Theoretical Informatics and Applications Will be set by the publisher
Informatique Théorique et Applications

EFFICIENCY OF AUTOMATA IN SEMI-COMMUTATION VERIFICATION TECHNIQUES

Gérard Cécé[1], Pierre-Cyrille Héam[1] and Yann Mainier[1]

Abstract. Computing the image of a regular language by the transitive closure of a relation is a central question in Regular Model Checking. In a recent paper Bouajjani, Muscholl and Touili [7] proved that the class of regular languages L – called APC – of the form $\cup_j L_{0,j} L_{1,j} L_{2,j} \dots L_{k_j,j}$, where the union is finite and each $L_{i,j}$ is either a single symbol or a language of the form B^* with B a subset of the alphabet, is closed under all semi-commutation relations R. Moreover a recursive algorithm on the regular expressions was given to compute $R^*(L)$. This paper provides a new approach, based on automata, for the same problem. Our approach produces a simpler and more efficient algorithm which furthermore works for a larger class of regular languages closed under union, intersection, semi-commutation relations and conjugacy. The existence of this new class, Pol$\mathcal{C}$, answers the open question proposed in the paper of Bouajjani and al.

1991 Mathematics Subject Classification. 68N30.

1. Introduction

A semi-commutation relation R allows to express rewriting of words such as $xaby \to xbay$, provided $(a,b) \in R$. Semi-commutations are used in several domains, for instance as a model of parallelism in Mazurkiewicz trace theory [11], in partial order reduction techniques [14], or to express exchange of a piece of information between neighbouring processes in linear or ring networks. In *regular model checking* [3, 5, 6], a key point is the computation of the image of a regular language by the transitive closure of a relation. However, such computation, in

Keywords and phrases: regular model checking, verification, parametric systems, semi-commutations

[1] LIFC ; CNRS FRE 2661 – Projet INRIA-CASSIS ; Université de Franche-Comté ; 16 route de Gray ; 25030 Besancon Cedex-FRANCE. e-mail: `{Gerard.Cece, Pierre-Cyrille.Heam, Yann.Mainier}@univ-fcomte.fr`

the case of semi-commutation relations, may lead to non regular languages. The classical example is the following one: let $L = (ab)^*$ and $R = \{(a,b)\}$. Then, $R^*(L) \cap b^*a^* = \{b^n a^n \mid n \in \mathbb{N}\}$. Therefore $R^*(L)$ is not regular. In [7], Bouajjani, Muscholl and Touili searched for a class of regular languages closed under all semi-commutation relations. They defined the class APC (finite union of products of languages of the form a_0 or $\{a_1, a_2, \ldots, a_n\}^*$ with a_i's single symbols) and gave an algorithm to compute $R^*(L)$ for any APC L and any semi-commutation relation R. Unfortunately, their algorithm is based on a series of mutually recursive transformations on the regular expressions defining the APC. During the computation, at each intermediate stage, the size of the APC is multiplied, which induces a final result of exponential size. Moreover, as they have proved that the inclusion problem for APC is PSPACE complete, there is no practical way of simplifying the intermediate APC during computation.

In this paper, we use a completely different approach. Instead of working on regular expressions, we use automata. This results in a simpler and, as confirmed by some experiments, much more efficient technique. In addition to leading to a more compact representation, using automata also makes the use of other techniques of regular model checking easier as these techniques are mainly based on automata.

As advocated by [7], APC is an interesting subclass of regular languages; several verification problems (sliding window protocols, parameterized mutual exclusion protocols, etc.) can be modeled with them. An open question was the existence of a larger class than APC, satisfying the same good closure properties. By investigating polynomial closure of varieties of regular languages, we give a positive answer to this question with the class Pol$\mathcal{C}$ (polynomial closure of commutative regular languages) composed of finite unions of languages of the form $L_0 a_0 L_1 a_1 \ldots a_k L_k$ where the a_i's are single symbols and the L_i's are commutative regular languages, that is languages that satisfy: $\forall a, b \in A\, \forall x, y \in A^* (xaby \in L_i \implies xbay \in L_i)$, with A an alphabet. This class allows to describe languages such as: $L_1 d L_2$, with $L_1 = \{u \in \{a,b\}^* \mid |u|_b$ is even and $|u|_a$ is even$\}$ and $L_2 = \{u \in \{a,d\}^* \mid |u|$ is odd$\}$.

Related Work

Regular model checking [3, 5, 6] is an approach to verify infinite state systems. One represents, symbolically, sets of states by regular languages and one develops *meta-transitions* which can compute, in one step, infinite sets of successors. This amounts to compute $R^*(L)$ for a given regular language L and a given relation R representing a subset of the transition relation T of the system. The transition relation T can be decomposed into several sub relations R_i (of semi-commutation or something else), each of them implying their ad-hoc techniques of computation. As most of the developed techniques are based on automata, it is more efficient and consistent to use automata during the whole computation. This last remark is another plus for our technique compared to that of [7].

Polynomial closure of varieties of regular languages is an operation widely studied in the literature (see *for example* [8, 9, 24, 28]). In this paper we consider the languages of level 3/2 in the Straubing-Thérien hierarchy [26,29] which represents the current border for decidability problems and whose structure makes them suitable for verification of certain systems [1,2,7,30]. Decomposable languages is a class of regular languages used for the simulation of process algebra [21]. It was conjectured in [25] that this class was exactly Pol$\mathcal{C}$. However this conjecture has just been invalidated in [15]. Finally, looking for the maximal (positive) variety closed under an operator is widely studied in the literature. One can cite the result for the shuffle operator for varieties [13,22] and for positive varieties [16].

Layout of the Paper

In Sect. 2 we recall the basic notions and notations. Then in Sect. 3 we give the main result of the paper: the key construction which allows the use of automata in computation of the transitive closure of ad hoc regular languages by a semi-commutation relation. In Sect. 4, we compare, in theory and in practice, the two approaches, the one manipulating regular expressions [7] and ours using automata. Then we extend, in Sect. 5 the class of regular languages for which this computation is feasible. Finally, we conclude in Sect. 6.

2. Background and Notations

We assume that the reader has a basic background in finite automata theory. For more information on automata the reader is referred to [4,18].

Recall that a finite automaton is a 5-tuple $\mathcal{A} = (Q, A, E, I, F)$ where Q is a finite set of states, A is the alphabet, $E \subseteq Q \times A \times Q$ is the set of transitions, $I \subseteq Q$ is the set of initial states and $F \subseteq Q$ is the set of final states. If $\mathcal{A}$ is a finite automaton, $L(\mathcal{A})$ denotes the language accepted by $\mathcal{A}$. If $C \subseteq Q$ and $D \subseteq Q$, $\mathcal{A}_{C,D}$ denotes the automaton (Q, A, E, C, D). Moreover, for all $p \in Q$, $p \cdot a = \{q \in Q \mid (p, a, q) \in E\}$. If $p \cdot a = \{q\}$ is a singleton, we also write $p \cdot a = q$. In this paper, minimal automata are deterministic but not necessary complete.

If $u \in A^*$, $\mathrm{Conj}(u) = \{vw \mid wv = u\}$ denotes the set of its conjugated words. This notion is extended to languages as follows

$$\mathrm{Conj}(L) = \bigcup_{u \in L} \mathrm{Conj}(u).$$

If u is a finite word, $\alpha(u)$ denotes the set of letters occurring in u. This notion is extended to languages: $\alpha(L) = \bigcup_{u \in L} \alpha(u)$.

A semi-commutation R is a relation on A which does not contain the identity. Given a finite word u on A, we denote by $R(u)$ the language $\{xbay \mid x, y \in A^*, (a, b) \in R \text{ and } xaby = u\}$ and by $R^*(u)$ the language $\{u\} \cup \cup_{k \geq 1} R^k(u)$. These notions are extended to languages by

$$R(L) = \bigcup_{u \in L} R(u) \quad \text{and} \quad R^*(L) = \bigcup_{u \in L} R^*(u).$$

Given two words u and v in A^*, the shuffle of u and v, denoted $u \sqcup\!\sqcup v$, is the set of words of the form $u_1v_1 \dots u_nv_n$ such that $u = u_1 \dots u_n$ and $v = v_1 \dots v_n$. The R-shuffle of u and v, denoted $u \sqcup\!\sqcup_R v$ is similar but with the added condition: $\alpha(u_i) \times \alpha(v_j) \subseteq R$ for all $j < i$. The intuition is as follows. To construct the set $u \sqcup\!\sqcup_R v$, one first starts from uv, then one adds all the words obtained by the commutation of two successive letters ab in an already added word and such that a belongs to u, b belongs to v and (a, b) belongs to R.

The R-shuffle operation is extended to languages L and K of A^* by

$$L \sqcup\!\sqcup_R K = \bigcup_{u \in L, v \in K} u \sqcup\!\sqcup_R v.$$

As stated in the following proposition [12], it is important to be able to compute the R-shuffle of two languages since this is the key which allows the computation of the transitive closure of a product of R-closed languages.

Proposition 2.1 ([12]). *Let $L_1, \dots, L_n$ be n R-closed sets, i.e. such that for every i, $1 \leq i \leq n$, $L_i = R^*(L_i)$, then we have:*

$$R^*(L_1L_2 \dots L_n) = L_1 \sqcup\!\sqcup_R (L_2 \sqcup\!\sqcup_R (\cdots (L_{n-1} \sqcup\!\sqcup_R L_n) \cdots))$$

Now, let us recall the formal definition of the class APC given in [7].

Definition 2.2 ([7]). Let A be a finite alphabet. An *atomic expression* over A is either a letter a of A or a star expression $\{a_1, \dots, a_n\}^*$, where $\{a_1, \dots, a_n\} \subseteq A$. A *product* p over A^* is a concatenation $e_1 \dots e_n$ of atomic expressions $e_1, \dots, e_n$ over A. An *Alphabetic Pattern Constraint* (APC) over A^* is an expression of the form $\cup_{i \leq n} p_i$, where p_i are products over A^*.

Since an APC language L is a finite union of products of trivially R-closed languages (single symbols or star expressions of subsets of the alphabet), computing $R^*(L)$ is reduced to the computation of the R-shuffle of languages. Since [7] provides an algorithm to compute the R-shuffle of two APC's, which is also an APC, $R^*(L)$ is computable. In the next section we give an automata approach for computing the R-shuffle of two regular languages.

3. R-shuffle Product and Finite Automata

We present our first main result: how to compute the R-shuffle automaton of two regular languages given by finite automata. The method used is based on the classical one for computing the shuffle of two regular languages. That is to say, construct a new automaton whose transitions are either from the first or from the second automaton. This implies that a state of that new automaton is a couple of states of the two given automata. Now we have to guarantee that the condition $\alpha(u_i) \times \alpha(v_j) \subseteq R$ for all $j < i$ is also fulfilled. To do this, it suffices to memorize the set of letters read by the second automaton (recognizing v) and to guarantee that we only read letters in the first automaton (recognizing u) which commute with all the memorized letters.

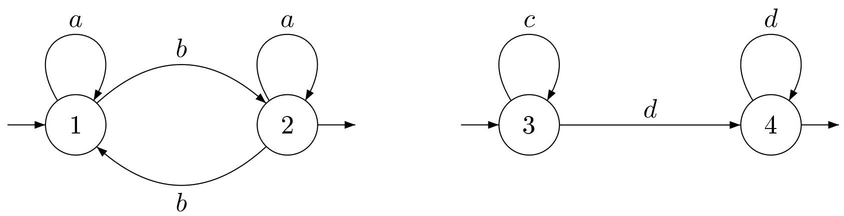
Proposition 3.1. *Let $\mathcal{A}_1 = (Q_1, A, E_1, I_1, F_1)$ and $\mathcal{A}_2 = (Q_2, A, E_2, I_2, F_2)$ be two finite automata and R a semi-commutation relation over A. If $B \subseteq \alpha(L(\mathcal{A}_2))$, we denote by $\overleftarrow{B}$ the set $\{a \in \alpha(L(\mathcal{A}_1)) \mid \{a\} \times B \subseteq R\}$ and by $\overleftrightarrow{B}$ the set $\{b \in \alpha(L(\mathcal{A}_2)) \mid \overleftarrow{B} \times \{b\} \subseteq R\}$.*

The finite automaton $\mathcal{A} = (Q, A, E, I, F)$ defined by:

- $Q = Q_1 \times Q_2 \times \mathcal{P}(A)$,
- $I = \{(p_1, p_2, \overleftrightarrow{\emptyset}) \mid p_1 \in I_1, p_2 \in I_2\}$,
- $F = \{(p_1, p_2, B) \mid p_1 \in F_1, p_2 \in F_2, B \subseteq A\}$,
- $E = G_1 \cup G_2$, *with*
  $G_1 = \{((p_1, p_2, B), a, (q_1, p_2, B)) \mid p_1 \in Q_1, p_2 \in Q_2, q_1 \in p_1 \cdot a, B \subseteq A$ *and* $a \in \overleftarrow{B}\}$ *and*
  $G_2 = \{((p_1, p_2, B), b, (p_1, q_2, \overleftrightarrow{B \cup \{b\}})) \mid p_1 \in Q_1, p_2 \in Q_2, q_2 \in p_2 \cdot b, B \subseteq A\}$.

is denoted $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ and accepts $L(\mathcal{A}_1) \sqcup\!\sqcup_R L(\mathcal{A}_2)$.

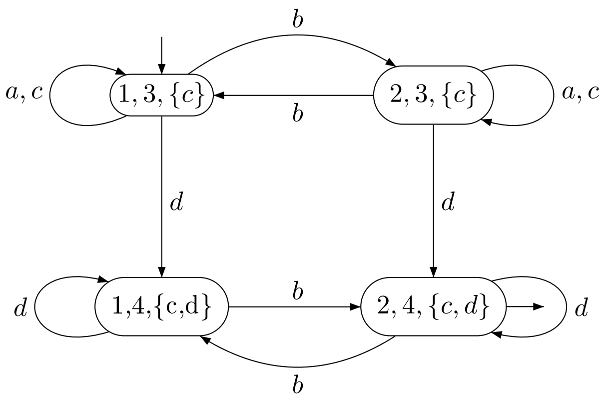
Example 3.2. Consider the following finite automata $\mathcal{A}_1$ and $\mathcal{A}_2$:



and the semi-commutation relation $R = \{(b, c), (b, d), (a, c)\}$. One has:

B	$\overleftarrow{B}$	$\overleftrightarrow{B}$
$\emptyset$	$\{a, b\}$	$\{c\}$
$\{c\}$	$\{a, b\}$	$\{c\}$
$\{d\}$	$\{b\}$	$\{c, d\}$
$\{c, d\}$	$\{b\}$	$\{c, d\}$

Then, $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ is the following automaton (we only represent accessible states):



Let us remark that if in Proposition 3.1 we replace G_2 by the set of transitions $G'_2 = \{((p_1, p_2, B), b, (p_1, q_2, B \cup \{b\})) \mid p_1 \in Q_1, p_2 \in Q_2, q_2 \in p_2 \cdot b, B \subseteq A\}$ and I by $I' = \{(p_1, p_2, \emptyset) \mid p_1 \in I_1, p_2 \in I_2\}$, we also obtain a finite automaton recognizing $L(\mathcal{A}_1) \sqcup\!\sqcup_R L(\mathcal{A}_2)$ and easier to construct but with a larger number of states. To get the intuition, let us recall that the role of B is to memorise the union of the $\alpha(v_j)$ appearing in the definition of the R-shuffle. But indeed, its effect is to constraint the transitions of $\mathcal{A}_1$ to consider at a given step (see definition of G_1). So the real information is $\overleftarrow{B}$. And as we will see, $\overleftarrow{B} = \overleftarrow{\overleftrightarrow{B}}$ and $B \subseteq \overleftrightarrow{B}$. Thus it is an optimization to use $\overleftrightarrow{B}$ instead of B.

Now we prove Proposition 3.1.

Proof. First we prove some technical properties of the functions $\overleftarrow{\cdot}$ and $\overleftrightarrow{\cdot}$.

(i) For all $B \subseteq \alpha(L(\mathcal{A}_2))$, $B \subseteq \overleftrightarrow{B}$: let $b \in B$. By definition of $\overleftarrow{B}$, for each $a \in \overleftarrow{B}$, $(a, b) \in R$. Thus $b \in \overleftrightarrow{B}$.

(ii) For all $B \subseteq \alpha(L(\mathcal{A}_2))$, $\overleftarrow{\overleftrightarrow{B}} = \overleftarrow{B}$ and $\overleftrightarrow{\overleftrightarrow{B}} = \overleftrightarrow{B}$: by (i), $\overleftarrow{\overleftrightarrow{B}} \subseteq \overleftarrow{B}$. Conversely, by definition of $\overleftrightarrow{B}$, $\overleftarrow{B} \times \overleftrightarrow{B} \subseteq R$. Consequently, $\overleftarrow{B} \subseteq \overleftarrow{\overleftrightarrow{B}}$. It follows that $\overleftarrow{\overleftrightarrow{B}} = \overleftarrow{B}$ and thus $\overleftrightarrow{\overleftrightarrow{B}} = \overleftrightarrow{B}$.

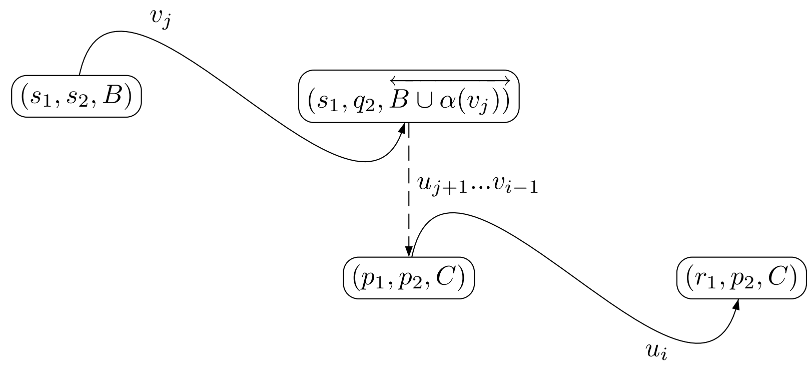
(iii) For all $b \in \alpha(L(\mathcal{A}_2))$, $\overleftrightarrow{\overleftrightarrow{B} \cup \{b\}} = \overleftrightarrow{B \cup \{b\}}$: by definition, a letter a belongs to $\overleftarrow{B \cup \{b\}}$ if and only if $a \in \overleftarrow{B}$ and $(a, b) \in R$. By (ii), $\overleftarrow{\overleftrightarrow{B}} = \overleftarrow{B}$. It follows that $a \in \overleftarrow{B \cup \{b\}}$ if and only if $a \in \overleftarrow{\overleftrightarrow{B}}$ and $(a, b) \in R$. Consequently, $\overleftarrow{\overleftrightarrow{B} \cup \{b\}} = \overleftarrow{B \cup \{b\}}$, and thus $\overleftrightarrow{\overleftrightarrow{B} \cup \{b\}} = \overleftrightarrow{B \cup \{b\}}$.

(iv) For all $B \subseteq C$, $\overleftrightarrow{B} \subseteq \overleftrightarrow{C}$. Direct consequence of the definitions: $B \subseteq C$ implies $\overleftarrow{C} \subseteq \overleftarrow{B}$, which implies $\overleftrightarrow{B} \subseteq \overleftrightarrow{C}$.

Now we prove that $L(\mathcal{A}) \subseteq L(\mathcal{A}_1) \sqcup\!\sqcup_R L(\mathcal{A}_2)$. Let $w \in L(\mathcal{A})$. By definition, there exists an accepting path m in $\mathcal{A}$ labelled by w. This path m can be decomposed into:

$$m = m_1 m_2 m_3 \dots m_k$$

such that k is an even integer, some m_i may be empty, m_{2i+1} $(0 \leq i \leq (k-1)/2)$ only uses transitions of G_1 and m_{2i} $(1 \leq i \leq k/2)$ only uses transitions of G_2. Now, let us denote by u_{i+1} the label of m_{2i+1} and v_i the label of m_{2i}. By construction, $w = u_1 v_1 u_2 \dots u_r v_r$ with $r = k/2$, $u_1 \dots u_r \in L(\mathcal{A}_1)$ and $v_1 \dots v_r \in L(\mathcal{A}_2)$. We claim that for all $1 \leq j < i \leq r$, $\alpha(u_i) \times \alpha(v_j) \subseteq R$. Indeed, let $1 \leq j < i \leq r$. Assume that u_i or v_j is empty. Then $\alpha(u_i) \times \alpha(v_j) = \emptyset \subseteq R$. Assume now that u_i and v_j are both non-empty. Let (s_1, s_2, B) be the first state of m_{2j}. Since m_{2j} only uses transitions of G_2 and by (iii), the last state of m_{2j} is of the form $(s_1, q_2, \overleftrightarrow{B \cup \alpha(v_j)})$. Let (p_1, p_2, C) be the first state of m_{2i+1}. Since m_{2i+1} only uses transitions of G_1, its last state is of the form (r_1, p_2, C).



By construction and by (iii), $C = \overleftrightarrow{B \cup \alpha(v_j v_{j+1} \dots v_{i-1})}$. By (iv), it follows that $\overleftrightarrow{B \cup \alpha(v_j)} \subseteq C$. Moreover, since the path m_{2i+1} only uses transitions of G_1, each letter $a \in \alpha(u_i)$ has to satisfy $\{a\} \times C \subseteq R$. It follows that $\alpha(u_i) \times \alpha(v_j) \subseteq R$, proving the claim. Consequently, $w \in L(\mathcal{A}_1) \sqcup\!\sqcup_R L(\mathcal{A}_2)$.

Finally we prove that $L(\mathcal{A}_1) \sqcup\!\sqcup_R L(\mathcal{A}_2) \subseteq L(\mathcal{A})$. Let z be in $L(\mathcal{A}_1) \sqcup\!\sqcup_R L(\mathcal{A}_2)$. By definition there exist $x_1, y_1, \dots, x_n, y_n$, such that $x_1 x_2 \dots x_n \in L(\mathcal{A}_1)$, $y_1 y_2 \dots y_n \in L(\mathcal{A}_2)$ for all $1 \leq i \leq n$ and for all $1 \leq j < i \leq n$, $\alpha(x_i) \times \alpha(y_j) \subseteq R$. Since $x_1 x_2 \dots x_n \in L(\mathcal{A}_1)$, there exist $p_0, p_1, \dots, p_n \in Q_1$ such that

- $p_0 \in I_1$,
- $p_n \in F_1$,
- for all $i \in \{1, \dots, n\}$, there exists a path in $\mathcal{A}_1$ from p_{i-1} to p_i labelled by x_i.

Since $y_1 y_2 \dots y_n \in L(\mathcal{A}_2)$, there exist $q_0, q_1, \dots, q_n \in Q_2$ such that

- $q_0 \in I_2$,
- $q_n \in F_2$,
- for all $i \in \{1, \dots, n\}$, there exists a path in $\mathcal{A}_2$ from q_{i-1} to q_i labelled by y_i.

For all $i \in \{1, \ldots, n\}$, let us denote by t_i the word $y_1 \ldots y_i$. Moreover, let $t_0 = \varepsilon$. We claim that for all $i \in \{1, \ldots, n\}$, there exists a path in $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ labelled by x_i from $(p_{i-1}, q_{i-1}, \overleftrightarrow{\alpha(t_{i-1})})$ to $(p_i, q_{i-1}, \overleftrightarrow{\alpha(t_{i-1})})$ and a path in $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ labelled by y_i from $(p_i, q_{i-1}, \overleftrightarrow{\alpha(t_{i-1})})$ to $(p_i, q_i, \overleftrightarrow{\alpha(t_i)})$.

$$\begin{tikzcd} (p_{i-1}, q_{i-1}, \overleftrightarrow{\alpha(t_{i-1})}) \arrow[r, "x_i"] & (p_i, q_{i-1}, \overleftrightarrow{\alpha(t_{i-1})}) \arrow[r, "y_i"] & (p_i, q_i, \overleftrightarrow{\alpha(t_i)}) \end{tikzcd}$$

Let i be in $\{1, \ldots, n\}$. Since for all j such that $1 \leq j < i$, $\alpha(x_i) \times \alpha(y_j) \subseteq R$, one has $\alpha(x_i) \times \overleftrightarrow{\alpha(t_{i-1})} \subseteq R$. Thus, by definition of p_{i-1}, p_i, q_{i-1} and by construction of $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$, there exists a path in $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ labelled by x_i from $(p_{i-1}, q_{i-1}, \overleftrightarrow{\alpha(t_{i-1})})$ to $(p_i, q_{i-1}, \overleftrightarrow{\alpha(t_{i-1})})$. Furthermore, by definition of q_{i-1}, p_i, q_i and by construction of $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$, there exists a path in $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ labelled by y_i from $(p_i, q_{i-1}, \overleftrightarrow{\alpha(t_{i-1})})$ to $(p_i, q_i, \overleftrightarrow{\alpha(t_i)})$, proving the claim.
Consequently, there exists a path in $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ from $(p_0, q_0, \overleftrightarrow{\emptyset})$ (an initial state) to $(p_n, q_n, \overleftrightarrow{\alpha(y_1 \ldots y_n)})$ (a final state) and labelled by z. It follows that $L(\mathcal{A}_1) \sqcup\!\sqcup_R L(\mathcal{A}_2) \subseteq L(\mathcal{A})$. $\square$

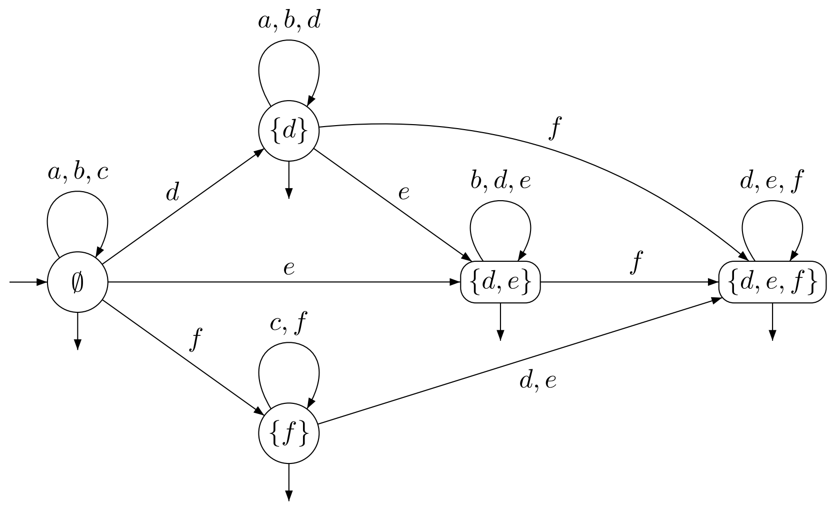
Remark that the automaton $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ may be non-deterministic, even when $\mathcal{A}_1$ and $\mathcal{A}_2$ are both deterministic.

4. Application to APC

Let us first start by an example. Let $C = \{a, b, c\}$, $D = \{d, e, f\}$ and $R = \{(a, d), (c, f), (b, d), (b, e)\}$. Using Proposition 3.1, one has

B	$\overleftarrow{B}$	$\overleftrightarrow{B}$
$\emptyset$	$\{a, b, c\}$	$\emptyset$
$\{d\}$	$\{a, b\}$	$\{d\}$
$\{e\}$	$\{b\}$	$\{d, e\}$
$\{f\}$	$\{c\}$	$\{f\}$
$\{d, e\}$	$\{b\}$	$\{d, e\}$
$\{e, f\}$	$\emptyset$	$\{d, e, f\}$
$\{d, f\}$	$\emptyset$	$\{d, e, f\}$
$\{d, e, f\}$	$\emptyset$	$\{d, e, f\}$

Thus, the language $C^* \sqcup\!\sqcup_R D^*$, which is indeed $R^*(C^* D^*)$ (cf. end of Sect. 2) is given by the following automaton:



Using [7, Example 2], one obtains that $R^*(C^*D^*) = \{a, b, c\}^*\{c, f\}^*\ \{d, e, f\}^* \cup \{a, b, c\}^*\{a, b, d\}^*\{b, d, e\}^*\{d, e, f\}^*$ which is precisely the language of the automaton given above. The compactness of automata is already revealed in this example by its sharing of the states representing respectively the expressions $\{a, b, c\}^*$ and $\{d, e, f\}^*$. Indeed, as shown next, our automaton is the minimal one.

Theorem 4.1. *Let A be an alphabet, R a semi-commutation relation on A, and C and D subsets of A such that $C \cap D = \emptyset$. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be the trivial minimal automata recognizing C^* and D^*, respectively. Then $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ is the minimal automaton recognizing $L(\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2)$.*

Proof. $\mathcal{A}_1$ and $\mathcal{A}_2$ are respectively made of a single state which is both initial and final, with loops on that state labelled by their respective letters. Therefore, in what follows, we identify states of $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ with their third component. By the definitions of G_1 and G_2 in Proposition 3.1, $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ is deterministic since $C \cap D = \emptyset$. Now, consider two different states $\overleftrightarrow{B_1}$ and $\overleftrightarrow{B_2}$ of $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ (recall that we identify states with their third component). Then, $\overset{\leftrightarrows}{B_1} \neq \overset{\leftrightarrows}{B_2}$ (by contradiction and with the help of (ii) in the proof of Proposition 3.1). This implies the existence of $a \in \overset{\leftrightarrows}{B_1}$ such that $a \notin \overset{\leftrightarrows}{B_2}$ (or conversely). By definition, this implies that $(\overleftrightarrow{B_1}, a, \overleftrightarrow{B_1})$ is a transition of $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ and this also implies the inexistence in $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ of a transition from $\overleftrightarrow{B_2}$ and labelled by a. Since the respective single state of $\mathcal{A}_1$ and $\mathcal{A}_2$ is final, all reachable states of $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ are final. All of this implies that $\overleftrightarrow{B_1}$ and $\overleftrightarrow{B_2}$ are distinguishable states and thus $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ is minimal [18]. □

In what follows, we compare our approach using automata with that of [7] using regular expressions.

Definition 4.2 ([27]). A finite automaton $\mathcal{A} = (Q, A, E, I, F)$ is called partially ordered if there exists a partial order $\leq$ on Q such that for every transition $(p, a, q) \in E$, $p \leq q$.

It is well known – and obvious – that partially ordered finite automata (POF automata for short) have the same expressivity than APC expressions. One can easily check that if $\mathcal{A}_1$ and $\mathcal{A}_2$ are POF automata, then $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ is a POF automaton too. Consequently, computing the semi-commutation closure of a language given by a POF automaton with our algorithm returns a partially ordered finite automaton. Therefore, with a simple recurrence using Proposition 2.1 we obtain a new proof of the stability of APC under semi-commutation closures.

One can wonder whether our algorithm reduces to encoding an APC expression into a finite partially ordered automaton and to apply the algorithm of [7] on it while merging equivalent states. The answer is no since it was proved in [7] that merging equivalent states in a partially ordered automaton is PSPACE-complete. So this method would be totally inefficient.

Using a regular expression, in our case an APC, may be useful for specifying a property that one closes by a semi-commutation relation. However, even in this case, deciding usual questions like inclusion and membership are more easily computed with automata. Furthermore, a POF-automaton equivalent to an APC expression can be easily computed in linear time and space [19]).

4.1. Theoretical complexity

Following [7], let us call an *atomic expression* a single symbol or a language of the form B^*, with B a subset of the alphabet, and a *product* a finite concatenation of atomic expressions. The *length* of a product is the number of atomic expression composing that product. The size of an APC is the total number of atomic expressions in the products composing that APC.

Let R be a semi-commutation relation over an alphabet A and p be a product over A. Then, from [7], $R^*(p)$ is an APC of size at most $2^{\mathcal{O}(|A|(\delta_R+1)^n)}$ with

$$\delta_R = \max_{a \in A}\{|Y| \subseteq A \mid \{a\} \times Y \subseteq R\}$$

Given two automata $\mathcal{A}_1 = (Q_1, A, E_1, I_1, F_1)$ and $\mathcal{A}_2 = (Q_2, A, E_2, I_2, F_2)$, the number of states of $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ is $\mathcal{O}(2^{|A|}|Q_1||Q_2|)$ and, hence, its size, *i.e.* the number of its states and the number of its transitions, is $\mathcal{O}((2^{|A|}|Q_1||Q_2|)^2)$. A language that contains only a single letter is trivially represented by an automaton of size 2 and a language of the form B^*, with B a subset of the alphabet, is also trivially represented by an automaton of size 1. Therefore, by Proposition 2.1, the numbre of states of the automaton we compute to represents $R^*(p)$ is at most $\mathcal{O}(2^{((|A|+1)n})$. Then, its size is at most $\mathcal{O}(2^{((|A|+1)2n})$ which is better than $2^{\mathcal{O}(|A|(\delta_R+1)^n)}$

Beside these theoretical considerations we give in what follows a pragmatic comparison of the two approaches.

4.2. Experimental Tests

In order to compare both techniques, the one of [7] and ours, we did several tests on randomly chosen products and relations. As criterion of comparison, we chose the size of the results: number of atomic expressions (a letter or B^* with B a subset of the alphabet) for APC's and number of states and transitions for automata. Development was achieved using the functional language Objective Caml [20].

As their effect on the algorithms are very different, we used as inputs, two kinds of products:

type 1:: $B_0^* a_1 B_2^* \dots a_{n-1} B_n^*$
type 2:: $B_0^* B_1^* \dots B_n^*$

Our procedure of comparison was as follows. For each test, we set a kind of product, a size n of the product, a size $|A|$ of the alphabet, and a size $|R|$ of the semi-commutation relation. With these given limits, we randomly generated a product and a semi-commutation relation. After that, we executed the algorithm of [7], then our algorithm on the equivalent automaton of the same product. We then measured the size of the two results. Tables 1 and 2 give a summary of the tests, each result is in fact an average of 15 tests.

Table 1. Comparison of techniques with respect to n with $|A| = 10$ and $|R| = 5$

Product size		2	3	5	7
APC	type 1	10	418	48361	897004
automata	type 1	28	82	333	836
APC	type 2	-	15	252	6402
automata	type 2	-	50	206	591

Table 2. Comparison of techniques with respect to $|R|$ with $|A| = 10$ and $n = 7$

Relation size		3	5	7	9
APC	type 1	785597	1162952	286499	4213859
automata	type 1	578	828	1031	1522
APC	type 2	7540	15153	16965	29730
automata	type 2	502	622	830	936

All of these tests were achieved in less than one or two minutes on an 1.3 GHz Athlon with 1 GB of memory. Processes implementing our technique used less than 4 MB of memory while the amount of memory of those corresponding to [7] increased more rapidly according to the size of the inputs (more than 800 MB for some tests in the right-hand columns of Tables 1 and 2)

We also applied our technique to a language of type 1 with $n = 40$, $|A| = 10$ and $|R| = 10$. The size of the generated automaton was about 450000 and computation takes 42 hours and 128 MB were used by the process. This last kind of test was not feasible with the technique of [7].

5. Permutation Rewriting and Polynomial Closure of Commutative Regular Languages

In this section we present our second main result: the extension of [7] to a larger class of regular languages. For a general reference on varieties of formal languages see [23].

A *class* of languages $\mathcal{V}$ is an application which associates to each finite alphabet A a set of regular languages of A^* denoted by $A^*\mathcal{V}$. A class of languages $\mathcal{V}$ is said to be closed under semi-commutation if for any finite alphabet A, any semi-commutation relation over A and any language in $L \in A^*\mathcal{V}$, $R^*(L) \in A^*\mathcal{V}$.

A *positive variety* of languages $\mathcal{V}$ is a class of languages such that:

(1) $A^*\mathcal{V}$ is closed under finite union and finite intersection.
(2) If φ is a monoid morphism from A^* into B^*, and if $L \in B^*\mathcal{V}$, then $\varphi^{-1}(L) \in A^*\mathcal{V}$.
(3) If $L \in A^*\mathcal{V}$ and if $a \in A$, then $a^{-1}L$ and La^{-1} are in $A^*\mathcal{V}$.

A *variety* of languages is a positive variety of languages $\mathcal{V}$ such that for each finite alphabet A, $A^*\mathcal{V}$ is closed under complement. Given a variety of languages $\mathcal{V}$, the polynomial closure of $\mathcal{V}$, denoted $\mathrm{Pol}\mathcal{V}$, is the class of regular languages such that $L \in A^*\mathrm{Pol}\mathcal{V}$ if and only if L is a finite union of languages of the form

$$L_0 a_1 L_1 \cdots a_k L_k$$

with $L_i \in A^*\mathcal{V}$ and $a_i \in A$.

The following result is proved in [24, Theorem 5.9]:

Theorem 5.1. *Let $\mathcal{V}$ be a variety of languages. Then $\mathrm{Pol}\mathcal{V}$ is a positive variety of languages.*

A regular language L of A^* is said *commutative* if for every $a, b \in A$, $xaby \in L$ implies $xbay \in L$. An automaton is said *commutative* if $q \cdot ab = q \cdot ba$ for every couple of letters a and b and every state q.

The following equivalences are well known and are just recalled.

Proposition 5.2. *Let L be a regular language on A^*. We have the following equivalences :*

(1) *L is commutative.*
(2) *The syntactic monoide of L is commutative.*
(3) *The minimal automaton of L is commutative.*

Note that a language recognized by an automaton (not necessarily the minimal one) which is commutative is commutative. As an immediate consequence, we have :

Lemma 5.3. *If $\mathcal{A} = (Q, A, E, i, F)$ is the minimal automaton of a commutative language, then for all $p, q \in Q$, $L(\mathcal{A}_{p,q})$ is a commutative language.*

Proof. Commutativity of automata does not depend on their initial and final states. □

The class of commutative regular languages is known to be a variety of languages and is denoted by $\mathcal{C}$. Therefore, some direct consequences are the following.

Lemma 5.4. *Let A be an alphabet:*

(1) *$A^*\mathrm{Pol}\mathcal{C}$ is closed under concatenation.*
(2) *A regular language belongs to $A^*\mathrm{Pol}\mathcal{C}$ if and only if it is a finite union of concatenations of regular commutative languages.*
(3) *An APC language over A^* belongs to $A^*\mathrm{Pol}\mathcal{C}$.*

Proof. (1) Let us take $L = L_0a_1L_1 \cdots a_nL_n$ and $K = K_0b_1K_1 \cdots b_mK_m$ with $L_i, K_j \in A^*\mathcal{C}$ and $a_i, b_j \in A$. If $\varepsilon \in L_n$ then

$$LK = K \cup \bigcup_{x \in A} L_0a_1L_1 \cdots a_nL_nx^{-1}xK_0b_1K_1 \cdots b_mK_m$$

If $\varepsilon \notin L$, then

$$LK = \bigcup_{x \in A} L_0a_1L_1 \cdots a_nL_nx^{-1}xK_0b_1K_1 \cdots b_mK_m$$

Since A is finite, the unions are finite. Moreover, the class of commutative languages is a variety of languages, thus L_nx^{-1} is a commutative language and LK is in $A^*\mathrm{Pol}\mathcal{C}$.

(2) From the definition of $\mathrm{Pol}\mathcal{C}$, (1) and the fact that a single symbol is a regular commutative language.
(3) From what precedes and the fact that B^*, with $B \subseteq A$, is a commutative language.

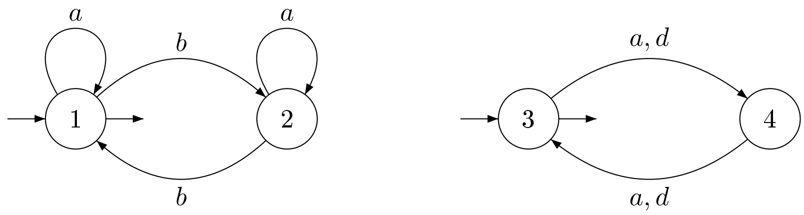
□

Now, we prove that $\mathrm{Pol}\mathcal{C}$ is closed under semi-commutation. Since any commutative language is trivially R-closed for all semi-commutation relation R, from Lemma 5.4 and Proposition 2.1 it is sufficient to prove that $L_n \sqcup\!\sqcup_R L_{n-1} \sqcup\!\sqcup_R \cdots \sqcup\!\sqcup_R L_1$ belongs to $A^*\mathrm{Pol}\mathcal{C}$ for every integer $n \geq 2$ and language $L_i \in A^*\mathcal{C}$. Let us begin with $n = 2$.

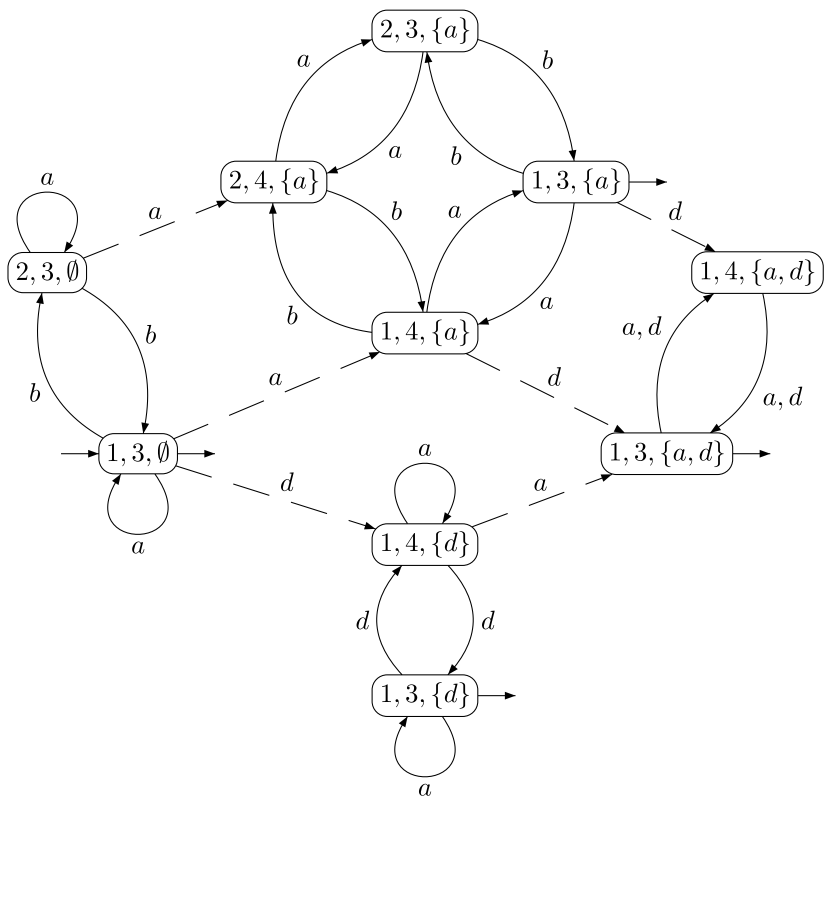
Lemma 5.5. *Let A be an alphabet, L_1 and L_2 be two regular commutative languages on A, and R be a semi-commutation relation over A. Then $L_1 \sqcup\!\sqcup_R L_2$ belongs to $A^*\mathrm{Pol}\mathcal{C}$.*

Before the proof, let us consider the following example.

Example 5.6. Consider the two following finite automata $\mathcal{A}_1$ and $\mathcal{A}_2$. They are commutative and their languages are as follows : $L(\mathcal{A}_1) = \{u \in \{a, b\}^* \mid |u|_b \text{ is even}\}$ and $L(\mathcal{A}_2) = \{u \in \{a, d\}^* \mid |u| \text{ is even}\}$.



Let us take $R = \{(a, d), (b, a)\}$. Using the constructive proofs of the above lemmas, $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ is given by the following finite automaton (transitions which change the third part of states are represented by dashed arrows; and only reachable states which lead to a final state have been represented):



Since the parts between the dashed arrows are commutative and since no dashed arrow belongs to a loop, $L(\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2)$ can be easily described as a finite union of concatenations of commutative languages (recall that a single symbol is a commutative language). Therefore, $L(\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2)$ belongs to $A^*\mathrm{Pol}\mathcal{C}$.

Proof. Let $\mathcal{A}_1 = (Q_1, A, E_1, I_1, F_1)$ and $\mathcal{A}_2 = (Q_2, A, E_2, I_2, F_2)$ be the two minimal automata recognizing L_1 and L_2, respectively. Let $\mathcal{A} = (Q, A, E, I, F)$ be the trim automaton obtained from $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$. For all subsets B of $\alpha(L(\mathcal{A}_2))$, we denote by $Q_{\overleftrightarrow{B}}$ the subset $\{(q_1, q_2, \overleftrightarrow{B}) \mid q_1 \in Q_1, q_2 \in Q_2\}$ of Q and by $E_{\overleftrightarrow{B}}$ the subset $E \cap Q_{\overleftrightarrow{B}} \times A \times Q_{\overleftrightarrow{B}}$ of E.

Let $t = ((p, q, \overleftrightarrow{C}), a, (p', q', \overleftrightarrow{D})) \in E \setminus \cup_{B \subseteq A} E_{\overleftrightarrow{B}}$. We claim that there is no loop in $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ using t: since $\overleftrightarrow{C} \neq \overleftrightarrow{D}$ and by (i) – proof of Proposition 3.1 – all states accessible from $(p', q', \overleftrightarrow{D})$ are of the form $(r, s, \overleftrightarrow{B})$, with $\overleftrightarrow{D} \subseteq \overleftrightarrow{B}$.

Each accepting path m in $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ can be decomposed into:

$$m = m_0 t_1 m_1 t_2 \ldots t_n m_n$$

with $t_i \in E \setminus \cup_{B \subseteq A} E_{\overleftrightarrow{B}}$ and m_i only using transitions of $E_{\overleftrightarrow{B_i}}$. Using the above claim, we have $n \leq |E \setminus \cup_{B \subseteq A} E_{\overleftrightarrow{B}}|$. Consequently, $L(\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2)$ is a finite union of languages of the form:

$$L_0 a_1 L_1 a_2 \ldots a_n L_n,$$

where the a_i's are letters and the L_i's are accepted by finite automata whose graphs of transitions are $(Q_{\overleftrightarrow{B_i}}, E_{\overleftrightarrow{B_i}})$.

By definition of $\mathrm{Pol}\mathcal{C}$, it remains to prove that the L_i's are commutative languages. Let $B \subseteq A$, we prove that $(Q_{\overleftrightarrow{B}}, E_{\overleftrightarrow{B}})$ is commutative. Let $r = (p, q, \overleftrightarrow{B})$, $r_a = (p_a, q_a, \overleftrightarrow{B})$ and $r_{ab} = (p_{ab}, q_{ab}, \overleftrightarrow{B})$ three states of $Q_{\overleftrightarrow{B}}$ such that there exist transitions $t_a = (r, a, r_a)$ and $t_{ab} = (r_a, b, r_{ab})$ in $E_{\overleftrightarrow{B}}$.

$$(p, q, \overleftrightarrow{B}) \xrightarrow{\quad a \quad} (p_a, q_a, \overleftrightarrow{B}) \xrightarrow{\quad b \quad} (p_{ab}, q_{ab}, \overleftrightarrow{B})$$

With the notation of Proposition 3.1, the following cases occur:

- $t_a, t_{ab} \in G_1$. Since $\mathcal{A}_1$ is minimal and since $L(\mathcal{A}_1)$ is commutative, it is commutative. Thus there exists p_b in Q_1 such that $p \cdot b = p_b$ and $p_b \cdot a = p_{ab}$. Moreover, since t_a and t_{ab} belong to G_1, $\{a\} \times \overleftrightarrow{B} \subseteq R$ and $\{b\} \times \overleftrightarrow{B} \subseteq R$. Consequently, $(r, b, (p_b, q, \overleftrightarrow{B}))$ and $((p_b, q, \overleftrightarrow{B}), a, r_{ab})$ are in $G_1 \cap E_{\overleftrightarrow{B}}$. It follows that $r_{ab} \in r \cdot ba$.
- $t_a, t_{ab} \in G_2$. By a similar argument on $\mathcal{A}_2$, one has $r_{ab} \in r \cdot ba$.
- $t_a \in G_1, t_{ab} \in G_2$. Thus $q_a = q$ and $p_{ab} = p_a$. Consequently, $(r, b, (p, q_{ab}, \overleftrightarrow{B})) \in G_2 \cap E_{\overleftrightarrow{B}}$ and $((p, q_{ab}, \overleftrightarrow{B}), a, r_{ab}) \in G_1 \cap E_{\overleftrightarrow{B}}$. It follows that $r_{ab} \in r \cdot ba$.

- $t_a \in G_2, t_{ab} \in G_1$. By a similar argument on $\mathcal{A}_2$, one has $r_{ab} \in r \cdot ba$.

Consequently $r \cdot ab \subseteq r \cdot ba$. Since the roles of a and b are symmetric, then $r \cdot ba \subseteq r \cdot ab$ and thus $r \cdot ab = r \cdot ba$. Therefore, $(Q_{\overleftrightarrow{B}}, E_{\overleftrightarrow{B}})$ is commutative, which concludes the proof. □

To do the recurrence step that will lead to the stability of Pol$\mathcal{C}$ under semi-commutation, let L' and L_i, with $1 \leq i \leq n+1$, be $n+2$ commutative regular languages. Suppose we have proved that $L = L' \sqcup\!\sqcup_R L_{n+1}L_n \cdots L_1$ can be decomposed into a finite union of languages of the form $(L'' \sqcup\!\sqcup_R L'_{n+1})(L''' \sqcup\!\sqcup_R L_n \cdots L_1$ with L'', L'_{n+1} and L''' some commutative regular languages. Then, by the inductive hypothesis, the right part belongs to A^*Pol$\mathcal{C}$. By the preceding lemma, the left part also belongs to A^*Pol$\mathcal{C}$. And by the stability of A^*Pol$\mathcal{C}$ under concatenation, we conclude that L belongs to A^*Pol$\mathcal{C}$.

Lemma 5.7. *Let $\mathcal{A} = (Q, A, E, I, F)$ be a finite automaton, L_1, L_2 be two languages on A and R be a semi-commutation relation over A. The following equality holds:*

$$L(\mathcal{A}) \sqcup\!\sqcup_R L_1L_2 = \bigcup_{\substack{q \in Q,\\ C \times B \subseteq R}} (L(\mathcal{A}_{I,q}) \sqcup\!\sqcup_R (L_1 \cap B^*))(L(\mathcal{A}_{q,F}) \cap C^*) \sqcup\!\sqcup_R L_2)$$

Proof. Let $q \in Q$, $C \times B \subseteq R$ and $u \in (L(\mathcal{A}_{I,q}) \sqcup\!\sqcup_R (L_1 \cap B^*))(L(\mathcal{A}_{q,F}) \cap C^*) \sqcup\!\sqcup_R L_2)$. Then u can be decomposed into:

$$u = x_1y_1 \dots x_ny_nz_1t_1 \dots z_kt_k$$

such that

(1) $x_1 \dots x_n \in L(\mathcal{A}_{I,q})$, $y_1 \dots y_n \in L_1 \cap B^*$,
(2) for all $1 \leq j < i \leq n$, $\alpha(x_i) \times \alpha(y_j) \subseteq R$,
(3) $z_1 \dots z_k \in L(\mathcal{A}_{q,F}) \cap C^*$, $t_1 \dots t_k \in L_2$,
(4) for all $1 \leq j < i \leq k$, $\alpha(z_i) \times \alpha(t_j) \subseteq R$,

Since $C \times B \subseteq R$ and by (1) and (3), for all $1 \leq i \leq n$ and for all $1 \leq j \leq k$, $\alpha(z_j) \times \alpha(y_i) \subseteq R$. Consequently and by (2) and (4), $u \in L(\mathcal{A}) \sqcup\!\sqcup_R L_1L_2$.

Conversely, let $u \in L(\mathcal{A}) \sqcup\!\sqcup_R L_1L_2$. By definition of the R-shuffle, there exist $x_1, \dots, x_n, y_1 \dots, y_n \in A^*$ such that

(5) $u = x_1y_1 \dots x_ny_n$
(6) for all $1 \leq j < i \leq n$, $\alpha(x_i) \times \alpha(y_j) \subseteq R$,
(7) $x_1 \dots x_n \in L(\mathcal{A})$,
(8) $y_1 \dots y_n \in L_1L_2$.

Statement (8) implies that there is $1 \leq k \leq n$ such that y_k may be decomposed into $y_k = st$, with $s, t \in A^*$ and $y_1 \dots y_{k-1}s \in L_1$ and $ty_{k+1} \dots y_n \in L_2$. Statement (7) implies that there exists a state q such that $x_1 \dots x_k \in L(\mathcal{A}_{I,q})$ and $x_{k+1} \dots x_n \in L(\mathcal{A}_{q,F})$. Now, by (5) and (6), $x_1y_1 \dots x_ks \in L(\mathcal{A}_{I,q}) \sqcup\!\sqcup_R (L_1 \cap \alpha(y_1 \dots y_{k-1}s))$ and $tx_{k+1}y_{k+1} \dots x_ny_n \in (L(\mathcal{A}_{q,F}) \cap \alpha(x_{k+1} \dots x_n)) \sqcup\!\sqcup_R L_2$. By (6), $\alpha(x_{k+1} \dots x_n) \times \alpha(y_1 \dots y_{k-1}s) \subseteq R$, which concludes the proof. □

We can now prove the main result.

Theorem 5.8. *The class* Pol$\mathcal{C}$ *is closed under conjugacy and semi-commutation.*

Proof. Let $L_0, L_1, \ldots, L_k$ be commutative languages on A, $a_1, \ldots, a_k$ be letters of A and $L = L_0 a_1 L_1 a_2 \cdots a_k L_k$. Let $\mathcal{A}_i$ be the minimal automaton of L_i. One has

$$\mathrm{Conj}(L) = \bigcup_{0 \leq i \leq k} L(\mathcal{A}_{i q_i, F_i}) a_{i+1} L(\mathcal{A}_{i+1}) \ldots a_k L(\mathcal{A}_k) L(\mathcal{A}_1) a_1 \ldots a_i L(\mathcal{A}_{i p_i, q_i})$$

where p_i is the initial state of $\mathcal{A}_i$, F_i is the set of final states of $\mathcal{A}_i$ and q_i is a state of $\mathcal{A}_i$. Thus using Lemmas 5.4 and 5.3, $\mathrm{Conj}(L) \in A^*\mathrm{Pol}\mathcal{C}$. Furthermore, if K_1 and K_2 are languages of A^*, then $\mathrm{Conj}(K_1 \cup K_2) = \mathrm{Conj}(K_1) \cup \mathrm{Conj}(K_2)$. It follows that Pol$\mathcal{C}$ is closed under conjugacy.

By a direct induction using Proposition 2.1, Lemma 5.5 and Lemma 5.7, Pol$\mathcal{C}$ is also closed under semi-commutation. □

Let us notice that the proof is constructive. By Theorem 5.1 and 5.8, the positive variety Pol$\mathcal{C}$ is closed under union, intersection, left and right quotients, conjugacy and semi-commutations.

6. Conclusion

In this paper we proved that computing the semi-commutation closure of an APC language is in practice more efficient using finite automata representations than using regular expressions.

Moreover, in [7] the question of finding other subclasses of regular languages which are closed under union, intersection, product, semi-commutation rewriting and conjugacy was opened. We showed that Pol$\mathcal{C}$, the positive variety of finite unions of finite products of commutative languages, contains APC languages and has these closure properties. Furthermore, using finite automata the semi-commutation closure of a language of this kind is effectively computable. However we do not know whether this class is maximal. A solution may be found in [16] where the maximal positive variety closed under the shuffle operation is exhibited. We do not know neither whether Pol$\mathcal{C}$ is decidable.

In practice, we may want to compute the transitive closure, by a semi-commutation relation, of a regular language which does not necessarily belong to a class stable by all semi-commutation relations. We investigated this problem, in a separate work [10]. We mainly used the fact that our technique of computing the R-shuffle works on any two regular languages. This allowed us to compute the reachability set of a lift-controller whose transition relation is not only composed by semi-commutations and whose reachability set does not belong to a class stable by all semi-commutation relations.

Acknowledgement
We would like to thank Giovanna Guaiana for the references she gave us about her

work concerning partial commutations (symetric semi-commutations). Theorem 4.1 is an adaptation of the proof of Theorem 1 in [17].

References

[1] P. Abdulla, A. Annichini, and A. Bouajjani. Algorithmic verification of lossy channel systems: An appliction to the bounded retransmission protocol. In *TACAS'99*, volume 1579 of *Lecture Notes in Computer Science*, pages 208–222, 1999.

[2] P. A. Abdulla, A. Bouajjani, and B. Jonsson. On-the-fly analysis of systems with unbounded, lossy FIFO channels. In *CAV'98*, volume 1427 of *Lecture Notes in Computer Science*, pages 305–322, 1998.

[3] Parosh Aziz Abdulla, Bengt Jonsson, Marcus Nilsson, and Julien d'Orso. Algorithmic improvements in regular model checking. In *CAV'03*, LNCS, 2003.

[4] Jean Berstel. *Transductions and Context-Free Languages*. B.G. Teubner, Stuttgart, 1979.

[5] B. Boigelot and P. Godefroid. Symbolic verification of communication protocols with infinite state spaces using QDDs. In *Proc. of* 8^{th} *CAV (August), USA*, volume 1102, pages 1–12. LNCS, 1996.

[6] Bernard Boigelot and Pierre Wolper. Verifying systems with infinite but regular state spaces. In *CAV'98*, volume 1427 of *LNCS*, pages 88–97, June 1998.

[7] A. Bouajjani, A. Muscholl, and T. Touili. Permutation rewriting and algorithmic verification. In *LICS'01*, IEEE Computer Society, pages 399–408, 2001.

[8] J.A. Brzozowski. Hierarchies of aperiodic languages. *Informatique théorique et Application/Theoretical Informatics and Applications*, 10:33–49, 1976.

[9] J.A. Brzozowski and I. Simon. Characterizations of locally testable languages. *Discrete Mathematics*, 4:243–271, 1973.

[10] G. Cécé, P.-C. Héam, and Y. Mainier. Clôture transitives de semi-commutations et model-checking régulier. In *AFADL'04*, 2004.

[11] V. Diekert and G. Rozenberg, editors. *Book of Traces*. World Scientific, Singapore, 1995.

[12] Volker Diekert and Yves Métivier. Partial commutation and traces. In G. Rozenberg and A. Salomaa, editors, *Handbook on Formal Languages*, volume III. Springer, Berlin-Heidelberg-New York, 1997.

[13] Z. Esik and I. Simon. Modeling literal morphisms by shuffle. *Semigroup Forum*, 56:225–227, 1998.

[14] P. Godefroid and P. Wolper. A partial approach to model checking. *Information and Computation*, 110(2):305–326, 1994.

[15] A. Cano Gomez and J.-E. Pin. On a conjecture of schnoebelen,. In *DLT'03*, Lecture Notes in Computer Science, 2003.

[16] A. Cano Gomez and J.-E. Pin. Shuffle on positive varieties of languages. *Theoretical Computer Science*, 312:433–461, 2004.

[17] Giovanna Guaiana, Antonio Restivo, and Sergio Salemi. On the trace product and some families of languages closed under partial commutations. *Journal of Automata, Languages and Combinatorics*, 9(1):61–79, 2004.

[18] J. Hopcroft and J. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley, 1980.

[19] P.-C. Héam. Some complexity results for polynomial rational expressions. *TCS: Theoretical Computer Science*, 299, 2003.

[20] X. Leroy, D. Doligez, J. Garrigue, D. Rémy, and J. Vouillon. *The Objective Caml system, release 3.06*. Inria, 2002.

[21] D. Lugiez and Ph. Schnoebelen. The regular viewpoint on pa-processes. In *9th Int. Conf. Concurrency Theory (CONCUR'98)*, volume 1466 of *Lecture Notes in Computer Science*. Springer, 1998.

[22] J.-F. Perrot. Variété de langages et opérations. *Theoretical Computer Science*, 7:197–210, 1978.

[23] J.-E. Pin. *Varieties of formal languages*. Foundations of Computer Science, 1984.

[24] Jean-Eric Pin and Pascal Weil. Polynomial closure and unambiguous product. *Theory Comput. Systems*, 30:1–39, 1997.

[25] Ph. Schnoeboelen. Decomposable regular languages and the shuffle operator. *EATCS Bull.*, 67:283–289, 1999.

[26] H. Straubing. Finite semigroups varieties of the form $\mathbf{V}*\mathbf{D}$. *Journal of Pure and Applied Algebra*, 36:53–94, 1985.

[27] P. Tesson and D. Thérien. Diamonds are forever: the variety da. In *International Conference on Semigroups, Algorithms, Automata and Languages*, 2002.

[28] W. Thomas. Classifying regular events in symbolic logic. *Journal of Computer and System Science*, 25:360–375, 1982.

[29] D. Thérien. Classification of finite monoids: the language approach. *Theoretical Computer Science*, 14:195–208, 1981.

[30] T. Touili. Regular model checking using widening techniques. In *1st Vepas Workshop*, volume 50 of *Electronic Notes in TCS*, 2001.

Communicated by (The editor will be set by the publisher).