = \epsilon \rightsquigarrow^0 \overline{\epsilon}\epsilon$ and $\epsilon \in \text{suffixes}(wx)$.
   (b) If $n = m + 1$ then we can introduce $u_1 \in \Gamma(w)$ and $u_2 \in (x'\Gamma(w)x)^m$ such that $u = x'u_1xu_2$.
      i. By induction hypothesis, $\exists v_1 \in \text{suffixes}(w)$ such that $u_1 \rightsquigarrow^\star \overline{v_1}v_1$.
      ii. By reccurence hypothesis, $\exists v_2 \in \text{suffixes}(wx)$ such that $u_2 \rightsquigarrow^\star \overline{v_2}v_2$.

   Thus we know that $u = x'u_1xu_2 \rightsquigarrow^\star x'\overline{v_1}v_1x\overline{v_2}v_2$. We will now do a case analysis on the length of v_2.
      i. If $|v_2| = 0$, then $v_2 = \epsilon$ so $u \rightsquigarrow^\star x'\overline{v_1}v_1x = \overline{v_1x}v_1x$.
      ii. If $|v_2| > 0$, as $v_2 \in \text{suffixes}(wx)$, we can write $v_2 = v_3x$ with $v_3 \in \text{suffixes}(w)$. We will now compare the sizes of v_1 and v_3, both being suffixes of w.
         A. If $|v_1| \leqslant |v_3|$, then $v_3 = v_4v_1$. Thus we have:

$$\begin{aligned} u \rightsquigarrow^\star x'\overline{v_1}v_1x\overline{v_3x}v_3x &= x'\overline{v_1}v_1xx'\overline{v_1}\ \overline{v_4}v_4v_1x \\ &= \overline{v_1x}v_1x\overline{v_1x}\ \overline{v_4}v_4v_1x \\ &\rightsquigarrow \overline{v_1x}\ \overline{v_4}v_4v_1x = \overline{v_2}v_2 \end{aligned}$$

         B. Otherwise we can write $v_1 = v_5v_3$ and thus:

$$\begin{aligned} u &\rightsquigarrow^\star x'\overline{v_5v_3}v_5v_3x\overline{v_3x}v_3x \\ &\rightsquigarrow \overline{v_3v_5x}v_5v_3x = \overline{v_1x}v_1x \end{aligned}$$

   So we have shown that either $u \rightsquigarrow^\star \overline{v_1x}v_1x$ or $u \rightsquigarrow^\star \overline{v_2}v_2$, and as we know that both v_1x and v_2 are suffixes of wx, we have finished.

A.3 Proof of $\widehat{\phi}_u(e) = \widehat{\phi}_u(\mathbf{e})$

We first give an alternative definition of $\mathbf{e}$: let χ and ξ be the following mutually recursive functions:

$\chi(\mathbb{0}) \triangleq \mathbb{0}$	$\xi(\mathbb{0}) \triangleq \mathbb{0}$
$\chi(\mathbb{1}) \triangleq \mathbb{1}$	$\xi(\mathbb{1}) \triangleq \mathbb{1}$
$\chi(x) \triangleq x$	$\xi(x) \triangleq x'$
$\chi(e + f) \triangleq \chi(e) + \chi(f)$	$\xi(e + f) \triangleq \xi(e) + \xi(f)$
$\chi(e \cdot f) \triangleq \chi(e) \cdot \chi(f)$	$\xi(e \cdot f) \triangleq \xi(f) \cdot \xi(e)$
$\chi(e^\star) \triangleq (\chi(e))^\star$	$\xi(e^\star) \triangleq (\xi(e))^\star$
$\chi(e^\vee) \triangleq \xi(e)$	$\xi(e^\vee) \triangleq \chi(e)$

χ and ξ are both functions mapping an expression in $\text{Reg}_X^\vee$ to an expression in $\text{Reg}_{\mathbf{X}}$. It is quite immediate that $\mathbf{e} = \nu(\tau(e)) = \chi(e)$.

Hence, what we want is to prove that $\widehat{\phi}_u(e) = \widehat{\phi}_u(\chi(e))$. Because of the mutually reccursive definition we gave, we will prove inductively on $e \in \text{Reg}_X^\vee$ the following:

$$\widehat{\phi}_u(\chi(e)) = \widehat{\phi}_u(e) \quad \wedge \quad \widehat{\phi}_u(\xi(e)) = \widehat{\phi}_u(e)^\vee$$

- $\chi(\mathbb{0}) = \mathbb{0}$ and $\widehat{\phi}_u(\xi(\mathbb{0})) = \widehat{\phi}_u(\mathbb{0}) = \mathbb{0} = \mathbb{0}^\vee = \widehat{\phi}_u(\mathbb{0})^\vee$, so this case and the case $\mathbb{1}$ don't hold any difficulty.
- $\widehat{\phi}_u(\chi(x)) = \widehat{\phi}_u(x)$, so no problem there, but $\widehat{\phi}_u(\xi(x)) = \widehat{\phi}_u(x') = \phi_u(x')$. By the definition of ϕ_u we get:

$$\begin{aligned}\phi_u(x') &= \{(i-1,i) \mid u(i) = x'\} \cup \{(i,i-1) \mid u(i) = x\} \\ &= \{(i,i-1) \mid u(i) = x'\}^\vee \cup \{(i-1,i) \mid u(i) = x\}^\vee \\ &= (\{(i,i-1) \mid u(i) = x'\} \cup \{(i-1,i) \mid u(i) = x\})^\vee \\ &= \phi_u(x)^\vee\end{aligned}$$

Every other case is then quite simple:

- $e + f$: $\begin{aligned}\widehat{\phi}_u(\chi(e+f)) &= \widehat{\phi}_u(\chi(e) + \chi(f)) = \widehat{\phi}_u(\chi(e)) \cup \widehat{\phi}_u(\chi(f)) \\ &= \widehat{\phi}_u(e) \cup \widehat{\phi}_u(f) = \widehat{\phi}_u(e+f)\end{aligned}$
  $\begin{aligned}\widehat{\phi}_u(\xi(e+f)) &= \widehat{\phi}_u(\xi(e) + \xi(f)) = \widehat{\phi}_u(\xi(e)) \cup \widehat{\phi}_u(\xi(f)) \\ &= \widehat{\phi}_u(e)^\vee \cup \widehat{\phi}_u(f)^\vee = \left(\widehat{\phi}_u(e) \cup \widehat{\phi}_u(f)\right)^\vee \\ &= \left(\widehat{\phi}_u(e+f)\right)^\vee\end{aligned}$
- $e \cdot f$: $\begin{aligned}\widehat{\phi}_u(\chi(e \cdot f)) &= \widehat{\phi}_u(\chi(e) \cdot \chi(f)) = \widehat{\phi}_u(\chi(e)) \circ \widehat{\phi}_u(\chi(f)) \\ &= \widehat{\phi}_u(e) \circ \widehat{\phi}_u(f) = \widehat{\phi}_u(e \cdot f)\end{aligned}$
  $\begin{aligned}\widehat{\phi}_u(\xi(e \cdot f)) &= \widehat{\phi}_u(\xi(f) \cdot \xi(e)) = \widehat{\phi}_u(\xi(f)) \circ \widehat{\phi}_u(\xi(e)) \\ &= \widehat{\phi}_u(f)^\vee \circ \widehat{\phi}_u(e)^\vee = \left(\widehat{\phi}_u(e) \circ \widehat{\phi}_u(f)\right)^\vee \\ &= \left(\widehat{\phi}_u(e \cdot f)\right)^\vee\end{aligned}$
- $e^\star$: $\widehat{\phi}_u(\chi(e^\star)) = \widehat{\phi}_u(\chi(e)^\star) = \left(\widehat{\phi}_u(\chi(e))\right)^\star = \left(\widehat{\phi}_u(e)\right)^\star = \widehat{\phi}_u(e^\star)$
  $\begin{aligned}\widehat{\phi}_u(\xi(e^\star)) &= \widehat{\phi}_u(\xi(e)^\star) = \left(\widehat{\phi}_u(\xi(e))\right)^\star = \left(\widehat{\phi}_u(e)^\vee\right)^\star \\ &= \left(\widehat{\phi}_u(e)^\star\right)^\vee = \widehat{\phi}_u(e^\star)^\vee\end{aligned}$
- $e^\vee$: $\widehat{\phi}_u(\chi(e^\vee)) = \widehat{\phi}_u(\xi(e)) = \widehat{\phi}_u(e)^\vee = \widehat{\phi}_u(e^\vee)$
  $\widehat{\phi}_u(\xi(e^\vee)) = \widehat{\phi}_u(\chi(e)) = \widehat{\phi}_u(e) = \widehat{\phi}_u(e)^{\vee\vee} = \widehat{\phi}_u(e^\vee)^\vee$

A.4 Proof of $\Gamma(uw\overline{w}w) \subseteq \Gamma(uw)$

We will prove in this section that for any $u, w \in \mathbf{X}^*$, $\Gamma(uw\overline{w}w) \subseteq \Gamma(uw)$. First recall that for any word w, the language $\Gamma(w)$ is recognised by the automaton given in Figure 1). With that in mind, we give in Figure 4 an abstract view of the automata recognising $\Gamma(uw\overline{w}w)$ and $\Gamma(uw)$ defined as before. With the notations of this figure, now define a relation $\leqslant$ as follows (this relation is also represented in dashed lines in Figure 4):

$$\begin{aligned} a_i &\leqslant b_i && \text{for all } i \leq n+m \ , \\ a_{n+m+i} &\leqslant b_{n+m-i} && \text{for all } i \leq n \ , \\ a_{2n+m+i} &\leqslant b_{m+i} && \text{for all } i \leq n \ ; \end{aligned}$$

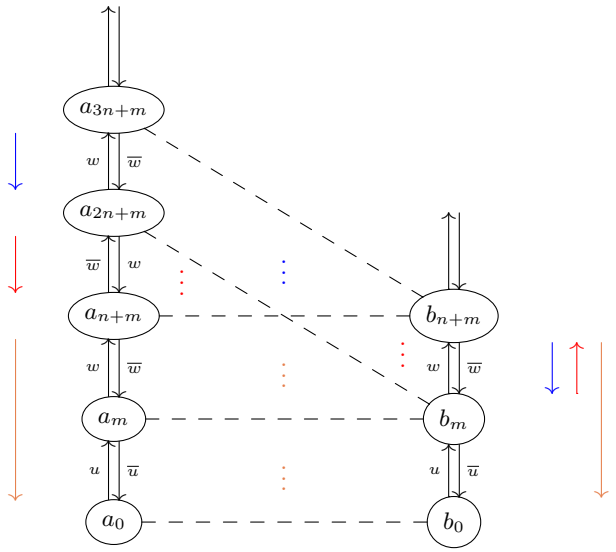


Fig. 4: Automata $\mathscr{G}(uw\overline{w}w)$ and $\mathscr{G}(uw)$, with $|u| = m$ and $|w| = n$

One easily checks that this relation is a simulation, thus establishing in particular that the language recognised by the left-hand side automaton (for $\Gamma(uw\overline{w}w)$) is contained in that of the right-hand side (for $\Gamma(uw)$).

A.5 Proof of result (13)

Recall that $n = |w|$ and $m = |u|$, and that for any $0 \leqslant i \leqslant 2n$, we have:

- $\Gamma_i = \Gamma\left((uw\overline{w}w)|_{m+n+i}\right) = \Gamma\left(uw(\overline{w}w)|_i\right)$
- $v_i \in \Gamma_i$.

We will give here a proof that

$$\forall 0 \leqslant i \leqslant 2n, \exists t_i \in \Gamma(uw) : (\overline{w}w)|_i v_i \rightsquigarrow^\star t_i(\overline{w}w)|_i.$$

As v_i is in $\Gamma\left(uw(\overline{w}w)|_i\right)$, we know that there is some suffix t of $uw(\overline{w}w)|_i$ such that $v_i \rightsquigarrow^\star \overline{t}t$. We will do a case analysis on the size of t:

- if $n + i \leqslant |t|$, then there is a suffix s of u such that $t = sw(\overline{w}w)|_i$, so $(\overline{w}w)|_i v_i \rightsquigarrow^\star (\overline{w}w)|_i\overline{(\overline{w}w)|_i}\overline{w}\ \overline{s}sw(\overline{w}w)|_i$.
  - If $i < n$ then there is a word p such that $\overline{w} = (\overline{w}w)|_i p$ so

$$\begin{aligned}(\overline{w}w)|_i v_i &\rightsquigarrow^\star (\overline{w}w)|_i\overline{(\overline{w}w)|_i}(\overline{w}w)|_i p\overline{s}sw(\overline{w}w)|_i \\ &\rightsquigarrow (\overline{w}w)|_i p\overline{s}sw(\overline{w}w)|_i = \overline{sw}sw(\overline{w}w)|_i.\end{aligned}$$

- Otherwise we can write $(\overline{w}w)|_i = \overline{w}w_1$ and $w = w_1w_2$, so

$$\begin{aligned}(\overline{w}w)|_i v_i &\rightsquigarrow^\star \overline{w}w_1\overline{\overline{w}w_1}\overline{w}\ \bar{s}sw(\overline{w}w)|_i\\ &= \overline{w_2}\ \overline{w}_1w_1\overline{w}_1w\overline{w}\ \bar{s}sw(\overline{w}w)|_i\\ &\rightsquigarrow \overline{w_2}\ \overline{w}_1w\overline{w}\ \bar{s}sw(\overline{w}w)|_i\\ &= \overline{w}w\overline{w}\ \bar{s}sw(\overline{w}w)|_i\\ &\rightsquigarrow \overline{sw}sw(\overline{w}w)|_i.\end{aligned}$$

  As $s \in \text{suffixes}(u)$ we know that $sw \in \text{suffixes}(uw)$, hence $\overline{sw}sw \in \Gamma(uw)$.

– If $i \leqslant |t| < n + i$ then $w = w_1w_2$ and $t = w_2(\overline{w}w)|_i$ so

$$(\overline{w}w)|_i v_i \rightsquigarrow^\star (\overline{w}w)|_i\overline{(\overline{w}w)|_i}\overline{w_2}w_2(\overline{w}w)|_i$$

  - If $i < n$ then there is a word p such that $\overline{w} = (\overline{w}w)|_i p$. As $\overline{w} = \overline{w_2}\ \overline{w_1}$, we can also compare $(\overline{w}w)|_i$ with $\overline{w_2}$:
    * If $(\overline{w}w)|_i = \overline{w_2}w_3$ then

$$\begin{aligned}(\overline{w}w)|_i v_i &\rightsquigarrow^\star \overline{w_2}w_3\overline{w_3}w_2\overline{w_2}w_2(\overline{w}w)|_i\\ &\rightsquigarrow \overline{w_2}w_3\overline{w_3}w_2(\overline{w}w)|_i\\ &= (\overline{w}w)|_i\overline{(\overline{w}w)|_i}(\overline{w}w)|_i\\ &= \overline{\overline{(\overline{w}w)|_i}}\ \overline{(\overline{w}w)|_i}(\overline{w}w)|_i\end{aligned}$$

      And as $\overline{w} = (\overline{w}w)|_i p$, $w = \overline{p}\overline{(\overline{w}w)|_i}$ so $\overline{(\overline{w}w)|_i} \in \text{suffixes}(w) \subseteq \text{suffixes}(uw)$, hence $\overline{\overline{(\overline{w}w)|_i}}\ \overline{(\overline{w}w)|_i} \in \Gamma(uw)$.
    * If on the other hand $\overline{w_2} = (\overline{w}w)|_i w_3$, we have

$$\begin{aligned}(\overline{w}w)|_i v_i &\rightsquigarrow^\star (\overline{w}w)|_i\overline{(\overline{w}w)|_i}(\overline{w}w)|_i w_3\overline{w_3}\overline{(\overline{w}w)|_i}(\overline{w}w)|_i\\ &\rightsquigarrow (\overline{w}w)|_i w_3\overline{w_3}\overline{(\overline{w}w)|_i}(\overline{w}w)|_i\\ &= \overline{w_2}w_2(\overline{w}w)|_i\end{aligned}$$

      $w_2 \in \text{suffixes}(w) \subseteq \text{suffixes}(uw)$ so $\overline{w}_2w_2 \in \Gamma(uw)$.
  - Otherwise we can write $(\overline{w}w)|_i = \overline{w}w_3$ and $w = w_3w_4$, so

$$\begin{aligned}(\overline{w}w)|_i v_i &\rightsquigarrow^\star \overline{w}w_3\overline{w_3}w_3w_4\overline{w_2}w_2(\overline{w}w)|_i\\ &\rightsquigarrow \overline{w}w_3w_4\overline{w_2}w_2(\overline{w}w)|_i\\ &= \overline{w}w\overline{w_2}w_2(\overline{w}w)|_i\\ &= \overline{w}w_1w_2\overline{w_2}w_2(\overline{w}w)|_i\\ &\rightsquigarrow \overline{w}w_1w_2(\overline{w}w)|_i\\ &= \overline{w}w(\overline{w}w)|_i\end{aligned}$$

      And obviously $\overline{w}w \in \Gamma(uw)$.

– If $|t| < i$ then $(\overline{w}w)|_i = st$. In this case we have $(\overline{w}w)|_i v_i \rightsquigarrow^\star st\bar{t}t \rightsquigarrow st = \bar{\epsilon}\epsilon(\overline{w}w)|_i$, and $\epsilon \in \text{suffixes}(uw)$ so $\bar{\epsilon}\epsilon \in \Gamma(uw)$.

In all cases, we have shown that $(\overline{w}w)_i v_i \rightsquigarrow^\star t_i(\overline{w}w)|_i$ with $t_i \in \Gamma(uw)$.

A.6 Proof of the bisimulation between the two closure constructions

Let us be more precise : starting from a non-deterministic automaton $\mathscr{A} = \langle Q, \mathbf{X}, I, Q_f, \Delta\rangle$, its determinised is $\mathcal{D} = \langle \mathcal{P}(Q), \mathbf{X}, I, T, \delta\rangle$ with

$$T = \{P : P \cap Q_f \neq \emptyset\} \text{ and } \delta(P, x) = P \cdot \Delta(x).$$

We can build two automata recognising its closure. The first one, derived from our construction, is

$$\mathscr{A}_1 = \langle \mathcal{P}(Q) \times G, \mathbf{X}, (I, [\epsilon]), T_1, \delta_1\rangle$$

where G is the set of equivalence relations of $\sim_\gamma$, $T_1 \triangleq \{(P, [u]) \mid P \cap Q_f \neq \emptyset\}$ and

$$\delta_1((P, [u]), x) = (P \cdot (\Delta(x) \circ \gamma(ux)), [ux]).$$

The second one, given by the original construction, is

$$\mathscr{A}_2 = \langle \mathcal{P}(M_\mathcal{D}) \times \mathcal{P}(M_\mathcal{D}), \mathbf{X}, (\underline{\epsilon}, \underline{\epsilon}), T_1, \delta_2\rangle$$

where $M_\mathcal{D}$ is the transition monoid of $\mathcal{D}$, a set of endomorphisms of $\mathcal{P}(Q)$ induced by words, $\underline{w} \triangleq \{w_\mathcal{D}\}$ is a singleton containing the interpretation of a word w in $M_\mathcal{D}$, $T_2 \triangleq \{(F, G) \mid \exists q_f \in Q_f, \exists f \in F : q_f \in f(I)\}$, and the transition function is

$$\delta_2((F, G), x) = (F \odot \underline{x} \odot (\underline{x}' \odot G \odot \underline{x})^\star, (\underline{x}' \odot G \odot \underline{x})^\star).$$

($A \odot B \triangleq \{g \circ f \mid f \in A \wedge g \in B\}$.) The fact that the elements of $M_\mathcal{D}$ are semilattice-homomorphisms can be easily checked, as $u_\mathcal{D}(P)$ is the only state of $\mathcal{D}$ (*i.e.* a set of states of $\mathscr{A}$) such that $P \xrightarrow{u}_\mathcal{D} u_\mathcal{D}(P)$. Then is is straightforward that :

$$\begin{aligned} u_\mathcal{D}(P_1 \cup P_2) &= \{q \in Q \mid \exists p \in P_1 \cup P_2 : p \xrightarrow{u}_\mathscr{A} q\} \\ &= \{q \in Q \mid \exists p \in P_1 : p \xrightarrow{u}_\mathscr{A} q\} \cup \{q \in Q \mid \exists p \in P_2 : p \xrightarrow{u}_\mathscr{A} q\} \\ &= u_\mathcal{D}(P_1) \cup u_\mathcal{D}(P_2). \end{aligned}$$

Now, to give the bisimulation we need the following morphism i from $\mathcal{P}(M_\mathcal{D})$ to $\mathcal{P}(Q^2)$ defined by

$$i(F) \triangleq \{(p, q) \mid \exists f \in F : q \in f(\{p\})\}.$$

Note that i is a KA-homomorphism because the elements of the transition monoid of the determinised automaton are semilattice-homomorphisms from

$\mathcal{P}(Q)$ to $\mathcal{P}(Q)$. Let's check that :

$$\begin{aligned}
\epsilon_{\mathcal{D}} &= \mathrm{Id}_{\mathcal{P}(Q)}, \text{ meaning that } i(\underline{\epsilon}) = \mathrm{Id}_Q;\\
i(F_1 \cup F_2) &= \{(p,q) \mid \exists f \in F_1 \cup F_2 : q \in f(\{p\})\}\\
&= \{(p,q) \mid \exists f \in F_1 : q \in f(\{p\})\} \cup \{(p,q) \mid \exists f \in F_2 : q \in f(\{p\})\}\\
&= i(F_1) \cup i(F_2);\\
i(F_1 \odot F_2) &= \{(p,q) \mid \exists f \in F_1 \odot F_2 : q \in f(\{p\})\}\\
&= \{(p,q) \mid \exists f, g \in F_1 \times F_2 : q \in g \circ f(\{p\})\}\\
&= \{(p,q) \mid \exists f \in F_1 : \exists p' \in f(\{p\}) : \exists g \in F_2 : q \in g(\{p'\})\}\\
&\qquad\qquad (g \text{ is a semilattice homomorphism})\\
&= \{(p,q) \mid \exists p' : (p,p') \in i(F_1) \wedge (p',q) \in i(F_2)\}\\
&= i(F_1) \circ i(F_2)
\end{aligned}$$

For the * operation, recall that

$$\forall F \in \mathcal{P}(M_{\mathcal{D}}), \exists n_1(F) \in \mathbb{N} : \forall m \leqslant n_1(F), F^\star = (F \cup \underline{\epsilon})^m;$$

and that

$$\forall R \in \mathcal{P}(Q)^2, \exists n_2(R) \in \mathbb{N} : \forall m \leqslant n_2(R), R^\star = (R \cup \mathrm{Id}_Q)^m.$$

Then, if we write $m = \max(n_1(F), n_2(u_{\mathcal{D}}(F)))$,

$$\begin{aligned}
i(F^\star) &= i((F \cup \underline{\epsilon})^m)\\
&= (i(F) \cup i(\underline{\epsilon}))^m\\
&= (i(F))^\star
\end{aligned}$$

We can also check that, for any $x \in \mathbf{X}$:

$$\begin{aligned}
i(\underline{x}) &= \{(p,q) \mid q \in x_{\mathcal{D}}(\{p\})\}\\
&= \{(p,q) \mid q \in \delta(\{p\}, x)\}\\
&= \{(p,q) \mid p \xrightarrow{x}_{\mathscr{A}} q\}\\
&= \Delta(x).
\end{aligned}$$

The bisimulation $\sim$ can thus be expressed :

$$\sim \triangleq \{((Q,[u]),\ (F,G)) \mid Q = I \cdot i(F) \text{ and } \gamma(u) = i(G)\}$$

where $(Q,[u])$ is a state of $\mathscr{A}_1$ and (F,G) is a state of $\mathscr{A}_2$. We will now show prove that it is indeed a bisimulation.

1. We need the inital states to be related. This is obvious as $\epsilon_{\mathcal{D}} = \mathrm{Id}_{\mathcal{P}(Q)}$, meaning that $i(\underline{\epsilon}) = \mathrm{Id}_Q$. Furthermore, $\gamma(\epsilon) = \mathrm{Id}_Q$ and $I = I \cdot \mathrm{Id}_Q$. That means $(I,[\epsilon]) \sim (\underline{\epsilon}, \underline{\epsilon})$.

2. For the final states, it isn't much more complicated :

$$\begin{aligned}(F,G) \in T_2 &\Leftrightarrow \exists q_f \in Q_f : \exists f \in F : q_f \in f(I) \\ &\Leftrightarrow \exists q_f \in Q_f : q_f \in I \cdot i(F) \\ &\Leftrightarrow I \cdot i(F) \cap Q_f \neq \emptyset \\ &\Leftrightarrow (I \cdot i(F), i(G)) \in T_1.\end{aligned}$$

3. What remains to be shown is that this relation is stable under transitions from both sides. Suppose that $(Q,[u]) \sim (F,G)$, and consider $x \in \mathbf{X}$. After reading x we get in $\mathscr{A}_2$ $(F \odot \underline{x} \odot G', G')$, with $G' = (\underline{x'} \odot G \odot \underline{x})^\star$, and in $\mathscr{A}_1$ $(Q \cdot (\Delta(x) \circ \gamma(ux)), [ux])$. We will prove that they are still related in two steps, first by looking at the second component, and then dealing with the first one.
   (a) We know that $\gamma(u) = i(G)$, and that $i(\underline{x}) = \Delta(x)$.

$$\begin{aligned}\gamma(ux) &= (\Delta(x') \circ \gamma(u) \circ \Delta(x))^\star \\ &= (i(\underline{x'}) \circ i(G) \circ i(\underline{x}))^\star \\ &= i(G') \qquad\qquad (i \text{ is a morphism})\end{aligned}$$

   (b) Now the first component comes quite easily :

$$\begin{aligned}Q \cdot (\Delta(x) \circ \gamma(ux)) &= (I \cdot i(F)) \cdot (i(\underline{x}) \circ i(G')) \\ &= I \cdot (i(F) \circ i(\underline{x}) \circ i(G')) \\ &= I \cdot i(F \odot \underline{x} \odot G').\end{aligned}$$