INFINITY 2003 Preliminary Version

The affine hull of a binary automaton is computable in polynomial time

Jérôme Leroux [1]

Laboratoire Spécification et Vérification,
CNRS UMR 8643 & ENS de Cachan,
61 av. du Président Wilson,
94235 Cachan cedex, France

Abstract

We present the class of binary automaton, a new representation for the subsets of $\mathbb{N}^m$ that naturally extends the NDD ([25], [10]). We prove that the affine hull of the set of vectors represented by a binary automaton is computable in polynomial time. As application, we show that the set of place invariants [11] of a counter system (an extension of the Broadcast Protocols [16], [13], [12], the Reset/Transfer Petri Nets [15],[11] and the linear systems [18]), is computable in polynomial time.

Key words: Presburger, affine hull, NDD, binary automaton, place invariant.

1 Introduction.

An infinite state system is by definition a system whose the reachability set may contain an infinite number of configurations. For such a system it is therefore impossible to represent this set just by enumerating its elements. To overcome this problem, symbolic model checkers implements symbolic representations adapted to the structure of the infinite sets represented. For instance the tool TREX [24], [4] implements SRE [1] for representing infinite set of words closed under the sub-word relation [2], the tool BABYLON [5], [3] uses CST [14] for representing upward closed sets [19], and so on.

In this article, we are interested in the NDD [25] [26] (a.k.a DFA [10], [23], [21], [20]), the symbolic representation used by the symbolic model checkers FAST [6], [17] and LASH [22] to represent Presburger definable subsets of $\mathbb{N}^m$ by a finite automaton.

These tools use NDD at different levels of their implementation:

[1] Email: `leroux@lsv.ens-cachan.fr`

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: `www.elsevier.nl/locate/entcs`

- To represent the input system. In fact, these tools analyze counter systems such that each action is represented by a NDD that represents the set of configurations which the action can be fired from (Lash requires that this set is convex [18]) and by an affine function that relates the value of the counters before and after the action is fired.
- To represent the property that the input system must ensure. This property is in fact a NDD that represents the set of bad configurations that the system must avoid.
- To accelerate transitions ([18],[9],[8]). Under some algebraic conditions (often met [18], [6]), a NDD that represents the transitive closure of a sequence of actions can be computed.
- To represent subsets of the reachability set.

In these tools, no structural analysis of the counter systems is done to simplify the computation of the reachability set. However, in the case of Petri nets [7], or more generally self/modifying Petri Nets [11], the place invariants is a useful tool:

- for reducing the number of true counters of the system [14] (recall that in practice, we are limited in the tools Fast and Lash by a dozen of counters).
- for helping the termination of the reachability set. In fact, we can compute an over-approximation of the configurations which a bad configuration can be reached from ([14]) and intersect the computation of the reachability set with this over-approximation.

We proved in this article an interesting result: place invariants can be extended to the class of counters systems used in Fast and Lash and remain computable in polynomial time. To obtain this result we prove an other result: the affine hull of the set of vectors accepted by a NDD is computable in polynomial time.

Our contributions

- We introduce the class of binary automaton, a natural extension of the NDD. Whereas NDD corresponds to a word by word representation, we show that the class of binary automata is a bit by bit representation.
- We prove that the affine hull of the set of vectors represented by a binary automaton is computable in polynomial time.
- We introduce the class of counter systems, an extension of the class of Broadcast protocols, Reset/Transfer Petri Nets and the linear systems.
- We extend the definition of place invariants to this class and proved that they remain computable in polynomial time.
- We show the link between place invariants and the affine hull of the reachability relation.

Plan of the paper

In section 2, some usual notations are recalled and in section 3 we explain why all the operations used in this article over the affine spaces can be done just by using a Gauss elimination algorithm. In section 4, the binary automata are presented and in the following section, an algorithm for computing the affine hull of the set of vectors accepted by a binary automaton is provided. Finally, in section 6, this affine hull is proved to be useful for computing the place invariants of an effective counter system, a model introduced in this last section.

2 Notations.

The cardinal of a finite set X is written $\mathrm{card}(X)$.

The set of rationals, integers and positive integers are respectively written $\mathbb{Q}$, $\mathbb{Z}$ and $\mathbb{N}$. The set of vectors with m components in a set X is written X^m. The i-th component of a vector $x \in X^m$ is written $x_i \in X$; we have $x = (x_1, \dots, x_m)$. For any vector $v, v' \in \mathbb{Q}^m$ and for any $t \in \mathbb{Q}$, we define $t.v$ and $v + v'$ in $\mathbb{Q}^m$ by $(t.v)_i = t.v_i$ and $(v + v')_i = v_i + v'_i$.

A vector subspace V of $\mathbb{Q}^m$ is a non empty subset $V \subseteq \mathbb{Q}^m$ such that for every $v, v' \in V$ and for every $t, t' \in \mathbb{Q}$, we have $t.v + t'v' \in V$. An affine subspace A of $\mathbb{Q}^m$ is a subset of $\mathbb{Q}^m$ (eventually empty) such that for every $a, a' \in A$ and for every $t, t' \in \mathbb{Q}$ such that $t + t' = 1$, we have $t.a + t'.a' \in A$. As any intersection of affine spaces remains an affine space, the affine hull of a subset X of $\mathbb{Q}^m$ is well defined as the least affine space that contains X; this affine space is written $\mathrm{aff}(X)$. Recall that for any set X, there exists a finite subset $X' \subseteq X$ such that $\mathrm{aff}(X) = \mathrm{aff}(X')$. In particular for any affine space A, there exists a finite subset $X' \subseteq A$ such that $A = \mathrm{aff}(X')$. Recall that the dimension of an affine space A is the minimal integer written $\dim(A) \in \{-1, \dots, m\}$ such that there exists a finite subset $X \subseteq A$ satisfying $\mathrm{aff}(X) = A$ and $\mathrm{card}(X) - 1 = \dim(A)$. Such a subset X is called a basis of A.

The set of square matrices over $\mathbb{Q}$ is written $\mathcal{M}_m(\mathbb{Q})$. A function $f : \mathbb{Q}^m \to \mathbb{Q}^m$ is said affine if these exists a square matrix $M \in \mathcal{M}_n(\mathbb{Q})$ and a vector $v \in \mathbb{Q}^m$ such that $f(x) = M.x + v$ for any $x \in \mathbb{Q}^m$. A function $f : \mathbb{Q}^m \to \mathbb{Q}$ is said linear if there exists a vector $v \in \mathbb{Q}^m$ such that for every $x \in \mathbb{Q}^m$, we have $f(x) = \sum_{i=1}^{m} v_i.x_i$. The set of linear functions is therefore a vector space isomorphic to $\mathbb{Q}^m$.

The set of words over a finite alphabet Σ is written Σ^*. The concatenation of two words σ and σ' in Σ^* is written $\sigma.\sigma'$. The empty word in Σ^* is written ϵ.

A finite automaton $\mathcal{A}$ is a tuple $\mathcal{A} = (Q, \Sigma, \Delta, Q_0, F)$; Q is the finite set of states, Σ is the finite alphabet, $\Delta \subseteq Q \times \Sigma \times Q$ is the transition relation, $Q_0 \subseteq Q$ is the set of initial states and $F \subseteq Q$ is the set of final states. A finite automaton $\mathcal{A}$ is said complete and deterministic if the set Q_0 is reduced to one element $Q_0 = \{q_0\}$ and if there exists a function $\delta : Q \times \Sigma \to Q$ such that $\Delta = \{(q, \delta(q, a));\ q \in Q;\ a \in \Sigma\}$. A path P in a finite automaton $\mathcal{A}$ from a state q to a state q' is a finite sequence $q = q_0, (q_0, a_1, q_1), q_1, \dots, (q_{n-1}, a_n, q_n), q_n = q'$ with $n \geq 0$ such that (q_{i-1}, a_i, q_i) is a transition in Δ. The label of P is the word $\sigma = a_1 \dots a_n \in \Sigma^*$. Such a path is written $q \xrightarrow{\sigma} q'$.

A binary relation $\mathcal{R}$ over a set X is a subset of $X \times X$; If a tuple $(x, x') \in \mathcal{R}$, we write $x\mathcal{R}x'$. The identity relation over X is written I_X or just I and it is defined by xIx' iff $x = x'$.

3 About the complexity of the affine spaces.

In this article, we often consider the affine hull of the union of two affine spaces, the image of an affine space by an affine function and we often test the inclusion of two affine spaces. We study briefly the complexity of these operations by representing an affine space A by a finite set X of at most $m+1$ vectors in A such that $A = \mathrm{aff}(X)$.

First remark that if A and A' are two affine spaces respectively represented by X and X', then by using a Gauss elimination, we can compute in polynomial time a finite subset P of at most $m + 1$ vectors in $X \cup X'$ such that $\mathrm{aff}(A \cup A') = \mathrm{aff}(P)$.

Moreover, the affine space $f(A)$ defined as the image of an affine space $A = \mathrm{aff}(X)$ by an affine function $f(x) = M.x + v$ where $M \in \mathcal{M}_m(\mathbb{Q})$ and $v \in \mathbb{Q}^m$ is represented by the set $f(X)$.

Finally, remark that by using a Gauss elimination, for any two sets X and X' of at most $m + 1$ vectors in $\mathbb{Q}^m$, we can check in polynomial time if $\mathrm{aff}(X) \subseteq \mathrm{aff}(X')$.

4 Binary automata.

Binary automata are proved to be a natural extension of NDD by providing a bit by bit representation of vectors of $\mathbb{N}^m$ whereas NDD is rather a word by word representation.

Recall that a NDD is a deterministic and complete automaton over the alphabet $\Sigma_r = \{0, \dots, r - 1\}$ where $r \geq 2$ is called the base of decomposition such that the length of any word accepted by the automaton can be divided by m. This representation is really well adapted for representing subsets of $\mathbb{N}$ because any $x \in \mathbb{N}$ can be represented as a finite sequence of "bits" in Σ_r. A vector $x \in \mathbb{N}^m$ is also easily represented by a word $b_1 b_2 \dots b_{n.m}$ such that each x_i is represented by $b_i b_{i+m} \dots b_{i+(n-1).m}$. With such a representation, we remark that the vectors x' represented by $b_1 b_2 \dots b_{n.m}$ is related to the vectors x represented by $b_{m+1} b_{m+1} \dots b_{n.m}$, by the following formula:

$$x' = r.x + (b_1, \dots, b_m)$$

For this reason, we claim that NDD is a word by word representation that needs to read m-bits by m-bits to compute the vector represented. This restriction seems to be in contradiction with a polynomial time computation of the affine hull of the set of vectors represented by a binary automaton due to the exponential size of $\mathrm{card}(\Sigma_r)^m$.

To overcome this limitation, we propose to associate to any word $\sigma \in \Sigma_r^*$ a vector in $\mathbb{N}^m$ without any restriction on the length of σ.

Let us introduce the affine function $\lambda_\sigma : \mathbb{Q}^m \rightarrow \mathbb{Q}^m$ defined for any word $\sigma \in \Sigma_r^*$ and for any bit $b \in \Sigma_r$ by the following induction:

$$\begin{cases} \lambda_b(x_1, \dots, x_m) = (x_2, \dots, x_m, r.x_1 + b) \\ \lambda_{b.\sigma} = \lambda_b \circ \lambda_\sigma \end{cases}$$

Definition 4.1 A binary representation of a vector $x \in \mathbb{N}^m$ is a word $\sigma \in \Sigma_r^*$ such that $x = \rho(\sigma)$ where $\rho(\sigma) = \lambda_\sigma(0, \dots, 0)$.

Definition 4.2 A binary automaton $\mathcal{A}$ is a finite automaton over the alphabet Σ_r. The set of vectors represented by a binary automaton $\mathcal{A}$ is the subset $\rho(\mathcal{L}(\mathcal{A}))$ of $\mathbb{N}^m$.

We can now give the definition of NDD with our notation that is equivalent to the original definition just because for any word $b_1 \dots b_m$ and for any $x \in \mathbb{Q}^m$, we have $\lambda_{b_1 \dots b_m}(x) = r.x + (b_1, \dots, b_m)$.

Definition 4.3 [[9]] A NDD is a deterministic and complete binary automaton such that m divides the length of any accepted words.

Remark 4.4 NDD also allow to represent vectors in $\mathbb{Z}^m$ by considering a 2-complement representation. We have not considered this special feature to simplify the presentation of this article. However, the extension can be easily done by considering a 2-complement representation or by remarking that for any vector $x \in \mathbb{Z}^m$, there exists a word $\sigma \in \Sigma_r^*$ such that $x = \lambda_\sigma(-\frac{1}{r}, \dots, -\frac{1}{r})$. It seems that this later representation, more algebraic than the 2-complement one, is also more concise (this is a work in progress).

Obviously, as proved by the following proposition, the binary automata and the NDD represent the same subsets of $\mathbb{N}^m$.

Proposition 4.5 *A subset $X \subseteq \mathbb{N}^m$ is representable by a binary automaton if and only if X is representable by a NDD.*

Proof. First remark that if a subset $X \subseteq \mathbb{N}^m$ is represented by a NDD, then X is represented by a binary automaton just because a NDD is a binary automaton. For the converse, let us consider a subset $X \subseteq \mathbb{N}^m$ represented by a binary automaton $\mathcal{A}$.

We first prove that we can assume that $\mathcal{L}(\mathcal{A}).0^* = \mathcal{L}(\mathcal{A})$. As $\mathcal{L}(\mathcal{A})$ and 0^* are two regular languages, the language $\mathcal{L}(\mathcal{A}).0^*$ is also regular. So, we have just to prove that $\rho(\mathcal{L}.0^*) = \rho(\mathcal{L})$ for any subset $\mathcal{L} \subseteq \Sigma_r^*$. From the inclusion $\mathcal{L} \subseteq \mathcal{L}.0^*$, we deduce $\rho(\mathcal{L}) \subseteq \rho(\mathcal{L}.0^*)$. Let us prove the converse inclusion. Let $x \in \rho(\mathcal{L}.0^*)$. There exists $i \geq 0$ and $\sigma \in \mathcal{L}$ such that $x = \rho(\sigma.0^i)$. From $x = \rho(\sigma.0^i) = \lambda_{\sigma.0^i}(0, \dots, 0) = \lambda_\sigma(\lambda_0^i(0, \dots, 0)) = \lambda_\sigma(0, \dots, 0) = \rho(\sigma) \in \rho(\mathcal{L})$, we deduce $\rho(\mathcal{L}.0^*) \subseteq \rho(\mathcal{L})$. Hence, we can assume that $\mathcal{L}(\mathcal{A}).0^* = \mathcal{L}(\mathcal{A})$.

Remark that we can also assume that $\mathcal{A} = (Q, \Sigma_r, \delta, \{q_0\}, F)$ is deterministic and complete. We denote by $[i] \in \{0, \dots, m-1\}$ the remainder of the Euclide division of i by m. Let us consider the binary automaton $\mathcal{A}' = (Q', \Sigma_r, \delta', \{q_0'\}, F')$ defined by

$$\begin{cases} Q' = Q \times \{0, \dots, m-1\} \\ \delta'((q, i), b) = (\delta(q, b), [i+1]) \\ q_0' = (q_0, 0) \\ F' = F \times \{0\} \end{cases}$$

Remark that $\mathcal{A}'$ is a NDD. So we have just to prove that $\rho(\mathcal{L}(\mathcal{A})) = \rho(\mathcal{L}(\mathcal{A}'))$. By construction, we have $\mathcal{L}(\mathcal{A}') \subseteq \mathcal{L}(\mathcal{A})$. Hence, we have $\rho(\mathcal{L}(\mathcal{A}')) \subseteq \rho(\mathcal{L}(\mathcal{A}))$. Let us prove the converse inclusion. Let $x \in \rho(\mathcal{L}(\mathcal{A}))$. There exists $\sigma \in \mathcal{L}(\mathcal{A})$ such that $x = \rho(\sigma)$. As $\mathcal{L}(\mathcal{A}).0^* = \mathcal{L}(\mathcal{A})$, we have $\sigma.0^{m-[|\sigma|]} \in \mathcal{L}(\mathcal{A})$. As the length of $\sigma.0^{m-[|\sigma|]}$ can be divided by m, we have proved that $\sigma.0^{m-[|\sigma|]} \in \mathcal{L}(\mathcal{A}')$. From $x = \rho(\sigma) = \lambda_\sigma(0, \dots, 0) = \lambda_\sigma(\lambda_0^{m-[\sigma]}(0, \dots, 0)) = \rho(\sigma.0^{m-[\sigma]}) \in \rho(\mathcal{L}(\mathcal{A}'))$, we deduce $\rho(\mathcal{L}(\mathcal{A})) \subseteq \rho(\mathcal{L}(\mathcal{A}'))$. Therefore $\mathcal{A}'$ is a NDD that represents X. $\square$

5 Affine hull of the set of vectors represented by a binary automaton.

The affine hull of the set of vectors represented by a binary automaton $\mathcal{A}$ is proved to be computable in polynomial time.

This affine hull is obtained by labeling the states of $\mathcal{A}$ by some affine spaces: an affine covering.

Definition 5.1 An affine covering of a binary automaton $\mathcal{A} = (Q, \Sigma_r, \Delta, Q_0, F)$ is a sequence of affine spaces $(A_q)_{q \in Q}$ of $\mathbb{Q}^m$ such that:

- for every state $q_f \in F$ we have $(0, \dots, 0) \in A_{q_f}$, and
- for every path $q \xrightarrow{\sigma} q'$, we have $\lambda_\sigma(A_{q'}) \subseteq A_q$.

Any binary automaton $\mathcal{A}$ has at least one affine covering $(A_q)_{q \in Q}$ because $A_q = \mathbb{Q}^m$ for every $q \in Q$ in an affine covering. Therefore, the following sequence $(\mathrm{cov}(\mathcal{A})_q)_{q\in Q}$ is defined.

$$\mathrm{cov}(\mathcal{A})_q = \bigcap_{\substack{(A_p)_{p\in Q}\\ \text{affine covering}}} A_q$$

Lemma 5.2 *For any binary automaton $\mathcal{A}$, the sequence $(\mathrm{cov}(\mathcal{A})_q)_{q\in Q}$ is an affine covering of $\mathcal{A}$.*

Proof. Let us note S the set of affine covering of $\mathcal{A}$. As any intersection of affine spaces is an affine space, $(\mathrm{cov}(\mathcal{A})_q)_{q\in Q}$ is a sequence of affine spaces. Moreover, for every $q_f \in F$, we have $(0,\dots,0) \in \mathrm{cov}(\mathcal{A})_{q_f}$ because for every $(A_p)_{p\in Q}$ in S, we have $(0,\dots,0) \in A_{q_f}$. Now, let us consider a path $q \xrightarrow{\sigma} q'$. As λ_σ is a one to one function, we have $\lambda_\sigma(\mathrm{cov}(\mathcal{A})_{q'}) = \lambda_\sigma(\bigcap_{(A_p)_{p\in Q}\in S} A_{q'}) = \bigcap_{(A_p)_{p\in Q}\in S} \lambda_\sigma(A_{q'}) \subseteq \bigcap_{(A_p)_{p\in Q}\in S} A_q = \mathrm{cov}(\mathcal{A})_q$. □ □

As proved by the previous lemma, the sequence $(\mathrm{cov}(\mathcal{A})_q)_{q\in Q}$ is an affine covering.

Definition 5.3 The least affine covering of a binary automaton $\mathcal{A}$ is the sequence of affine spaces $(\mathrm{cov}(\mathcal{A})_q)_{q\in Q}$.

From the least affine covering of a binary automaton, the following proposition 5.4 proves that we can compute the affine hull of the set of vectors accepted by a binary automaton.

Proposition 5.4 *For any binary automaton $\mathcal{A}$, we have:*

$$\mathrm{aff}(\rho(\mathcal{L}(\mathcal{A}))) = \mathrm{aff}(\bigcup_{q_0\in Q_0} \mathrm{cov}(\mathcal{A})_{q_0})$$

Proof. For every $q \in Q$, we consider the set $X_q = \{\rho(\sigma);\ \exists q \xrightarrow{\sigma} q_f;\ q_f \in F\} \subseteq \mathbb{N}^m$. We first prove that $\mathrm{cov}(\mathcal{A})_q = \mathrm{aff}(X_q)$ for every $q \in Q$.

To prove that $\mathrm{cov}(\mathcal{A})_q \subseteq \mathrm{aff}(X_q)$ for every $q \in Q$, we show that $(\mathrm{aff}(X_q))_{q\in Q}$ is an affine covering. For every $q_f \in F$, we have $(0,\dots,0) = \rho(\epsilon) \in X_{q_f}$. Let us consider a path $q \xrightarrow{\sigma} q'$ and a vector $x \in X_{q'}$. There exists a state $q_f \in F$ and a path $q' \xrightarrow{\sigma'} q_f$ such that $x = \rho(\sigma')$. Remark that $\lambda_\sigma(x) = \lambda_\sigma(\lambda_{\sigma'}(0,\dots,0)) = \rho(\sigma\sigma')$. As $q \xrightarrow{\sigma.\sigma'} q_f$ is a path, we have $\lambda_\sigma(x) = \rho(\sigma\sigma') \in X_q$. Hence $\lambda_\sigma(X_{q'}) \subseteq X_q$. Therefore $\mathrm{aff}(\lambda_\sigma(X_{q'})) \subseteq \mathrm{aff}(X_q)$. As λ_σ is an affine function, we have $\mathrm{aff}(\lambda_\sigma(X_{q'})) = \lambda_\sigma(\mathrm{aff}(X_{q'}))$. We have proved that $(\mathrm{aff}(X_q))_{q\in Q}$ is an affine covering. By minimality of the sequence $(\mathrm{cov}(\mathcal{A})_q)_{q\in Q}$, we have proved that $\mathrm{cov}(\mathcal{A})_q \subseteq \mathrm{aff}(X_q)$ for every $q \in Q$.

To prove that $\mathrm{aff}(X_q) \subseteq \mathrm{cov}(\mathcal{A})_q$ for every $q \in Q$, we show that $X_q \subseteq \mathrm{cov}(\mathcal{A})_q$ for every $q \in Q$. Let $x \in X_q$. There exists a path $q \xrightarrow{\sigma} q_f$ with $q_f \in F$ such that $x = \rho(\sigma)$. As $(\mathrm{cov}(\mathcal{A})_q)_{q\in Q}$ is an affine covering, we have

$(0,\ldots,0) \in \mathrm{cov}(\mathcal{A})_{q_f}$ and $\lambda_\sigma(\mathrm{cov}(\mathcal{A})_{q_f}) \subseteq \mathrm{cov}(\mathcal{A})_q$. Therefore $x = \rho(\sigma) = \lambda_\sigma(0,\ldots,0) \in \mathrm{cov}(\mathcal{A})_q$. Hence $X_q \subseteq \mathrm{cov}(\mathcal{A})_q$. As $\mathrm{cov}(\mathcal{A})_q$ is an affine space, we have proved that $\mathrm{aff}(X_q) \subseteq \mathrm{cov}(\mathcal{A})_q$.

Hence, for every $q \in Q$, we have $\mathrm{aff}(X_q) = \mathrm{cov}(\mathcal{A})_q$. Now, just remark that:

$$\begin{aligned}\mathrm{aff}(\rho(\mathcal{L}(\mathcal{A}))) &= \mathrm{aff}(\bigcup_{q_0 \in Q_0} X_{q_0}) \\ &= \mathrm{aff}(\bigcup_{q_0 \in Q_0} \mathrm{aff}(X_{q_0})) \\ &= \mathrm{aff}(\bigcup_{q_0 \in Q_0} \mathrm{cov}(\mathcal{A})_{q_0})\end{aligned}$$

Hence, we are done. □

Algorithm 1 Compute the affine hull of the set of vectors accepted by a binary automaton.

1: **Input:** A binary automaton $\mathcal{A} = (Q, \Sigma_r, \Delta, Q_0, F)$.
2: **Output:** The affine space $\mathrm{aff}(\rho(\mathcal{L}(\mathcal{A})))$.
3:
4: Let $(A_q)_{q\in Q}$ be the sequence defined by $A_q = \begin{cases} \{(0,\ldots,0)\} & \text{if } q \in F \\ \emptyset & \text{otherwise} \end{cases}$
5: **while** there exists $q \xrightarrow{b} q'$ in Δ such that $\lambda_b(A_{q'}) \not\subseteq A_q$ **do**
6: $\quad A_q \Leftarrow \mathrm{aff}(A_q \cup \lambda_b(A_{q'}))$
7: **return** $\mathrm{aff}(\bigcup_{q_0 \in Q_0} A_{q_0})$

By using a fix-point algorithm, we compute the affine covering of a binary automaton in polynomial time as proved in the following theorem.

Theorem 5.5 *The algorithm 1 computes the affine hull of the set of vectors accepted by a binary automaton $\mathcal{A}$ in polynomial time.*

Proof. We first prove that line 6 is executed at most $(m+1).\mathrm{card}(Q)$ times. In fact, remark that each times line 6 is executed, the dimension of the affine space A_q strictly increase. As the dimension of an affine space of $\mathbb{Q}^m$ is in the finite set $\{-1,\ldots,m\}$, we have the result.

To prove that the complexity of the algorithm is polynomial, we prove that the representations of the affine spaces computed by the algorithm are some finite subsets of at most $m+1$ vectors in the set $\{0,\ldots,r^{2.\mathrm{card}(Q)}-1\}^m$.

Let us prove that the representations X of the affine spaces A computed by the algorithm are some finite subset of at most $m+1$ vectors in $\{\rho(\sigma);\ \sigma \in \Sigma_r^{\leq (m+1).\mathrm{card}(Q)}\}$. To do so, we denote by $(A_q^i)_{q\in Q}$ the sequence of affine spaces $(A_q)_{q\in Q}$ on line 5 in function of the number of times $i \geq 0$ that the "while" loop has been executed. We denote by X_q^i the representation of A_q^i. Let us prove by induction over $i \geq 0$ that for every $q \in Q$, X_i^q is a finite subset of at most $m+1$ vectors in $\{\rho(\sigma);\ \sigma \in \Sigma_r^{\leq i}\}$. Remark that the induction is true at rank $i = 0$. Assume the induction true at rank $i \geq 0$ and let us prove that the induction remains true at rank $i+1$. Remark that there exists $q \xrightarrow{b} q'$ in Δ such that for every $p \in Q/\{q\}$, we have $A_p^{i+1} = A_p^i$ and $A_q^{i+1} = \text{aff}(A_q^i \cup \lambda_b(A_{q'}^i))$. Therefore, for every $p \neq q$, we have $X_p^{i+1} = X_p^i$ and $X_q^{i+1} \subseteq X_q^i \cup \lambda_b(X_{q'}^i)$. By using the induction at rank i, we obtain $X_p^{i+1} \subseteq \{\rho(\sigma);\ \sigma \in \Sigma_r^{\leq i+1}\}$ for every $p \in Q$. There, we have proved the induction at rank $i+1$. We have proved the induction. In particular, as $i \leq (m+1).\text{card}(Q)$, we have proved that the representations X of the affine spaces computed by the algorithm are finite subsets of at most $m+1$ vectors in $\{\rho(\sigma);\ \sigma \in \Sigma_r^{\leq (m+1).\text{card}(Q)}\}$.

Now let us prove that $\{\rho(\sigma);\ \sigma \in \Sigma_r^{\leq (m+1).\text{card}(Q)}\} \subseteq \{0, \ldots, r^{2.\text{card}(Q)} - 1\}^m$. Let us consider $\sigma \in \Sigma_r^{\leq (m+1).\text{card}(Q)}$. There exists $i \geq 0$ such that $\sigma.0^i$ is a word of length $m.2.\text{card}(Q)$. Therefore $\rho(\sigma) \in \{0, \ldots, r^{2.\text{card}(Q)} - 1\}^m$.

We have proved that the affine spaces computed by the algorithm are represented by at most $m+1$ vectors in $\{0, \ldots, r^{2.\text{card}(Q)}\}^m$. The complexity of the algorithm is therefore polynomial.

To prove the correctness of the algorithm, we first show that the following assertion is an invariant: " $A_q \subseteq \text{cov}(\mathcal{A})_q$ for every $q \in Q$ and $(0, \ldots, 0) \in A_{q_f}$ for every $q_f \in F$". Remark that just after the execution of line 4, the assertion is true. So assume the assertion true before the executing of line 6 and let us prove that the assertion remains true just after. Let $q \xrightarrow{b} q'$ be a transition in Δ. As $(\text{cov}(\mathcal{A})_q)_{q \in Q}$ is an affine covering, we have $\lambda_b(\text{cov}(\mathcal{A})_{q'}) \subseteq \text{cov}(\mathcal{A})_q$. From the assertion we deduce $\lambda_b(A_{q'}) \subseteq \text{cov}(\mathcal{A})_q$. Moreover the assertion also gives $A_q \subseteq \text{cov}(\mathcal{A})_q$. Hence $A_q \cup \lambda_b(A_{q'}) \subseteq \text{cov}(\mathcal{A})_q$. By considering the affine hull of the previous inclusion, we obtain $\text{aff}(A_q \cup \lambda_b(A_{q'})) \subseteq \text{cov}(\mathcal{A})_q$. Therefore the assertion is an invariant of the algorithm.

In particular, on line 7 we have $A_q \subseteq \text{cov}(\mathcal{A})_q$ for every $q \in Q$ and $(0, \ldots, 0) \in A_{q_f}$ for every $q_f \in F$. However, on this line, the while condition on line 5 is no longer valid. Hence, for every $q \xrightarrow{b} q'$ in Δ, we have $\lambda_b(A_{q'}) \subseteq A_q$. By induction, we obtain $\lambda_\sigma(A_{q'}) \subseteq A_q$ for every path $q \xrightarrow{\sigma} q'$. Moreover, as $(0, \ldots, 0) \in A_{q_f}$ for every $q_f \in F$, we have prove that $(A_q)_{q \in Q}$ is an affine covering. Therefore, for every $q \in Q$, we have $\text{cov}(\mathcal{A})_q \subseteq A_q$. We deduce $A_q = \text{cov}(\mathcal{A})_q$ for every $q \in Q$ on line 7.

Proposition 5.4 proves that on line 7, we have $\text{aff}(\bigcup_{q_0 \in Q_0} A_{q_0}) = \text{aff}(\rho(\mathcal{L}(\mathcal{A})))$. □

6 Application.

We study a natural extension of the Petri Nets, the effective counter systems, a class of systems that contains all the counter systems studied by Fast and Lash. For these systems, we naturally extend the definition of place invariants and prove that the computation of these invariants remains polynomial.

In the subsection 6.1, we prove that we can compute the affine hull of the reachability relation of a counter system in polynomial time in function of the affine hull of the reachability relation in one step. This polynomial time complexity is explained in the next subsection 6.2. In fact, we show the polynomial time link between the affine hull of the reachability relation of a counter system and a natural extension of place invariants. In the last subsection 6.3 we prove that the place invariants of an effective counter system can be computed in polynomial time.

6.1 The affine hull of the reachability relation of a counter system.

In this subsection, we prove that the affine hull of the reachability relation of a counter system can be computed in polynomial time in function of the affine hull of the reachability relation in one step.

A counter system is a general model for representing a system that uses m integer variables and has a finite set of actions Σ such that the new values $x' \in \mathbb{N}^m$ of the counters after executing an action $a \in \Sigma$ are related to the current value $x \in \mathbb{N}^m$ of the counters by a binary relation $x\mathcal{R}_a x'$ over $\mathbb{N}^m$.

Definition 6.1 A counter system S is a tuple $S = (\mathbb{N}^m, \Sigma, (\mathcal{R}_a)_{a\in\Sigma})$ where Σ is a finite set of actions and $(\mathcal{R}_a)_{a\in\Sigma}$ is a sequence of binary relations over $\mathbb{N}^m$.

The *one step reachability relation* associated to a counter system S is the binary relation written $\mathcal{R}_S$ and defined by $\mathcal{R}_S = \bigcup_{a\in\Sigma} \mathcal{R}_a$. The *reachability relation* associated to a counter system S is the binary relation $\mathcal{R}_S^*$ equal to reflexive and transitive closure of the one step reachability relation. Remark that $x\mathcal{R}_S^* x'$ iff there exists a finite sequence of $n \geq 0$ actions $(a_i)_{1\leq i\leq n}$ in Σ and a sequence of vectors $(x_i)_{0\leq i\leq n}$ in $\mathbb{N}^m$ such that $x = x_0$, $x_0\mathcal{R}_{a_1}x_1$, ..., $x_{n-1}\mathcal{R}_{a_n}x_n$ and $x_n = x'$.

The reachability problem consists in deciding for a counter system S and two vectors x and x' in $\mathbb{N}^m$ if we have $x\mathcal{R}_S^* x'$. Recall that this problem is undecidable for the class of Reset/Transfer Petri Nets [15], a subclass of counter systems. For this reason, we are interested in the computation of an over approximation of the reachability relation, easily computable.

The following proposition proves that the computation of the affine hull of $I \cup \mathcal{R}_S$ provides the affine hull of the reachability relation $\mathcal{R}_S^*$.

Proposition 6.2 *For any binary relation $\mathcal{R}$ over $\mathbb{Q}^m$, we have*

$$\mathrm{aff}(\mathcal{R}^*) = \mathrm{aff}(I \cup \mathcal{R})$$

Proof. From $I \cup \mathcal{R} \subseteq \mathcal{R}^*$, we deduce $\mathrm{aff}(I \cup \mathcal{R}) \subseteq \mathrm{aff}(\mathcal{R}^*)$. To prove the converse, we first show that $\mathrm{aff}(I\cup\mathcal{R})$ is a transitive relation. Let us consider (x, x') and (x', x'') in $\mathrm{aff}(I \cup \mathcal{R})$. Remark that if $x = x' = x''$ then $(x, x'') =$

$(x',x') \in I \subseteq \mathrm{aff}(I \cup \mathcal{R})$. Hence, we can assume that either $x \neq x'$ or $x' \neq x''$. The rank of the following linear system is then equal to 2 and hence admits at least one solution $(\alpha,\beta,\gamma) \in \mathbb{Q}^3$.

$$\begin{cases}\alpha + \beta.x + \gamma.x' = x \\ \alpha + \beta.x' + \gamma.x'' = x''\end{cases}$$

Let $\delta = 1 - (\alpha+\beta+\gamma)$. From $\alpha+\beta+\gamma+\delta = 1$, and as $(1,1)$, (x,x'), (x',x'') and $(0,0)$ are in the affine space $\mathrm{aff}(I \cup \mathcal{R})$, we deduce $(x,x'') = \alpha.(1,1) + \beta.(x,x') + \gamma.(x',x'') + \delta.(0,0) \in \mathrm{aff}(I \cup \mathcal{R})$. Therefore, we have proved that the relation $\mathrm{aff}(I \cup \mathcal{R})$ is transitive.

From $\mathcal{R} \subseteq \mathrm{aff}(I \cup \mathcal{R})$ and the transitivity of $\mathrm{aff}(I \cup \mathcal{R})$, we deduce $\mathcal{R}^* \subseteq \mathrm{aff}(I \cup \mathcal{R})$. Therefore $\mathrm{aff}(\mathcal{R}^*) \subseteq \mathrm{aff}(I \cup \mathcal{R})$. □

The previous proposition shows that the affine hull of $\mathcal{R}_S^*$ is equal to the affine hull of $I \cup \mathcal{R}_S$. From $\mathrm{aff}(I \cup \mathcal{R}_S) = \mathrm{aff}(I \bigcup_{a\in\Sigma} \mathrm{aff}(\mathcal{R}_a))$, we deduce the following corollary:

Corollary 6.3 *The affine hull of the reachability relation of a counter system S can be computed in polynomial time from the sequence of affine spaces $(\mathrm{aff}(\mathcal{R}_a))_{a\in\Sigma}$.*

6.2 The definition of place invariants.

In this section we show the link between $\mathrm{aff}(\mathcal{R}_S^*)$ and the notion of place invariant of a counter system.

We slightly extend the definition of place invariants given in [11] in a natural way.

Definition 6.4 A place invariant of a counter system S is a linear function $l : \mathbb{Q}^m \to \mathbb{Q}$ such that for every $a \in \Sigma$ and for every (x,x') satisfying $x\mathcal{R}_a x'$, we have $l(x) = l(x')$. The set of place invariants of a counter system S is written $\mathrm{inv}(S)$.

Remark 6.5 The set of place invariants of a counter system S is an affine space (and even a vector space).

The following proposition proves the link between $\mathrm{inv}(S)$ and $\mathrm{aff}(\mathcal{R}_S^*)$ by proving that the set of place invariants $\mathrm{inv}(S)$ is computable in polynomial time from $\mathrm{aff}(\mathcal{R}_S^*)$ and that $\mathrm{aff}(\mathcal{R}_S^*)$ is computable in polynomial time from $\mathrm{inv}(S)$.

Proposition 6.6 *For any counter system S, we have:*

$$\begin{cases}\mathrm{aff}(\mathcal{R}_S^*) = \{(x,x');\ l(x) = l(x')\ \forall l \in \mathrm{inv}(S)\} \\ \mathrm{inv}(S) = \{\textit{linear functions } l : \mathbb{Q}^m \to \mathbb{Q};\ l(x) = l(x')\ \forall (x,x') \in \mathrm{aff}(\mathcal{R}_S^*)\}\end{cases}$$

Proof. We first prove the equality $\mathrm{aff}(\mathcal{R}_S^*) = \{(x,x');\ l(x) = l(x')\ \forall l \in \mathrm{inv}(S)\}$.

Let $(x_0, x_0') \in \mathrm{aff}(\mathcal{R}_S^*)$ and let us prove that for any $l \in \mathrm{inv}(S)$, we have $l(x_0) = l(x_0')$. From the proposition 6.2, we have $(x_0, x_0') \in \mathrm{aff}(I \cup \mathcal{R}_S)$. There exist $t \in \mathbb{Q}$, a sequence $(t_a)_{a\in\Sigma}$ in $\mathbb{Q}$, a vector $(x,x) \in I$ and a sequence $((x_a, x_a'))_{a\in\Sigma}$ in $\mathcal{R}_a$ such that $t+\sum_{a\in\Sigma} t_a = 1$ and such that $(x_0, x_0') = t.(x,x)+\sum_{a\in\Sigma} t_a.(x_a, x_a')$. Let $l \in \mathrm{inv}(S)$. By definition of the place invariants, we have $l(x_a) = l(x_a')$ for every $a \in \Sigma$. Hence, we have $l(x_0) = l(t.x + \sum_{a\in\Sigma} t_a.x_a) = t.l(x) + \sum_{a\in\Sigma} t_a.l(x_a) = t.l(x) + \sum_{a\in\Sigma} t_a.l(x_a') = l(t.x + \sum_{a\in\Sigma} t_a.x_a') = l(x_0')$. Therefore, we have proved the inclusion $\mathrm{aff}(\mathcal{R}_S^*) \subseteq \{(x,x');\ l(x) = l(x')\ \forall l \in \mathrm{inv}(S)\}$. Let us prove the converse.

Let us consider $(x_0, x_0') \notin \mathrm{aff}(\mathcal{R}_S^*)$. In this case there exists an affine function $f : \mathbb{Q}^m \times \mathbb{Q}^m \to \mathbb{Q}$ such that $f(x,x') = 0$ for every $(x,x') \in \mathrm{aff}(\mathcal{R}_S^*)$ and such that $f(x_0, x_0') \neq 0$. As f is an affine function, there exist two linear functions l and l' and a rational $c \in \mathbb{Q}$ such that for every $(x,x') \in \mathbb{Q}^m \times \mathbb{Q}^m$, we have $f(x,x') = l(x) - l'(x') + c$. From $(x,x) \in I \subseteq \mathrm{aff}(\mathcal{R}^*)$ we deduce $0 = f(x,x) = l(x) - l'(x') + c$. Therefore $l = l'$ and $c = 0$. Moreover, for every $a \in \Sigma$ and for every $(x,x') \in \mathcal{R}_a$, we have $0 = f(x,x') = l(x) - l(x')$. Therefore $l \in \mathrm{inv}(S)$. From $f(x_0, x_0') \neq 0$, we deduce $l(x_0) \neq l(x_0')$. We have proved the inclusion $\{(x,x');\ l(x) = l(x')\ \forall l \in \mathrm{inv}(S)\} \subseteq \mathrm{aff}(\mathcal{R}_S^*)$.

We deduce the first equality $\mathrm{aff}(\mathcal{R}_S^*) = \{(x,x');\ l(x) = l(x')\ \forall l \in \mathrm{inv}(S)\}$. Now, let us prove the second equality. Let $l \in \mathrm{inv}(S)$. From the previous equality, we have $l(x) = l(x')$ for every $(x,x') \in \mathrm{aff}(\mathcal{R}_S^*)$. Therefore $\mathrm{inv}(S) \subseteq \{\text{linear functions } l : \mathbb{Q}^m \to \mathbb{Q};\ l(x) = l(x')\ \forall (x,x') \in \mathrm{aff}(\mathcal{R}_S^*)\}$. For the converse inclusion, let us consider a linear function l such that $l(x) = l(x')$ for every $(x,x') \in \mathrm{aff}(\mathcal{R}_S^*)$. From $\mathcal{R}_S \subseteq \mathrm{aff}(\mathcal{R}_S^*)$, we deduce that for every $a \in \Sigma$ and for every $(x,x') \in \mathcal{R}_a$, we have $l(x) = l(x')$. Therefore $l \in \mathrm{inv}(S)$. We have proved the second equality. □

6.3 Effective counter systems.

In this subsection, we define the class of effective counter systems and we prove that the affine hull of the reachability relation is computable in polynomial time. As a corollary, we prove that the set of place invariants of an effective counter systems are computable in polynomial time. Finally, we present results obtained by implementing these results in our tool Fast.

Tools like Fast or Lash takes as input a class of counter systems such that each binary relation $\mathcal{R}_a$ is represented by a BA-affine function f_a.

Definition 6.7 A BA-affine function f is a tuple $(\mathcal{A}, M, v)$ such that $\mathcal{A}$ is a binary automaton, $M \in \mathcal{M}_m(\mathbb{Q})$ is a square matrix and $v \in \mathbb{Q}^m$ is a vector. The relation $\mathcal{R}_f$ associated to a BA-affine relation f is defined by:

$$x \mathcal{R}_f x' \iff x' = M.x + v \text{ and } x \in \rho(\mathcal{L}(\mathcal{A}))$$

Definition 6.8 An effective counter system S is a counter system $S = (\mathbb{N}^m, \Sigma, \mathcal{R}_\Sigma)$

such that every binary relation $\mathcal{R}_a$ is either represented by a binary automaton $\mathcal{A}_a$ such that $\mathcal{R}_a = \rho(\mathcal{L}(\mathcal{A}_a)) \subseteq \mathbb{N}^{2.m}$ or represented by a BA-affine function f_a such that $\mathcal{R}_a = \mathcal{R}_{f_a}$.

For effective counter systems, the complexity for the computation of place invariants, known to be polynomial time for Petri Nets remains polynomial time.

Theorem 6.9 *The affine hull of the reachability relation of an effective counter system is computable in polynomial time.*

Proof. From the theorem 6.2, we prove that $\mathrm{aff}(\mathcal{R}_S^*)$ is equal to $\mathrm{aff}(I \cup \mathcal{R}_S)$. Remark that $\mathrm{aff}(I \cup \mathcal{R}_S) = \mathrm{aff}(I \bigcup_{a\in\Sigma} \mathrm{aff}(\mathcal{R}_a))$. Let $a \in \Sigma$; the binary relation $\mathcal{R}_a$ is either represented by a binary automaton $\mathcal{A}_a$ or by a BA-affine function f_a. If $\mathcal{R}_a$ is represented by a binary automaton $\mathcal{A}_a$, theorem 5.5 proves that the affine space $\mathrm{aff}(\mathcal{R}_a)$ is computable in polynomial time. Assume that $\mathcal{R}_a$ is represented by a BA-affine function $f_a = (\mathcal{A}_a, M_a, v_a)$. Theorem 5.5 proves that $\mathrm{aff}(\rho(\mathcal{L}(\mathcal{A}_a)))$ is computable in polynomial time. Therefore, $\mathrm{aff}(\mathcal{R}_{f_a}) = (\mathrm{aff}(\rho(\mathcal{L}(\mathcal{A}_a))) \times \mathbb{Q}^m) \cap \{(x, x') \in \mathbb{Q}^m \times \mathbb{Q}^m;\ x' = M_a.x + v_a\}$ is computable in polynomial time. Therefore, we can compute $\mathrm{aff}(\mathcal{R}_a)$ in polynomial time for each action $a \in \Sigma$. We have proved that we can compute $\mathrm{aff}(\mathcal{R}_S^*)$ in polynomial time. □

From the proposition 6.6 and the previous theorem, we deduce that the place invariants of an effective counter systems are computable in polynomial time.

Corollary 6.10 *The set of place invariants of an effective counter system is computable in polynomial time.*

We have implemented the computation of the affine covering of a binary automaton as a plug-in for our symbolic model checker Fast. The computation of the place invariants of the 40 examples of counter systems provided with Fast takes between 1 minute to 1 hour to be computed. However, this time complexity do not correponds to the time complexity of an optimized algorithm. In fact, we are quite sure that the required time for computing these place invariants can be reduced to less than one second. For this reason, no benchmark are provided. Fast with invariants will be available as soon as this algorithm is optimized on the fast-webpage `www.lsv.ens-cachan.fr/fast/`

Conclusion.

We have presented the class of binary automaton, a new representation of subsets of $\mathbb{N}^m$ that naturally extends the NDD. We have proved that the affine hull of the set of vectors represented by a binary automaton is computable in polynomial time. We have introduced the model of effective counter systems. We have proved that we can extend the definition of place invariants to this new class. The computation of the place invariants remains polynomial time for the class of effective counter systems. All the algorithms have been implemented in C++ as a plug-in for Fast and they should be soon available.

References

[1] Abdulla, P. A., A. Bouajjani and B. Jonsson, *On-the-fly analysis of systems with unbounded, lossy FIFO channels*, in: *Proc. 10th Int. Conf. Computer Aided Verification (CAV'98), Vancouver, BC, Canada, June-July 1998*, Lecture Notes in Computer Science **1427** (1998), pp. 305–318.

[2] Abdulla, P. A. and B. Jonsson, *Verifying programs with unreliable channels*, in: *Proc. 8th IEEE Symp. Logic in Computer Science (LICS'93), Montreal, Canada, June 1993* (1993), pp. 160–170.

[3] Ammirati, P., G. Delzanno, P. Ganty, G. Geeraerts, J.-F. Raskin and L. V. Begin, *Babylon: An integrated toolkit for the specification and verification of parameterized systems*, in: G. Delzanno, S. Etalle and M. Gabbrielli, editors, *Specification, Analysis and Validation for Emerging Technologies in Computational Logic*, Datalogiske Skrifter **94**, 2002, p. (unpaginated).

[4] Annichini, A., A. Bouajjani and M. Sighireanu, *TReX: A tool for reachability analysis of complex systems*, in: *Proc. 13th Int. Conf. Computer Aided Verification (CAV'2001), Paris, France, July 2001*, Lecture Notes in Computer Science **2102** (2001), pp. 368–372.

[5] *BABYLON*, http://www.ulb.ac.be/di/ssd/lvbegin/CST/index.html.

[6] Bardin, S., A. Finkel, J. Leroux and L. Petrucci, *FAST: Fast Acceleration of Symbolic Transition systems*, in: *Proc. 15th Int. Conf. Computer Aided Verification (CAV'2003), Boulder, CO, USA, July 2003*, Lecture Notes in Computer Science **2725** (2003), pp. 118–121.

[7] Berthelot, G., "Transformations et Analyse de Reseaux de Petri. Applications aux Protocoles." These d'Etat, Univ. Paris VI, 1983, newsletterInfo: 16.

[8] Boigelot, B., *On iterating linear transformations over recognizable sets of integers*, Theoretical Computer Science To appear.

[9] Boigelot, B., "Symbolic Methods for Exploring Infinite State Spaces," Ph.D. thesis, Université de Liège (1998).

[10] Boudet, A. and H. Comon, *Diophantine equations, Presburger arithmetic and finite automata*, in: *Proc. 21st Int. Coll. on Trees in Algebra and Programming (CAAP'96), Linköping, Sweden, Apr. 1996*, Lecture Notes in Computer Science **1059** (1996), pp. 30–43.

[11] Ciardo, G., *Petri nets with marking-dependent arc cardinality: Properties and analysis*, in: *Proc. 15th Int. Conf. Application and Theory of Petri Nets (ICATPN'94), Zaragoza, Spain, June 1994*, Lecture Notes in Computer Science **815** (1994), pp. 179–198.

[12] Delzanno, G., *Constraint-based verification of parameterized cache coherence protocols*, Formal Methods in System Design To appear.

[13] Delzanno, G., *Verification of consistency protocols via infinite-state symbolic model checking: A case study*, in: *Proc. IFIP Joint Int. Conf. Formal Description Techniques & Protocol Specification, Testing, and Verification (FORTE-PSTV'00), Pisa, Italy, Oct. 2000*, IFIP Conference Proceedings **183** (2000), pp. 171–186.

[14] Delzanno, G., J.-F. Raskin and L. Van Begin, *Attacking symbolic state explosion*, in: *Proc. 13th Int. Conf. Computer Aided Verification (CAV'2001), Paris, France, July 2001*, Lecture Notes in Computer Science **2102** (2001), pp. 298–310.

[15] Dufourd, C., A. Finkel and P. Schnoebelen, *Reset nets between decidability and undecidability*, in: *Proc. 25th Int. Coll. Automata, Languages, and Programming (ICALP'98), Aalborg, Denmark, July 1998*, Lecture Notes in Computer Science **1443** (1998), pp. 103–115.

[16] Emerson, E. A. and K. S. Namjoshi, *On model checking for non-deterministic infinite-state systems*, in: *Proc. 13th IEEE Symp. Logic in Computer Science (LICS'98), Indianapolis, IN, USA, June 1998* (1998), pp. 70–80.

[17] *FAST*, http://www.lsv.ens-cachan.fr/∼leroux/fast/.

[18] Finkel, A. and J. Leroux, *How to compose Presburger-accelerations: Applications to broadcast protocols*, in: *Proc. 22nd Conf. Found. of Software Technology and Theor. Comp. Sci. (FST&TCS'2002), Kanpur, India, Dec. 2002*, Lecture Notes in Computer Science **2556** (2002), pp. 145–156.

[19] Finkel, A. and P. Schnoebelen, *Well structured transition systems everywhere!*, Theoretical Computer Science **256** (2001), pp. 63–92.

[20] Klarlund, N., *Mona and fido: The logic-automaton connection in practice* (1997).

[21] Klarlund, N., A. Møller and M. I. Schwartzbach, *MONA implementation secrets*, Int. J. of Foundations Computer Science **13** (2002), pp. 571–586.

[22] *Lash*, http://www.montefiore.ulg.ac.be/∼boigelot/research/lash/.

[23] *MONA*, http://www.brics.dk/mona/index.html.

[24] *TREX*, http://www.liafa.jussieu.fr/ sighirea/trex/.

[25] Wolper, P. and B. Boigelot, *An automata-theoretic approach to Presburger arithmetic constraints*, in: *Proc. 2nd Int. Symp. Static Analysis (SAS'95), Glasgow, UK, Sep. 1995*, Lecture Notes in Computer Science **983** (1995), pp. 21–32.

[26] Wolper, P. and B. Boigelot, *On the construction of automata from linear arithmetic constraints*, in: *Proc. 6th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2000), Berlin, Germany, Mar.-Apr. 2000*, Lecture Notes in Computer Science **1785** (2000), pp. 1–19.