Efficiency of Automata in Semi-Commutation Verification Techniques

Gérard Cécé, Pierre-Cyrille Héam, Yann Mainier

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Efficiency of Automata in Semi-Commutation Verification Techniques

Gérard Cécé — Pierre-Cyrille Héam — Yann Mainier

N° 5001

Novembre 2003

THÈME 2

ISSN 0249-6399 ISRN INRIA/RR--5001--FR+ENG

Efficiency of Automata in Semi-Commutation Verification Techniques

Gérard Cécé *, Pierre-Cyrille Héam *, Yann Mainier*

Thème 2 — Génie logiciel
et calcul symbolique
Projet Cassis

Rapport de recherche n° 5001 — Novembre 2003 — 20 pages

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 proved that the class of APC regular languages is closed under all semi-commutation relations R. Moreover a recursive algorithm on the regular expression is given to compute the image of an APC language by the transitive closure of R. This paper provides a new approach, based on automata, of the same question. 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, PolC, answered the open question proposed in Bouajjani and al.'s paper.

Key-words: Verification, Regular Model Cheking, Finite Automata, Semi-commutations

* LIFC - Université de Franche-Comté cece/heampc/mainier@lifc.univ-fcomte.fr

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy (France)
Téléphone : +33 3 83 59 30 00 —Télécopie : +33 3 83 27 83 19

Efficacité des automates pour les techniques de vérification à bases de semi-commutations

Résumé : Calculer l'image d'un langage régulier par la clôture transitive d'une relation est une question centrale en *Regular Model Checking*. Dans un article récent, Bouajjani, Muscholl and Touili ont prouvé que la classe des langages réguliers APC est stable par toute les relations de semi-commutation R. De plus, un algorithme récursif sur l'expression régulière est donnée pour calculer l'image d'un tel langage par la clôture transitive de R. Dans notre article, nous proposons une nouvelle approche, utilisant les automates finis, pour résoudre cette question. Nous obtenons un algorithme plus simple et plus efficace qui s'applique à une classe plus large de langages réguliers stable par union, intersection, conjugaision et par les relations de semi-commutation. L'existence de cette nouvelle classe PolC répond à une question ouverte proposée par Bouajjani et al. dans leur papier.

Mots-clés : Vérification, langages réguliers, automates finis, semi-commutations

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 parallelisme in Mazurkiewicz trace theory [DR95], in partial order reduction techniques [GW94] or to express exchange of a piece of information between neighbouring processes in linear or ring networks. In *regular model checking* [BG96, BW98] [AJNd03], a key point is the computation of the image of a regular language by the transitive closure of a relation. However, such computation, in 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, $R^*(L) \cap b^*a^* = b^n a^n$. Therefore $R^*(L)$ is not regular. In [BMT01], 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 product of languages of the form a_0 or $\{a_1, a_2, \ldots, a_n\}^*$ with a_i's letters) 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 [BMT01], APC is an interesting subclass of regular languages; several verification problems (sliding window protocols, parameterized mutual exclusion protocols, etc...) can be modelized 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 union of languages of the form $L_0 a_0 L_1 a_1 \ldots a_k L_k$ where the a_i's are letters and the L_i's are commutative regular languages. This class allows to describe languages such as: $L_1 d L_2$, with $L_1 = \{u \in \{a,b\}^* \mid |u|_b \text{ is even and } |u|_a \text{ is even}\}$ and $L_2 = \{u \in \{a,d\}^* \mid |u| \text{ is odd}\}$

1.1 Related Works

Regular model checking [BG96, BW98, AJNd03] 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 computing $R^*(L)$ for a given regular languages 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 used automata during the whole computation. This last remark is another plus for our technique compared to that of [BMT01].

Polynomial closure of varieties of regular languages is an operation widely studied in the literature (see *for example* [PW97, Tho82, Brz76, BS73]). In this paper we consider the languages of level 3/2 in the Straubing-Thérien hierarchy [Str85, Thé81] which represents the current border for decidability problems and whose structure makes them suitable for verification of certain systems [ABJ98, AAB99, BMT01, Tou01]. Decomposable languages is a class of regular languages used for the simulation of process algebra [LS98]. It was conjectured in [Sch99] that this class was exactly Pol$\mathcal{V}$. However this conjecture has just been invalidated in [GP03a]. 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 [ES98, Per78] and for positive varieties [GP03b].

1.2 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, that manipulating regular expressions [BMT01] and ours using automata. Then we extends, 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 [Ber79, HU80].

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

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

If u is a finite word, $\alpha(u)$ denotes the set of letters occurring in u. This notion is extended to languages as follow

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

A semi-commutation relation R on a finite alphabet A, is a relation on A. Given a finite word u on A, we denote by $R(u) = \{xbay \mid x, y \in A^*,\ a, b \in A^*,\ (a, b) \in R \text{ and } xaby = u\}$ and by $R^*(u)$ the language $\{u\} \cup_{k \geq 1} R^k(u)$. This 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 R-shuffle of u and v, denoted $u \sqcup\!\sqcup_R v$, is the set of words of the form $u_1 v_1 \dots u_n v_n$ with $u = u_1 \dots u_n$, $v = v_1 \dots v_n$, $u_i, v_i \in A^*$ for all $1 \leq i \leq n$ and such that $(\alpha(u_i), \alpha(v_j)) \subseteq R$ for all $j < i$. The R-shuffle operation is extended to languages L and K of A^* by

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

As stated in the following Proposition, deduced from Lemma 3.1 and 3.2 of [BMT01], 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 1 ([BMT01]) *Let L_i's be R-closed sets, i.e. such that $L_i = R^*(L_i)$, then we have:*

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

As an APC language L is a finite union of products of trivially R-closed languages (mere letters or star expressions of subsets of the alphabet), computing $R^*(L)$ is reduced to the computation of the R-shuffle of languages. Since [BMT01] 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.

Proposition 2 *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 R_B the set $\{a \in \alpha(L(\mathcal{A}_1)) \mid (a, B) \subseteq R\}$ and by $\overline{B}$ the set $\{a \in \alpha(L(\mathcal{A}_2)) \mid (R_B, a) \subseteq R\}$.*

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

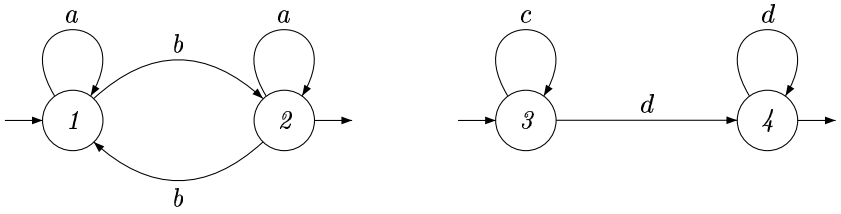
- $Q = Q_1 \times Q_2 \times \mathcal{P}(A)$,
- $I = \{(p_1, p_2, \overline{\emptyset}) \mid p_1 \in I_1, p_2 \in I_2\}$,
- $F = \{(p_1, p_2, B) \mid p_1 \in F_1, p_2 \in F_2, B \subseteq A\}$,
- $E = G_1 \cup G_2$, *with*

  $G_1 = \{((p_1, p_2, B), a, (q_1, p_2, B)) \mid p_1 \in Q_1, p_2 \in Q_2, q_1 \in p_1 \cdot a, B \subseteq A$ *and* $(a, B) \subseteq R\}$ *and*

  $G_2 = \{((p_1, p_2, B), a, (p_1, q_2, \overline{B \cup \{a\}})) \mid p_1 \in Q_1, p_2 \in Q_2, q_2 \in p_2 \cdot a, B \subseteq A\}$.

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

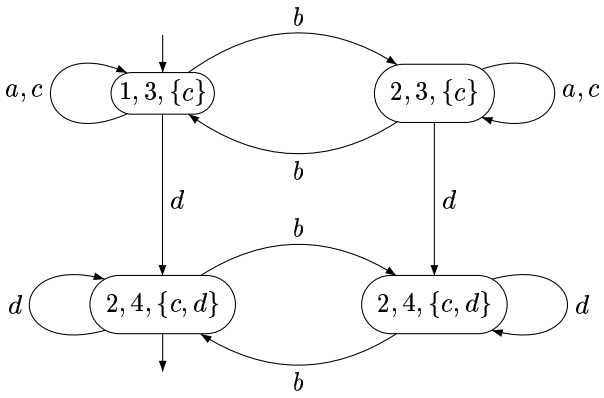
Example 3 *Consider the following finite automata* $\mathcal{A}_1$ *and* $\mathcal{A}_2$*:*



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

B	R_B	$\overline{B}$
$\emptyset$	$\{a, b\}$	$\{c\}$
$\{c\}$	$\{a, b\}$	$\{c\}$
$\{d\}$	$\{b\}$	$\{c, d\}$
$\{c, d\}$	$\{b\}$	$\{c, d\}$

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 2 we replace G_2 by the set of transitions $G'_2 = \{((p_1, p_2, B), a, (p_1, q_2, B \cup \{a\})) \mid p_1 \in Q_1, p_2 \in Q_2, q_2 \in p_2 \cdot a, B \subseteq A\}$ 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 recognising $L(\mathcal{A}_1) \sqcup\!\sqcup_R L(\mathcal{A}_2)$ and easier to construct but with a larger number of states.

Now we prove Proposition 2.

PROOF. First we prove some technical properties of the function $\overline{\cdot}$.

(i) For all $B \subseteq A$, $B \subseteq \overline{B}$: let $b \in B$. By definition of R_B, for each $a \in R_B$, $(a, b) \in R$. Thus $b \in \overline{B}$.

(ii) For all $B \subseteq A$, $R_B = R_{\overline{B}}$ and $\overline{\overline{B}} = \overline{B}$: by (i), $R_{\overline{B}} \subseteq R_B$. Conversely, by definition of $\overline{B}$, $(R_B, \overline{B}) \subseteq R$. Consequently, $R_B \subseteq R_{\overline{B}}$. It follows that $R_B = R_{\overline{B}}$ and $\overline{\overline{B}} = \overline{B}$.

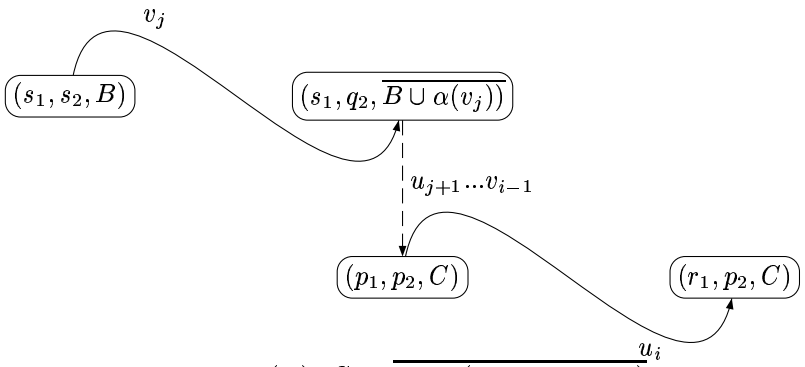
(iii) For all $a_1, a_2 \in \alpha(L(\mathcal{A}_2))$, $\overline{\overline{B \cup \{a_1\}} \cup \{a_2\}} = \overline{B \cup \{a_1, a_2\}}$: by (ii), $R_{\overline{B \cup \{a_1\}}} = R_{B \cup \{a_1\}}$. Furthermore, by definition, a letter a belongs to $R_{B \cup \{a_1, a_2\}}$ if and only if $a \in R_{B \cup \{a_1\}}$ and $(a, a_2) \in R$. It follows that $a \in R_{B \cup \{a_1, a_2\}}$ if and only if $a \in R_{\overline{B \cup \{a_1\}}}$ and $(a, a_2) \in R$. Consequently, $R_{\overline{B \cup \{a_1\}} \cup \{a_2\}} = R_{B \cup \{a_1, a_2\}}$, proving (iii).

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 a successful path m in $\mathcal{A}$ labeled by w. The path m can be decomposed into:

$$m = m_1, m_2, m_3, \ldots, m_k$$

such that each m_{2i+1} $(0 \leq i \leq (k-1)/2)$ only uses transitions of G_1 and each m_{2i} $(1 \leq i \leq k/2)$ only uses transitions of G_2 (some of them may be empty). Now, let us denote by u_{i+1}

the label of m_{2i+1} and v_i the label of m_{2i}. By construction, $w = u_1v_1u_2 \dots u_rv_r$ ($r = k/2$ if k is even and $r = (k-1)/2$ if k is odd) with $u_1 \dots u_r \in L(\mathcal{A}_1)$ and $v_1 \dots v_r \in L(\mathcal{A}_2)$. We claim that for all $1 \leq j < i \leq r$, $(\alpha(u_i), \alpha(v_j)) \subseteq R$. Indeed, let $1 \leq j < i \leq r$. Assume that u_i or v_j is empty. Then $(\alpha(u_i), \alpha(v_j)) = \emptyset \subseteq R$. 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 $(s_1, q_2, \overline{B \cup \alpha(v_j)})$. Let (p_1, p_2, C) 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), $C = \overline{B \cup \alpha(v_jv_{j+1} \dots v_{i-1})}$. It follows that $\overline{B \cup \alpha(v_j)} \subseteq C$. Moreover, since the path m_{2i+1} only uses transitions of G_1, each letter $a \in \alpha(u_i)$ has to satisfy $(a, C) \subseteq R$. It follows that $(\alpha(u_i), \alpha(v_j)) \subseteq R$, 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 $x_1x_2 \dots x_n \in L(\mathcal{A}_1)$, $y_1y_2 \dots y_n \in L(\mathcal{A}_2)$ for all $1 \leq i \leq n$ and for all $1 \leq j < i \leq n$, $(\alpha(x_i), \alpha(y_j)) \subseteq R$. Since $x_1x_2 \dots x_n \in L(\mathcal{A}_1)$, there exist $p_0, p_1, \dots, p_n \in Q_1$ such that

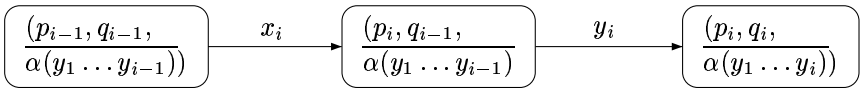
- $p_0 \in I_1$,
- $p_n \in F_1$,
- for all $i \in \{1, \dots, n\}$, there exists a path in $\mathcal{A}_1$ from p_{i-1} to p_i labeled by x_i.

Since $y_1y_2 \dots y_n \in L(\mathcal{A}_1)$, 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 labeled by y_i.

For all $i \in \{1, \dots, n\}$, let us denote by t_i the word $y_1 \dots y_i$. Moreover, let $t_0 = \varepsilon$. We claim that for all $i \in \{1, \dots, n\}$, there exist a path in $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ labeled by x_i from

$(p_{i-1}, q_{i-1}, \overline{\alpha(t_{i-1})})$ to $(p_i, \underline{q_{i-1}}, \overline{\alpha(t_{i-1})})$ and a path in $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ labeled by y_i from $(p_i, q_{i-1}, \overline{\alpha(t_{i-1})})$ to $(p_i, q_i, \alpha(t_i))$.



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

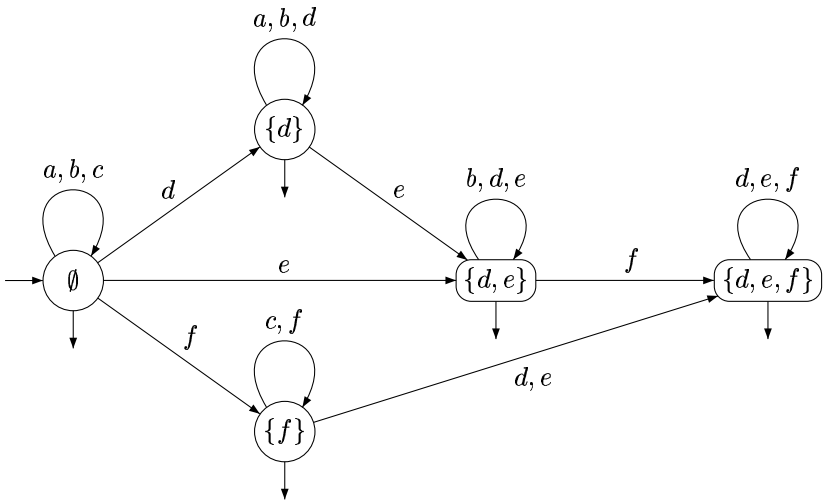
Remark that the automaton $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ may be non-deterministic, even when $\mathcal{A}_1$ and $\mathcal{A}_2$ are both deterministic.

4 Application to APC

Let us first start by an example. Let $B = \{a, b, c\}, C = \{d, e, f\}$ and $R = \{(a, d), (c, f), (b, d), (b, e)\}$. Using Proposition 2, one has

B	R_B	$\overline{B}$
$\emptyset$	$\{a, b, c\}$	$\emptyset$
$\{d\}$	$\{a, b\}$	$\{d\}$
$\{e\}$	$\{b\}$	$\{d, e\}$
$\{f\}$	$\{c\}$	$\{f\}$
$\{d, e\}$	$\{b\}$	$\{d, e\}$
$\{e, f\}$	$\emptyset$	$\{d, e, f\}$
$\{d, f\}$	$\emptyset$	$\{d, e, f\}$
$\{d, e, f\}$	$\emptyset$	$\{d, e, f\}$

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



Using [BMT01, Example 2], one obtains that $R^*(B^*C^*) = \{a,b,c\}^*\{c,f\}^*\ \{d,e,f\}^* \cup \{a,b,c\}^*\{a,b,d\}^*\{b,d,e\}^*\{d,e,f\}^*$ which is precisely the language of the automaton given above. The compactness of automata is already revealed in this example by sharing of the states representing respectively the expressions $\{a,b,c\}^*$ and $\{d,e,f\}^*$.

Definition 4 ([TT02]) *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 as 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. One can wonder whether our algorithm reduces to encode an APC expression in a finite partially ordered automaton and to apply the algorithm of [BMT01] on it while merging equivalent states. The answer is no since it was proved in [BMT01] that merging equivalent state 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 easily done with automata. Furthermore, a POF-automaton equivalent to an APC expression can be easily done in linear time and space [Héa03]).

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

Experimental Tests

In order to compare both techniques, this of [BMT01] 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 [LDG+02].

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

type 1: $A_0^* a_1 A_2^* \dots a_{n-1} A_n^*$

type 2: $A_0^* A_1^* \dots A_n^*$

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 [BMT01], then our algorithm on the equivalent automaton of the same product. And, we got sizes 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 in function of 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 in function of $|R|$ 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 Athlon of 1.3 GHz with 1 GB of memory. Processes of our techniques used less than 3 or 4 MB of memory while those corresponding to [BMT01] 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 kind of test was not feasible with the technique of [BMT01].

5 Permutation Rewriting and Polynomial Closure of Commutative Regular Languages

In this section we present our second main result: the extention of [BMT01] to a larger class of regular languages.

For a general reference on varieties of formal languages see [Pin84].

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 all finite alphabet A, all semi-commutation relation over A and all 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 Pol$\mathcal{V}$, is the class of regular languages such that $L \in A^*\text{Pol}\mathcal{V}$ 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 [PW97, Theorem 5.9]:

Theorem 5 *Let $\mathcal{V}$ be a variety of languages. Then* Pol$\mathcal{V}$ *is a positive variety of languages.*

A regular language of A^* is commutative if its syntactic monoid is commutative. It is equivalent to say that it is recognized by a commutative finite monoid. The class of commutative regular languages is known to be a variety of languages and is denoted $\mathcal{C}$.

Now, we prove that Pol$\mathcal{C}$ is closed under union, intersection, conjugacy and semi-commutation (see Theorem 11). First we need several lemma.

Lemma 6 *Let $\mathcal{A} = (Q, A, E, I, F)$ be a finite automata, L_1, L_2 be two languages on A and R a semi-commutation relation over A. The following equality holds:*

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

where the union is taken for all subsets B and C of A such that $(C, B) \subseteq R$.

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

$$u = x_1 y_1 \dots x_n y_n z_1 t_1 \dots k_k t_k$$

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 $1 \leq j < i \leq n$, $(\alpha(x_i), \alpha(y_j)) \subseteq R$,

(3) $z_1 \dots z_k \in L(\mathcal{A}_{q,F})$, $t_1 \dots t_k \in L_2 \cap C^*$,

(4) for all $1 \leq j < i \leq k$, $(\alpha(z_i), \alpha(t_j)) \subseteq R$,

Since $(C, B) \subseteq R$ and by (1) and (3), for all $1 \leq i \leq n$ and for all $1 \leq j \leq k$, $(\alpha(z_j), \alpha(y_j)) \subseteq R$. Consequently and by (2) and (4), $u \in L(\mathcal{A}) \sqcup\!\sqcup_R L_1 L_2$.

Conversely, let $u \in L(\mathcal{A}) \sqcup\!\sqcup_R L_1 L_2$. By definition of the R-shuffle, there exist $x_1, \dots, x_n, y_1 \dots, y_n \in A^*$ such that

(5) $u = x_1 y_1 \dots x_n y_n$

(6) for all $1 \leq j < i \leq n$, $(\alpha(x_i), \alpha(y_j)) \subseteq R$,

(7) $x_1 \dots x_n \in L(\mathcal{A})$,

(8) $y_1 \dots y_n \in L_1 L_2$.

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 $t y_{k+1} \dots y_n \in L_2$. 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), $x_1 y_1 \dots x_k s \in L(\mathcal{A}_{I,q}) \sqcup\!\sqcup_R (L_1 \cap \alpha(y_1 \dots y_{k-1} s))$ and $t x_{k+1} y_{k+1} \dots x_n y_n \in (L(\mathcal{A}_{q,F}) \cap \alpha(x_{k+1} \dots x_n)) \sqcup\!\sqcup_R L_2$. By (6), $(\alpha(x_{k+1} \dots x_n), \alpha(y_1 \dots y_{k-1} s)) \subseteq R$, which concludes the proof. □

Lemma 7 *If L and K are commutatively closed and regular language of A^*, then $LK \in A^* \mathrm{Pol}\mathcal{C}$.*

PROOF. If $\varepsilon \in L$, then $LK = K \cup \bigcup_{a \in A} La^{-1}aK$. If $\varepsilon \notin L$, then $LK = \bigcup_{a \in A} La^{-1}aK$. Since L is regular, it has a finite number of right quotients, thus the unions are finite. Moreover, the class of commutative languages is a variety of languages, thus La^{-1} is a commutative language, which concludes the proof. □

Lemma 8 *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. Since $L(\mathcal{A})$ is commutative, its syntactic monoid is commutative. Since $\mathcal{A}_{p,q}$ and $\mathcal{A}$ have the same monoid of transitions, and since the monoid of transitions of $\mathcal{A}$ is the syntactic monoid of $L(\mathcal{A})$, $L(\mathcal{A}_{p,q})$ is accepted by a commutative monoid, proving the lemma. □

Lemma 9 *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 $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$ are commutative languages, then $L(\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2) \in A^*\mathrm{Pol}\mathcal{C}$*

PROOF.

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 $Q_{\overline{B}}$ the subset $\{(q_1, q_2, \overline{B}) \mid q_1 \in Q_1, q_2 \in Q_2\}$ of Q and by $E_{\overline{B}}$ the subset $E \cap Q_{\overline{B}} \times A \times Q_{\overline{B}}$ of E.

Let $t = ((p, q, \overline{C}), a, (p', q', \overline{D})) \in E \setminus \cup_{B \subseteq A} E_{\overline{B}}$. We claim that there is no loop in $\mathcal{A}_1 \sqcup\!\sqcup_R \mathcal{A}_2$ using t: since $\overline{C} \subsetneq \overline{D}$ and by (i) – proof of Proposition 2 – all states accessible from $(p', q', \overline{D})$ are of the form $(r, s, \overline{B})$, with $\overline{D} \subseteq \overline{B}$.

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

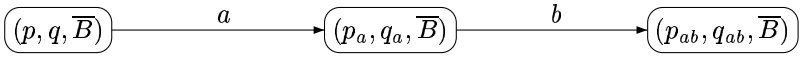
$$m = m_0 t_1 m_1 t_2 \dots t_n m_n$$

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

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

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

By Lemma 7, it remains to prove that the L_i's are commutative languages. Let $B \subseteq A$, we prove that the monoid of transitions generated by $(Q_{\overline{B}}, E_{\overline{B}})$ is commutative. Let $r = (p, q, \overline{B})$, $r_a = (p_a, q_a, \overline{B})$ and $r_{ab} = (p_{ab}, q_{ab}, \overline{B})$ three states of $Q_{\overline{B}}$ such that there exist transitions $t_a = (r, a, r_a)$ and $t_{ab} = (r_a, b, r_{ab})$ in $E_{\overline{B}}$.



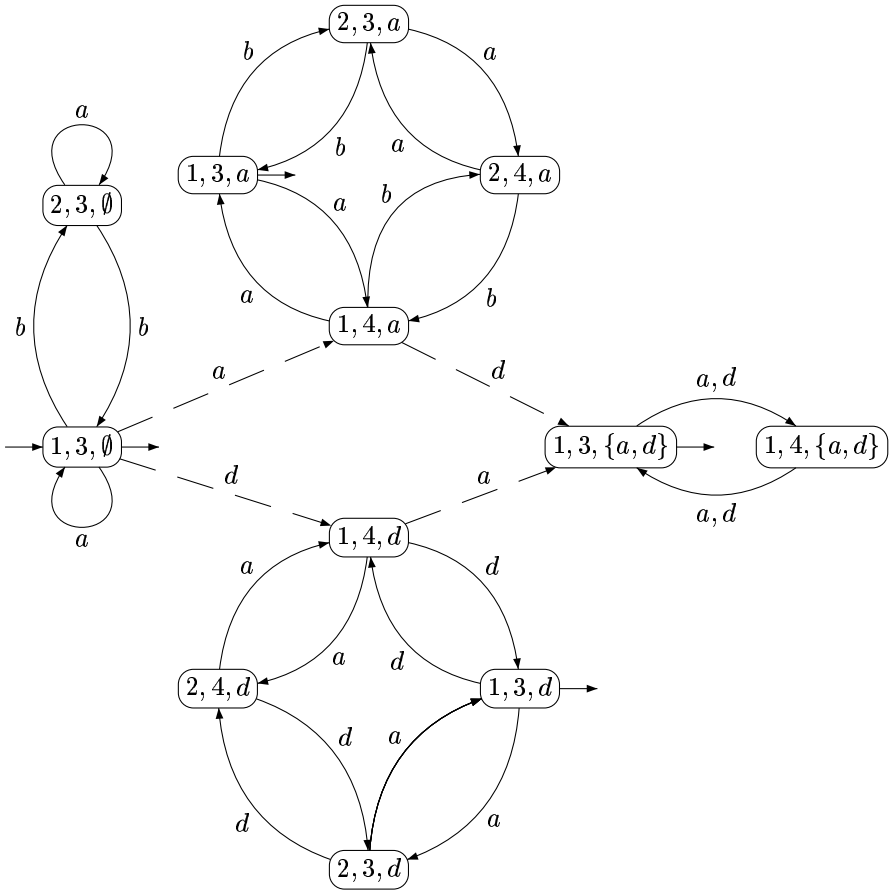
With the notation of Proposition 2, the following cases occur:

- $t_a, t_{ab} \in G_1$. Since $\mathcal{A}_1$ is minimal and since $L(\mathcal{A}_1)$ is commutative, the transition monoid of $\mathcal{A}_1$ 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_b belong to G_1, $(a, \overline{B}) \subseteq R$ and $(b, \overline{B}) \subseteq B$. Consequently, $(r, b, (p_b, q, \overline{B}))$ and $((p_b, q, \overline{B}), a, r_{ab})$ are in $G_1 \cap E_{\overline{B}}$. 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 $(r, b, (p, q_{ab}, \overline{B})) \in G_2 \cap E_{\overline{B}}$ and $((p, q_{ab}, \overline{B}), a, r_{ab}) \in G_1 \cap E_{\overline{B}}$. 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, $r \cdot ba \subseteq r \cdot ab$. Therefore, the monoid of transitions generated by $(Q_{\overline{B}}, E_{\overline{B}})$ is commutative, which concludes the proof.

□

Example 10 *Let* $L_1 = \{u \in \{a, b\}^* \mid |u|_b$ *is even*$\}$ *and* $L_2 = \{u \in \{a, d\}^* \mid |u|$ *is even*$\}$ *and* $R = \{(a, d), (b, a)\}$. *Then* $L_1 \sqcup\!\sqcup_R L_2$ *is given by the following finite automaton (transition of* $E \setminus \cup_{B \in A} E_{\overline{B}}$ *are represented by dash arrows):*



It follows that $L_1 \sqcup\!\sqcup_R L_2 = L_1aL_2 \cup L_1aL_4dL_5 \cup L_1dL_6 \cup L_1dL_7aL_5$, *where*

$$
\begin{aligned}
L_3 =& \{u \in \{a,b\}^* \mid |u|_b \textit{ is even and } |u|_a \textit{ is odd }\}\\
L_4 =& \{u \in \{a,b\}^* \mid |u|_b \textit{ is even and } |u|_a \textit{ is even}\}\\
L_5 =& \{u \in \{a,d\}^* \mid |u| \textit{ is even}\}\\
L_6 =& \{u \in \{a,d\}^* \mid |u|_d \textit{ is odd and } |u|_a \textit{ is even}\}\\
L_6 =& \{u \in \{a,d\}^* \mid |u|_d \textit{ is even and } |u|_a \textit{ is even}\}.
\end{aligned}
$$

We can now prove the main result.

Theorem 11 *The class* Pol$\mathcal{C}$ *is closed under union, intersection, left and right quotient, conjugacy and semi-commutation.*

PROOF. Since the class of commutative languages is a variety of languages, by Theorem 5, Pol$\mathcal{C}$ is a positive variety of languages and thus is closed under union, intersection and left and right quotients.

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

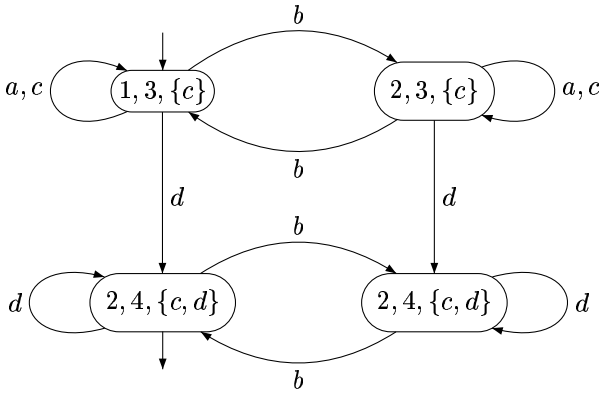
$$\mathrm{Conj}(L) = \bigcup_{0 \leq i \leq k} L(\mathcal{A}_{i\,q_i,F_i}) a_{i+1} L(\mathcal{A}_{i+1}) \ldots a_k L(\mathcal{A}_k) L(\mathcal{A}_1) a_1 \ldots a_i L(\mathcal{A}_{i\,p_i,q_i})$$

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 Lemma 7 and 8, $\mathrm{Conj}(L) \in A^*\mathrm{Pol}\mathcal{C}$. 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 Lemma 6 and 9 and Proposition 1, Pol$\mathcal{C}$ is also closed under semi-commutation. □

Let us notice that the proof is constructive.

Example 12 *Let $L = (a^*ba^*ba^*)^*$ the set of words on $\{a, b\}$ with an even number of b's. Let $R = \{(b, c), (b, d), (a, c)\}$ the semi-commutation relation over $\{a, b, c, d\}$. Then $R^*(Lc^*dd^*) = L \sqcup\!\sqcup_R c^*dd^*$. Then, using example 3, $R^*(Lc^*dd^*)$ is given by the following automaton:*



Using Lemma 9, one obtains the following decompostion:

$$R^*((a^*ba^*ba^*)^*c^*dd^*) = K_1 dL_1 \cup K_2 dL_2$$

where $K_1 = (\{a,c\}^*b\{a,c\}^*b\{a,c\}^*)^*$, $K_2 = (\{a,c\}^*b\{a,c\}^*b\{a,c\}^*)^*b\{a,c\}^*$, $L_1 = (d^*bd^*bd^*)^*$ *and* $L_2 = (d^*bd^*bd^*)^*bd^*$ *are commutative closed.*

6 Conclusion

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

Moreover, in [BMT01] the question of finding other subclasses of regular languages which is closed under union, intersection, product, semi-commutation rewriting and conjugacy was opened. We showed that 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 [GP03b] where the maximal positive variety closed under the shuffle operation is exhibited.

References

[AAB99] 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.

[ABJ98] 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.

[AJNd03] Parosh Aziz Abdulla, Bengt Jonsson, Marcus Nilsson, and Julien d'Orso. Algorithmic improvements in regular model checking. In *CAV'03*, LNCS, 2003.

[Ber79] Jean Berstel. *Transductions and Context-Free Languages.* B.G. Teubner, Stuttgart, 1979.

[BG96] 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.

[BMT01] A. Bouajjani, A. Muscholl, and T. Touili. Permutation rewriting and algorithmic verification. In *LICS'01*, IEEE Computer Society, pages 399–408, 2001.

[Brz76] J.A. Brzozowski. Hierarchies of aperiodic languages. *Informatique théorique et Application/Theoretical Informatics and Applications*, 10:33–49, 1976.

[BS73] J.A. Brzozowski and I. Simon. Characterizations of locally testable languages. *Discrete Mathematics*, 4:243–271, 1973.

[BW98] 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.

[DR95] V. Diekert and G. Rozenberg, editors. *Book of Traces.* World Scientific, Singapore, 1995.

[ES98] Z. Esik and I. Simon. Modeling literal morphisms by shuffle. *Semigroup Forum*, 56:225–227, 1998.

[GP03a] A. Cano Gomez and J.-E. Pin. On a conjecture of schnoebelen,. In *DLT'03*, Lecture Notes in Computer Science, 2003.

[GP03b] A. Cano Gomez and J.-E. Pin. Shuffle on positive varieties of languages. *Theoretical Computer Science*, 2003. to appear.

[GW94] P. Godefroid and P. Wolper. A partial approach to model checking. *Information and Computation*, 110(2):305–326, 1994.

[Héa03] P.-C. Héam. Some complexity results for polynomial rational expressions. *TCS: Theoretical Computer Science*, 299, 2003.

[HU80] J. Hopcroft and J. Ullman. *Introduction to automata theory, languages, and computation.* Addison-Wesley, 1980.

[LDG+02] X. Leroy, D. Doligez, J. Garrigue, D. Rémy, and J. Vouillon. *The Objective Caml system, release 3.06.* Inria, 2002.

[LS98] 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.

[Per78] J.-F. Perrot. Variété de langages et opérations. *Theoretical Computer Science*, 7:197–210, 1978.

[Pin84] J.-E. Pin. *Varieties of formal languages.* Foundations of Computer Science, 1984.

[PW97] Jean-Eric Pin and Pascal Weil. Polynomial closure and unambiguous product. *Theory Comput. Systems*, 30:1–39, 1997.

[Sch99] Ph. Schnoeboelen. Decomposable regular languages and the shuffle operator. *EATCS Bull.*, 67:283–289, 1999.

[Str85] H. Straubing. Finite semigroups varieties of the form $\mathbf{V}*\mathbf{D}$. *Journal of Pure and Applied Algebra*, 36:53–94, 1985.

[Thé81] D. Thérien. Classification of finite monoids: the language approach. *Theoretical Computer Science*, 14:195–208, 1981.

[Tho82] W. Thomas. Classifying regular events in symbolic logic. *Journal of Computer and System Science*, 25:360–375, 1982.

[Tou01] T. Touili. Regular model checking using widening techniques. In *1st Vepas Workshop*, volume 50 of *Electronic Notes in TCS*, 2001.

[TT02] P. Tesson and D. Thérien. Diamonds are forever: the variety da. In *International Conference on Semigroups, Algorithms, Automata and Languages*, 2002.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399