Transitive Closures of Ultimately Periodic Relations *

published as VERIMAG Technical Report TR-2011-14

Marius Bozga[1], Radu Iosif[1], and Filip Konečný[1,2]

[1] VERIMAG, CNRS, 2 av. de Vignate, 38610 Gières, France, {bozga,iosif}@imag.fr
[2] FIT BUT, Božetěchova 2, 61266, Brno, Czech Republic, {ikonecny}@fit.vutbr.cz

Abstract. Computing transitive closures of integer relations is the key to finding precise invariants of integer programs. In this paper, we describe an efficient algorithm for computing the transitive closures of difference bounds, octagonal and finite monoid affine relations. On the theoretical side, this framework provides a common solution to the acceleration problem, for all these three classes of relations. In practice, according to our experiments, the new method performs up to four orders of magnitude better than the previous ones, making it a promising approach for the verification of integer programs.

1 Introduction

Counter automata (counter systems) are important models of computation, that can naturally encode many classes of systems with unbounded (or very large) data domains, such as hardware circuits with cache memory, programs with integer variables, integer arrays and recursive data structures. From a purely theoretical point of view, any Turing complete class of systems can be simulated by counter automata, as shown by Minsky in [30]. Unfortunately, the general reduction scheme is rather complex, and is mostly used to prove undecidability of various problems, such as e.g., reachability, emptiness, boundedness, disjointness, containment, equivalence.

On the other hand, recent research has revealed several methods reducing verification problems of several classes of systems to decision problems on counter automata in a more practical way, hence the growing interest for analysis tools working on counter automata. Examples of such systems that can be effectively verified by means of counter automata include: specifications of hardware components [35], programs with singly-linked lists [6, 21, 12], trees [24], and integer arrays [9].

Since the result of Minsky [30] proving Turing-completeness of 2-counter machines with increment, decrement and zero test, research on counter automata has been pursued in two orthogonal directions. The first one is defining subclasses of counter automata for which various decision problems are found to be decidable. Examples include Reversal-bounded Counter Machines, [26], Petri Nets and Vector Addition Systems, [32], or Flat Counter Automata, [28]. Often, decidability of various problems is achieved by

* This work was supported by the French national project ANR-09-SEGI-016 VERIDYC, by the Barrande programme (MEB021023), the Czech Science Foundation (projects P103/10/0306 and 102/09/H042), the Czech Ministry of Education (projects COST OC10009 and MSM 0021630528), and the internal FIT BUT grant FIT-S-10-1.

defining the set of reachable configurations in a decidable logic, such as Presburger arithmetic, [31]. Such definitions are precise, i.e. no information is lost by the use of over-approximation.

Another, orthogonal, direction of work is concerned with finding sound (but not necessarily complete) answers to the decision problems mentioned above, in a cost-effective way. Such approaches use abstract domains (Polyhedra [16], Octagons [29], etc.) and compute fixed points that are overapproximations of the set of reachable configurations. When the abstraction is too coarse to yield meaningful results, refinement techniques can be applied to rule out several spurious counterexamples. This technique is known as Counter Example-based Abstraction Refinement (CEGAR) [13].

The work reported in this paper lies within the lines of the first direction mentioned above. Given a counter automaton, we aim at computing the relation between the input and the output counter valuations, whenever this relation can be expressed in Presburger arithmetic. The main application of deriving the input-output relation of a counter automaton in a decidable logic is an algorithm for verifying safety properties of systems, which can be encoded as reachability conditions.

At the heart of our method lies a technique for computing transitive closures of loops labeled by conjunctive transition relations. In general, the formula defining the transitive closure of a linear relation falls outside of known decidable fragments of arithmetic. To this end, it is important to know for which classes of transition relations it is possible to compute the transitive closure precisely and fast. To the best of our knowledge, the three main classes of integer relations for which transitive closures can be computed effectively and precisely are: (1) difference bounds constraints [14, 8], (2) octagons [29, 7], and (3) finite monoid affine transformations [5, 20]. For these three classes, the transitive closures can be moreover defined in Presburger arithmetic.

The contributions of this paper are three-fold. First, we study the three non-trivial classes of relations mentioned in the previous and show that they are ultimately periodic, i.e. that each relation R in these classes can be mapped into an integer matrix M_R such that the sequence $\{M_{R^k}\}_{k=0}^{\infty}$ is periodic. The proof of the fact that a sequence of matrices is ultimately periodic relies on a result from tropical semiring theory [34]. This provides shorter proofs to the fact that the transitive closures for these classes can be effectively computed, and that they are Presburger definable, as initially proved in [14, 7, 5].

Second, the algorithm introduced in this paper computes the transitive closure of difference bounds and octagonal relations up to four orders of magnitude faster than our original methods from [8, 7], and also scales much better in the number of variables. The experimental comparison with the FAST tool [4] for difference bounds relations shows that large relations (up to hundred variables), causing FAST to run out of memory, can now be handled by our implementation in a matter of few seconds, on average.

Third and last, we give a semi-algorithmic method for computing the input-output relation of a counter automaton. The algorithm builds the relation incrementally, by eliminating control states and composing incoming with outgoing relations. The main difficulty here is the elimination of states with several self-loops. We tackle this issue by first computing the transitive closures of the self-loops individually, and then exploring all interleavings between them, until no new relations are discovered. Obviously, this exploration might not end, in which case we stop it at a certain depth and give up.

The techniques reported here are implemented in the FLATA toolset [22] for the analysis of counter automata models. So far, this toolset has been successfully applied on models that where automatically generated from VHDL specifications of hardware, programs with lists, and verification conditions of programs with integer arrays.

The material contained in Sections 2, 3 and 4 presents an extended version of the work reported in [10]. Section 5 contains novel, unpublished material, concerning the application of the transitive closure computation techniques to the analysis of counter automata, and presents an experimental comparison with existing tools, based on different techniques (e.g. predicate abstraction, widening, etc.).

1.1 Related Work

Early attempts to apply Model Checking techniques to the verification of infinite-state systems consider the problem of accelerating transition relations by successive under-approximations, without any guarantee of termination. For systems with integer variables, the acceleration of affine relations has been considered primarily in the works of Annichini et. al [1], Boigelot [5], and Finkel and Leroux [20]. Finite monoid affine relations have been first studied by Boigelot [5], who shows that the finite monoid property is decidable, and that the transitive closure is Presburger definable in this case. On what concerns non-deterministic transition relations, difference bounds constraints appear in the context of timed automata verification. The transitive closure of a difference bounds constraint is shown to be Presburger definable first by Comon and Jurski [14]. Their proof was subsequently simplified and extended to parametric difference bounds constraints in [8]. We also showed that octagonal relations can be accelerated precisely, and that the transitive closure is also Presburger definable [7]. The proofs of ultimate periodicity from this paper are based on some of our previous results [8, 7]. For difference bounds constraints, the proof from [8] was simplified using a result from tropical semiring theory [34].

Roadmap The paper is organized as follows. Section 2 introduces defines the working concepts and Section 3 describes the generic transitive closure algorithm for ultimately periodic relations. Most of the technical detail is found in Section 4, which studies three classes of ultimately periodic relations. Finally, Section 5 shows how the transitive closure of ultimately periodic relations helps answering reachability and emptiness queries on counter automata, and Section 6 provides experimental evidence of the usefulness of the techniques described in the paper.

2 Preliminaries

We denote by $\mathbb{Z}$, $\mathbb{N}$ and $\mathbb{N}_+$ the sets of integers, positive (including zero) and strictly positive integers, respectively. The first order additive theory of integers is known as Presburger Arithmetic. For a first-order logical formula φ, let $FV(\varphi)$ denote the set of its free variables. By writing $\varphi(x_1, \ldots, x_n)$ we imply that $FV(\varphi) \subseteq \{x_1, \ldots, x_n\}$.

For a formula $\varphi(x_1, \ldots, x_n)$, we denote by $\varphi[t_1/x_1, \ldots, t_n/x_n]$ the formula obtained from φ by syntactically replacing each free occurrence of $x_1, \ldots, x_n$ by the terms $t_1, \ldots, t_n$, respectively. An integer relation is represented by an arithmetic formula $R(x_1, \ldots, x'_1, \ldots, x'_n)$, where $x_1, \ldots, x_n$ denote input variables, and $x'_1, \ldots, x'_n$ denote output variables. By $\models \varphi$ we denote the fact that φ is valid. We use the symbols $\Rightarrow, \Leftrightarrow$ to denote logical implication and equivalence, respectively.

2.1 Ultimately Periodic Sequences

The tropical semiring is defined as $\mathbb{T} = (\mathbb{Z}_\infty, \min, +, \infty, 0)$ [34], where $\mathbb{Z}_\infty = \mathbb{Z} \cup \{\infty\}$, with the extended arithmetic operations $x + \infty = \infty$, $\min(x, \infty) = x$, for all $x \in \mathbb{Z}$, where $\min(x, y)$ denotes the minimum between the values x and y. For two square matrices $A, B \in S^{m \times m}$, we define $(A + B)_{ij} = A_{ij} + B_{ij}$ and $(A \times B)_{ij} = \min_{k=1}^{m}(a_{ik} + b_{kj})$, for all $1 \leq i, j \leq m$. Let $\mathbf{I} \in \mathbb{T}^{m \times m}$ be the identity matrix, i.e. $\mathbf{I}_{ii} = 0$ and $\mathbf{I}_{ij} = \infty$, for all $1 \leq i, j \leq m$, $i \neq j$.

Definition 1. *[34] An infinite sequence $\{s_k\}_{k=0}^{\infty} \in \mathbb{T}$ is called ultimately periodic if:*

$$\exists K \; \exists c > 0 \; \exists \lambda_0, \lambda_1, \ldots, \lambda_{c-1} \in \mathbb{T} \; . \; s_{(k+1)c+i} = \lambda_i + s_{kc+i}$$

for all $k \geq K$ and $i = 0, 1, \ldots, c-1$. The smallest c and $\lambda_0, \lambda_1, \ldots, \lambda_{c-1}$ for which the above holds are called the period and rates of $\{s_k\}_{k=0}^{\infty}$, respectively.

Example 1. The sequence $\sigma_k = \{3k+1 \mid k = 2l,\ l \geq 2\} \cup \{5k+3 \mid k = 2l+1,\ l \geq 2\}$ is ultimately periodic, with $K = 4$, period $c = 2$ and rates $\lambda_0 = 3$, $\lambda_1 = 5$.

A sequence of matrices $\{A_k\}_{k=0}^{\infty} \in \mathbb{T}^{m \times m}$ is said to be ultimately periodic if, for all $1 \leq i, j \leq m$, the sequence $\{(A_k)_{ij}\}_{k=0}^{\infty}$ is ultimately periodic. A matrix $A \in \mathbb{T}^{m \times m}$ is called ultimately periodic if the sequence $\{A^k\}_{k=1}^{\infty}$ is ultimately periodic, where $A^0 = \mathbf{I}$ and $A^k = A \times A^{k-1}$, for any $k > 0$. It is known that, every matrix is ultimately periodic in the tropical semiring [34].

If $A \in \mathbb{T}^{m \times m}$ is a square matrix and $n \in \mathbb{T}$, we define the matrix $(n \cdot A)_{ij} = n \cdot A_{ij}$, for all $1 \leq i, j \leq m$. If k is a parameter (typically interpreted over $\mathbb{T}$), then $\mathbb{T}[k]$ denotes the set of all terms where k may occur, built from the constants and operators of $\mathbb{T}$. For instance, if $A, B \in \mathbb{T}^{m \times m}$, then $k \cdot A + B \in \mathbb{T}[k]^{m \times m}$ denotes the matrix of terms $(k \cdot A + B)_{ij} = k \cdot A_{ij} + B_{ij}$, for all $1 \leq i, j \leq m$.

We have the following characterization of ultimately periodic sequences of matrices:

Lemma 1. *A sequence of matrices $\{A_k\}_{k=1}^{\infty} \in \mathbb{T}^{m \times m}$ is ultimately periodic if and only if:*

$$\exists K \; \exists c > 0 \; \exists \Lambda_0, \Lambda_1, \ldots, \Lambda_{c-1} \in \mathbb{T}^{m \times m} \; . \; A_{(k+1)c+i} = \Lambda_i + A_{kc+i}$$

for all $k \geq K$ and $i = 0, 1, \ldots, c-1$.

Proof. According to the definition, $\{A_k\}_{k=1}^{\infty}$ is ultimately periodic if and only if, for each $1 \leq i,j \leq m$ there exist K_{ij}, $c_{ij} > 0$ and $\lambda_l^{ij} \in \mathbb{T}$ such that $\left(A_{(k+1)c_{ij}+l}\right)_{ij} = \lambda_l^{ij} + \left(A_{kc_{ij}+l}\right)_{ij}$ for all $l = 0,1,\ldots,c_{ij}-1$. Let c be the least common multiple of all c_{ij}, further let $b_{ij} = c_{ij} \cdot K_{ij}$ for each $1 \leq i,j \leq n$, $\overline{b}$ be the maximum of all b_{ij}, $b = c \cdot \left\lceil \frac{\overline{b}}{c} \right\rceil$ and let Λ_t, $t = 0,1,\ldots,c-1$ be the matrix defined as:

$$(\Lambda_t)_{ij} = \left(\lambda^{ij}_{(b-b_{ij}+t) \mod c_{ij}}\right)^{\frac{c}{c_{ij}}}$$

The condition $A_{(k+1)c+i} = \Lambda_i + A_{kc+i}$ is verified for all $k \geq \lceil \frac{b}{c} \rceil$ and $i = 0,1,\ldots,c-1$, with the above definitions.

□

2.2 Ultimately Periodic Relations

Let $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$ be a set of variables, $N > 0$, and let $\mathbf{x}' = \{x'_1, x'_2, \ldots, x'_N\}$. A relation is an arithmetic formula $R(\mathbf{x}, \mathbf{x}')$ with free variables $\mathbf{x} \cup \mathbf{x}'$. We say that two relations R and R' are equivalent, denoted $R \Leftrightarrow R'$ if under all valuations of $\mathbf{x}$ and $\mathbf{x}'$, R is true if and only if R' is true. A relation is called consistent if and only if there exist valuations of $\mathbf{x}$ and $\mathbf{x}'$ under which it holds. We denote a consistent relation R by writing $R \not\Leftrightarrow$ false, and an inconsistent relation by writing $R \Leftrightarrow$ false.

The composition of two relations is defined as $R \circ R' \equiv \exists \mathbf{y} \,.\, R(\mathbf{x}, \mathbf{y}) \wedge R'(\mathbf{y}, \mathbf{x}')$. Let $\mathcal{I}$ be the identity relation $\bigwedge_{x \in \mathbf{x}} x' = x$. We define $R^0 \equiv \mathcal{I}$ and $R^n \equiv R^{n-1} \circ R$, for any $n > 0$. With these notations, $R^* \equiv \bigvee_{i=0}^{\infty} R^i$ denotes the transitive closure of R. A relation R is called ω-consistent if R^n is consistent for all $n > 0$. For the rest of this section, let $\mathcal{R}$ be a class of relations[3].

Definition 2. *A relation $R(\mathbf{x}, \mathbf{x}') \in \mathcal{R}$ is called ultimately periodic if and only if either:*

1. *there exists $i_0 \geq 0$ such that R^{i_0} is inconsistent, or*
2. *for all $i \geq 0$, R^i is consistent, and there exists two functions:*
   - *$\sigma : \mathcal{R} \to \mathbb{T}_{\perp}^{m \times m}$ mapping each consistent relation in $\mathcal{R}$ into a $m \times m$ matrix of $\mathbb{T}$, for some $m > 0$, and each inconsistent relation into $\perp$.*
   - *$\rho : \mathbb{T}^{m \times m} \to \mathcal{R}$ mapping each $m \times m$ matrix of $\mathbb{T}$ into a relation in $\mathcal{R}$, such that $\rho(\sigma(R)) \Leftrightarrow R$, for each consistent relation $R \in \mathcal{R}$*

   such that the infinite sequence of matrices $\{\sigma(R^i)\}_{i=0}^{\infty} \in \mathbb{T}^{m \times m}$ is ultimately periodic.

Notice that the first condition of the definition implies that $\sigma(R^i) = \perp$, for all $i \geq i_0$. If each relation $R \in \mathcal{R}$ is ultimately periodic, then $\mathcal{R}$ is called ultimately periodic as well. The following lemma gives an alternative characterization of ω-consistent ultimately periodic relations.

Lemma 2. *An ω-consistent relation R is ultimately periodic if and only if there exist $K \geq 0$, $b \geq 0$, $c > 0$ and $\Lambda_0, \Lambda_1, \ldots, \Lambda_{c-1} \in \mathbb{T}^{m \times m}$ such that the following hold:*

[3] A class of relations is usually defined by syntactic conditions.

1. $\sigma(R^{(k+1)c+i}) = \Lambda_i + \sigma(R^{kc+i})$*, for all* $k \geq K$.
2. $R^{kc+b+i} \Leftrightarrow \rho(k \cdot \Lambda_i + \sigma(R^{b+i}))$*, for all* $k \geq 0$.

for all $i = 0, 1, \ldots, c-1$*, where* σ *and* ρ *are the functions from Def. 2.*

Proof. By Lemma 1, if R is ω-consistent, then it is ultimately periodic if and only if

$$\exists K \; \exists c > 0 \; \exists \Lambda_0, \Lambda_1, \ldots, \Lambda_{c-1} \in \mathbb{T}^{N \times N} \; . \; \sigma(R^{(k+1)c+i}) = \Lambda_i + \sigma(R^{kc+i})$$

for all $k \geq K$ and $i = 0, 1, \ldots, c-1$. By induction on $k \geq K$, one shows first that

$$R^{kc+i} \Leftrightarrow \rho({\Lambda_i}^{k-K} + \sigma(R^{Kc+i})), \; \forall k \geq K$$

Let $b = Kc$. By replacing $k - K$ with k, we obtain

$$R^{kc+b+i} \Leftrightarrow \rho({\Lambda_i}^{k} + \sigma(R^{b+i})), \; \forall k \geq 0$$

□

For practical reasons related to the representation of R^*, we are interested in finding the symbolic expression R^k, where k is a parameter (because $R^* \equiv \exists k \, . \, R^k$). Notice that the second point of lemma 2 can be used to compute the expression R^k symbolically (as a formula over $\mathbf{x}$, $\mathbf{x}'$ and k), assuming that we are given a function, call it $\pi : \mathbb{T}[k]^{m \times m} \to \mathcal{R}(k)$, where $\mathcal{R}(k)$ is the class of all parametric relations over $\mathbf{x}, \mathbf{x}'$ and k. Intuitively, π is the parametric counterpart of the ρ function from Def. 2, mapping a matrix of terms over k into a parametric relation $R(\mathbf{x}, \mathbf{x}', k)$. Concrete definitions of π will be given in Section 4.

3 Computing Transitive Closures of Ultimately Periodic Relations

In this section we give a generic algorithm that computes the transitive closure of a given ultimately periodic relation. The algorithm needs to be instantiated for a specific class $\mathcal{R}$ of ultimately periodic relations by providing the mappings σ, ρ (Def. 2) and π (the parametric counterpart of ρ) as discussed in the previous. Next, in Section 4, we show how this algorithm can be used for accelerating three classes of relations: difference bounds, octagons, and finite monoid affine transformations.

Fig. 1 shows the generic framework for computing transitive closures. The input to the algorithm is a relation R, and the mappings $\sigma : \mathcal{R} \to \mathbb{T}^{m \times m}$, $\rho : \mathbb{T}^{m \times m} \to \mathcal{R}$, and $\pi : \mathbb{T}[k]^{m \times m} \to \mathcal{R}(k)$. The algorithm is guaranteed to terminate if R is ultimately periodic, as it will be explained in the following.

The main idea of the algorithm is to discover the prefix b and period c of the sequence $\{\sigma(R^i)\}_{i=0}^{\infty}$ – cf. the second point of lemma 2. If R is ultimately periodic, such values are guaranteed to exist. The dove-tailing enumeration on lines 1 and 2 is guaranteed to yield the smallest values (b, c) for which the sequence is shown to be periodic.

Second, the algorithm attempts to compute the first rate of the sequence (line 6), by comparing the matrices $\sigma(R^b)$, $\sigma(R^{c+b})$ and $\sigma(R^{2c+b})$. If the difference Λ between $\sigma(R^{c+b})$ and $\sigma(R^b)$ equals the difference between $\sigma(R^{2c+b})$ and $\sigma(R^{c+b})$, then Λ is a valid candidate for the first rate of the progression (see lemma 2). Notice that the consistency check on line 4 is needed to ensure that we apply σ to consistent relations –

otherwise, the relation is not ω-consistent, and the algorithm returns directly the transitive closure, i.e. the finite disjunction $\bigvee_{i=0}^{kc+b-1} R^i,\ 0 \le k \le 2$ (line 4).

Once a candidate Λ for the initial rate was found, the test $\mathcal{Q}_1$ on line 7 is used to check that R is ultimately periodic and ω-consistent. Notice that the characterization of ultimately periodic relations from lemma 2 cannot be applied here, since R^k is not known in general, for arbitrary $k \ge 0$. The condition used here is local, i.e. it needs only the relation R^b, for a typically small constant $b \ge 0$. The next lemma establishes the correctness of the criterion used by $\mathcal{Q}_1$:

Lemma 3. *An ω-consistent relation R is ultimately periodic if and only if*

$$\exists b\, \exists c > 0\, \exists \Lambda_0, \Lambda_1, \ldots, \Lambda_{c-1} \in \mathbb{T}^{m\times m} \;.\; \rho(k{\cdot}\Lambda_i{+}\sigma(R^{b+i})){\circ}R^c \Leftrightarrow \rho((k{+}1){\cdot}\Lambda_i{+}\sigma(R^{b+i}))$$

for all $k \ge 0$ and $i = 0, 1, \ldots, c-1$, where σ and ρ are the functions from Def. 2. Moreover, $\Lambda_0, \Lambda_1, \ldots, \Lambda_{c-1}$ satisfy the equivalences of Lemma 2.

Proof. "$\Rightarrow$" If R is ω-consistent and ultimately periodic, by Lemma 2, there exist $b \ge 0$, $c > 0$ and $\Lambda_0, \Lambda_1, \ldots, \Lambda_{c-1} \in \mathbb{T}^{m\times m}$ such that

$$R^{kc+b+i} \Leftrightarrow \rho({\Lambda_i}^k + \sigma(R^{b+i}))$$

for all $k \ge 0$ and $i = 0, 1, \ldots, c-1$. We have:

$$\begin{array}{ccc} R^{(k+1)c+b+i} & \Leftrightarrow & R^{kc+b+i} \circ R^c \\ \rho({\Lambda_i}^{k+1} + \sigma(R^{b+i})) & \Leftrightarrow & \rho({\Lambda_i}^k + \sigma(R^{b+i})) \circ R^c \end{array}$$

"$\Leftarrow$" We prove the equivalent condition of Lemma 2 by induction on $k \ge 0$. The base case $k = 0$ is immediate. The induction step is as follows:

$$\begin{array}{rl} R^{(k+1)c+b+i} \Leftrightarrow & R^{kc+b+i} \circ R^c \\ \Leftrightarrow & \rho({\Lambda_i}^k + \sigma(R^{b+i})) \circ R^c \text{ , by the induction hypothesis} \\ \Leftrightarrow & \rho({\Lambda_i}^{k+1} + \sigma(R^{b+i})) \end{array}$$

□

The universal query $\mathcal{Q}_1$ on line 7 is in general handled by procedures that are specific to the class of relations $\mathcal{R}$ we work with. Notice furthermore that $\mathcal{Q}_1$ can be handled symbolically by checking the validity of the first order formula: $\forall k \;.\; \pi(k \cdot \Lambda + \sigma(R^b)) \circ R^c \Leftrightarrow \pi((k+1) \cdot \Lambda + \sigma(R^b))$, where π is the parametric counterpart of ρ. Next, in Section 4, we detail two ways in which this test can be performed efficiently (for difference bounds and octagonal relations), without resorting to external proof engines, such as SMT or Presburger solvers.

If the universal query on line 7 holds, the rate Λ can be used now to express the transitive closure (line 8) as a finite disjunction over the prefix ($\bigvee_{i=0}^{b-1} R^i$) followed by a formula defining an arbitrary number of iterations ($\exists k \;.\; \bigvee_{i=0}^{c-1} \pi(k \cdot \Lambda + \sigma(R^b)) \circ R^i$). Note that the formula on line 8 defines indeed the transitive closure of R, as a consequence of lemma 2. Moreover, this is a formula of Presburger arithmetic, provided that the classes of relations $\mathcal{R}$ and $\mathcal{R}(k)$ are Presburger definable.

1. foreach $b := 0, 1, 2, \dots$ do
2. foreach $c := 0, 1, \dots, b$ do
3. foreach $k := 0, 1, 2$ do
4. if $R^{kc+b} \Leftrightarrow \text{false}$ then return $R^* \equiv \bigvee_{i=0}^{kc+b-1} R^i$
5. endfor
6. if exists $\Lambda \in \mathbb{T}^{m \times m} : \sigma(R^{c+b}) = \Lambda + \sigma(R^b)$ and $\sigma(R^{2c+b}) = \Lambda + \sigma(R^{c+b})$ then
7. if forall $k \geq 0 : \rho(k \cdot \Lambda + \sigma(R^b)) \circ R^c \Leftrightarrow \rho((k+1) \cdot \Lambda + \sigma(R^b)) \not\Leftrightarrow \text{false}$ $(\mathcal{Q}_1)$ then
8. return $R^* \equiv \bigvee_{i=0}^{b-1} R^i \ \vee \ \exists k \geq 0 \ . \ \bigvee_{i=0}^{c-1} \pi(k \cdot \Lambda + \sigma(R^b)) \circ R^i$
9. else if exists $k \geq 0 : \rho(k \cdot \Lambda + \sigma(R^b)) \circ R^c \Leftrightarrow \text{false}$ $(\mathcal{Q}_2)$ then
10. let $k_0 = \min\{k \mid \rho(k \cdot \Lambda + \sigma(R^b)) \circ R^c \Leftrightarrow \text{false}\}$
11. if forall $k \in [0, k_0 - 1] : \rho(k \cdot \Lambda + \sigma(R^b)) \circ R^c \Leftrightarrow \rho((k+1) \cdot \Lambda + \sigma(R^b))$ then
12. return $R^* \equiv \bigvee_{i=0}^{b-1} R^i \ \vee \ \bigvee_{k=0}^{k_0-1} \bigvee_{i=0}^{c-1} \rho(k \cdot \Lambda + \sigma(R^b)) \circ R^i$
13. endif
14. endif
15. endfor
16. endfor

Fig. 1: Transitive Closure Algorithm

Otherwise, if $\mathcal{Q}_1$ does not hold, there are two possibilities: either (i) Λ is actually not the first rate of the sequence $\{\sigma(R^i)\}_{i=0}^{\infty}$ for given $b \geq 0$ and $c > 0$, or (ii) the relation is not ω-consistent. In the first case, we need to reiterate with another prefix-period pair, which will give us another candidate Λ.

In the second case, R^m becomes inconsistent, for some $m > 0$ – in this case the computation of its transitive closure is possible, in principle, by taking the disjunction of all powers of R up to m. However, in practice this may take a long time, if m is large. In order to speed up the computation, we check whether:

- $\rho(k \cdot \Lambda + \sigma(R^b)) \circ R^c$ is inconsistent (line 9); the existential query $\mathcal{Q}_2$ (and namely finding the smallest value for which it holds) is dealt with in Section 4, specifically for the classes of difference bounds and octagonal relations.
- R is periodic with first rate Λ between 0 and $k_0 - 1$ (line 11), where k_0 is the smallest k satisfying the first point (line 10).

If both conditions above hold, then $m = (k_0 + 1)c + b$ is the smallest value for which R^m becomes inconsistent, and moreover, R is periodic with rate Λ between 0 and m. If this is the case, we compute the transitive closure using the period Λ and return (line 12). The following theorem can be proved along the lines of the discussion above:

Theorem 1. *If R is an ultimately periodic relation, the algorithm in Fig. 1 eventually terminates and returns the transitive closure of R.*

4 Some Ultimately Periodic Classes of Arithmetic Relations

This section is dedicated to the application of the transitive closure computation algorithm from the previous section (Fig. 1) to three classes of arithmetic relations, for which the transitive closure is Presburger-definable: difference bounds relations [8], octagonal relations [7], and finite monoid affine transformations [5].

In order to apply the transitive closure computation method, one needs to address two issues. First, the class of relations considered needs to be proved ultimately periodic (for else, our algorithm is not guaranteed to terminate). The proofs rely mostly on the fact that any matrix A is ultimately periodic in $\mathbb{T}$ [34] (see Section 2 for the definition of ultimately periodic matrices).

Second, the queries $\mathcal{Q}_1$ and $\mathcal{Q}_2$ (Fig. 1) need to be answered efficiently, by avoiding excessive calls to external decision procedures. In theory, all these queries can be expressed in Presburger arithmetic, for the classes of difference bounds constraints, octagons and affine transformations, yet in practice we would like to avoid as much as possible using Presburger solvers, due to reasons of high complexity. For the classes of difference bounds and octagons, we give direct decision methods for handling these queries. The class of affine transformations without guards can also be dealt with by simply checking equivalence between Diophantine systems, whereas the general case still needs to be handled by a Presburger solver.

4.1 Difference Bounds Constraints

Let $\mathbf{x} = \{x_1, x_2, ..., x_N\}$ be a set of variables ranging over $\mathbb{Z}$.

Definition 3. *A formula $\phi(\mathbf{x})$ is a* <u>*difference bounds constraint*</u> *if it is equivalent to a finite conjunction of atomic propositions of the form $x_i - x_j \leq a_{ij}$, $1 \leq i, j \leq N, i \neq j$, where $a_{ij} \in \mathbb{Z}$.*

For example, $x = y + 5$ is a difference bounds constraint, as it is equivalent to $x - y \leq 5 \ \wedge \ y - x \leq -5$. Let $\mathcal{R}_{db}$ denote the class of difference bound relations. Difference bounds constraints are alternatively represented as matrices or, equivalently, weighted graphs.

Given a difference bounds constraint ϕ, a <u>difference bounds matrix</u> (DBM) representing ϕ is a matrix $M_\phi \in \mathbb{T}^{N \times N}$ such that $(M_\phi)_{ij} = a_{ij}$, if $x_i - x_j \leq a_{ij}$ is an atomic proposition in ϕ, and ∞, otherwise. Dually, if $M \in \mathbb{T}^{N \times N}$ is a DBM, the corresponding difference bounds constraint is $\Delta_M \equiv \bigwedge_{M_{ij} < \infty} x_i - x_j \leq M_{ij}$.

A DBM M is said to be consistent if and only if its corresponding constraint φ_M is consistent. An <u>elementary path</u> in a DBM M is a sequence of indices $1 \leq i_1, i_2, \ldots, i_k \leq N$, where $i_{1,\ldots,k-1}$ are pairwise distinct, such that $M_{i_j i_{j+1}} < \infty$, for all $1 \leq j < k$. An elementary path is called an <u>elementary cycle</u> if moreover $i_1 = i_k$. An elementary cycle is said to be <u>strictly negative</u> if $\sum_{j=1}^{k-1} M_{i_j i_{j+1}} < 0$. A DBM M is inconsistent if and only if it has a strictly negative elementary cycle – a proof can be found in [15]. The next definition gives a canonical form for consistent DBMs.

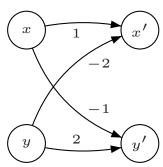
Definition 4. *A consistent DBM $M \in \mathbb{T}^{N \times N}$ is said to be* <u>*closed*</u> *if and only if $M_{ii} = 0$ and $M_{ij} \leq M_{ik} + M_{kj}$, for all $1 \leq i, j, k \leq N$.*

Given a consistent DBM M, we denote by M^* the (unique) closed DBM such that $\varphi_M \Leftrightarrow \varphi_{M^*}$. It is well-known that, if M is consistent, then M^* is unique, and can be computed from M in time $\mathcal{O}(N^3)$, by the classical Floyd-Warshall algorithm. Moreover, if M is a consistent DBM, we have, for all $1 \leq i, j \leq N$:

$$M^*_{ij} = \min \left\{ \sum_{l=0}^{k-1} M_{i_l i_{l+1}} \; i = i_0 \dots i_{k-1} = j \text{ is an elementary path in } M \right\} \tag{1}$$

The closed form of DBMs is needed for the elimination of existentially quantified variables – if ϕ is a difference bounds constraint, then $\exists x \,.\, \phi$ is also a difference bounds constraint [29]. Consequently, we have that the class of difference bounds relations is closed under relational composition: $R_1(\mathbf{x}, \mathbf{x}') \circ R_2(\mathbf{x}, \mathbf{x}') \equiv \exists \mathbf{y} \,.\, R_1(\mathbf{x}, \mathbf{y}) \wedge R_2(\mathbf{y}, \mathbf{x}')$.

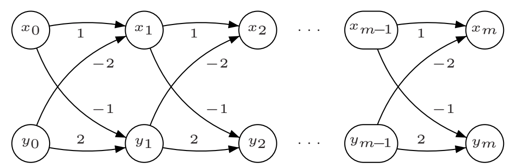
Running Example Let $R(x, y, x', y') \equiv (x - x' \leq 1) \wedge (x - y' \leq -1) \wedge (y - x' \leq -2) \wedge (y - y' \leq 2)$ be a difference bounds relation. Figure 2a shows the graph representation $\mathcal{G}_R$, Figure 2b the canonic matrix representation $M^*_R = \sigma(R)^*$ of R. Computation of the m-th power amounts to computation of shortest paths between extremal vertices of the graph $\mathcal{G}^m_R$, the m-times unfolding of $\mathcal{G}_R$, shown in Figure 2c.



(a) $\mathcal{G}_R$

$$\begin{array}{c} \\ x \\ y \\ x' \\ y' \end{array} \begin{array}{c} \begin{array}{cccc} x & y & x' & y' \end{array} \\ \begin{pmatrix} 0 & \infty & 1 & -1 \\ \infty & 0 & -2 & 2 \\ \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 0 \end{pmatrix} \end{array}$$

(b) M^*_R



(c) $\mathcal{G}^m_R$

Fig. 2: Graph and matrix representation of a relation. Graph unfolding.

Difference Bounds Relations are Ultimately Periodic Given a consistent difference bounds relation $R(\mathbf{x}, \mathbf{x}') \in \mathcal{R}_{db}$, let $\sigma(R) = M_R \in \mathbb{T}^{2N \times 2N}$ be the characteristic DBM of R, and for any $M \in \mathbb{T}^{2N \times 2N}$, let $\rho(M) = \Delta_M \in \mathcal{R}_{db}$ be the difference bounds relation corresponding to R. Clearly, $\rho(\sigma(R)) \Leftrightarrow R$, as required by Def. 2.

With these definitions, the algorithm in Fig. 1 will return the transitive closure of a difference bounds relation R, provided that the sequence $\{\sigma(R^i)\}_{i=0}^{\infty}$ is ultimately periodic. If R is not ω-consistent then, by Def. 2, it is ultimately periodic. We consider henceforth the case where R is ω-consistent, i.e. $\sigma(R^i) = M_{R^i}$, for all $i \geq 0$.

For a difference bounds relation R, we define the directed graph $\mathcal{G}_R$, whose set of vertices is the set $\mathbf{x} \cup \mathbf{x}'$, and in which there is an edge from x_i to x_j labeled α_{ij} if and only if the atomic proposition $x_i - x_j \leq \alpha_{ij}$ occurs in R. Clearly, M_R is the incidence matrix of $\mathcal{G}_R$.

Next, we define the concatenation of $\mathcal{G}_R$ with itself as the disjoint union of two copies of $\mathcal{G}_R$, in which the $\mathbf{x}'$ vertices of the second copy overlap with the $\mathbf{x}$ vertices of the first copy. Then R^m corresponds to the graph $\mathcal{G}^m_R$, obtained by concatenating the graph of R to itself $m > 0$ times. Since $\mathcal{R}_{db}$ is closed under relational composition, then $R^m \in \mathcal{R}_{db}$, and moreover we have:

$$\bigwedge_{1 \le i,j \le N} x_i - x_j \le \min\{x_i^0 \to x_j^0\} \wedge x'_i - x'_j \le \min\{x_i^m \to x_j^m\} \wedge \\ x_i - x'_j \le \min\{x_i^0 \to x_j^m\} \wedge x'_i - x_j \le \min\{x_i^m \to x_j^0\}$$

where $\min\{x_i^p \to x_j^q\}$ is the minimal weight of all paths between the extremal vertices x_i^p and x_j^q in $\mathcal{G}_R^m$, for $p, q \in \{0, m\}$. In other words, we have the equalities from Fig. 3 (a), for all $1 \le i, j \le N$.

$$\begin{array}{rcl}
(\sigma(R^m))_{i,j} & = & \min\{x_i^0 \to x_j^0\} \\
(\sigma(R^m))_{i+N,j+N} & = & \min\{x_i^m \to x_j^m\} \\
(\sigma(R^m))_{i,j+N} & = & \min\{x_i^0 \to x_j^m\} \\
(\sigma(R^m))_{i+N,j} & = & \min\{x_i^m \to x_j^0\} \\
 & (a) &
\end{array}
\qquad
\begin{array}{rcl}
\min\{x_i^0 \to x_j^0\} & = & (\mathcal{M}_R^{m+2})_{I_{ef}(x_i),F_{ef}(x_j)} \\
\min\{x_i^m \to x_j^m\} & = & (\mathcal{M}_R^{m+2})_{I_{eb}(x_i),F_{eb}(x_j)} \\
\min\{x_i^0 \to x_j^m\} & = & (\mathcal{M}_R^{m+2})_{I_{of}(x_i),F_{of}(x_j)} \\
\min\{x_i^m \to x_j^0\} & = & (\mathcal{M}_R^{m+2})_{I_{ob}(x_i),F_{ob}(x_j)} \\
 & (b) &
\end{array}$$

Fig. 3

As proved in [8], the paths between x_i^p and x_j^q, for arbitrary $1 \le i, j \le N$ and $p, q \in \{0, m\}$, can be seen as words (over a finite alphabet of subgraphs of $\mathcal{G}_R^m$) recognized by a finite weighted automaton (with at most 5^N states). For the sake of completeness, its definition follows.

Definition of Zigzag Automata Let $\mathbf{x} = \{x_1, \ldots, x_N\}$ be a set of variables. In the following, we will work with a more convenient (yet equivalent) form of difference bounds relations: all constraints of the form $x - y \le \alpha$ are replaced by $x - t' \le \alpha \wedge t' - y \le 0$, and all constraints of the form $x' - y' \le \alpha$ are replaced by $x' - t \le \alpha \wedge t - y' \le 0$, by introducing fresh variables $t \notin \mathbf{x}$. In other words, we can assume without loss of generality that the constraint graph corresponding to R ($\mathcal{G}_R$) is <u>bipartite</u>, i.e. it does only contain edges from $\mathbf{x}$ and $\mathbf{x}'$ and viceversa. We denote the m-times composition of $\mathcal{G}_R$ with itself as $\mathcal{G}_R^m$, and the i-th step nodes of $\mathcal{G}_R^m$, for $0 \le i \le m$, with $\mathbf{x}^i$.

Intuitively, a path π between, say, x^0 and x^m, with $x, y \in \mathbf{x}$ is represented by a word w of length m, as follows: the w_i symbol represents <u>simultaneously</u> all edges of π that involve only nodes from $\mathbf{x}^i \cup \mathbf{x}^{i+1}$, $0 \le i < m$. Since we assumed that $\mathcal{G}_R^m$ is bipartite, it is easy to see that, for a path from x^0 to y^m, coded by a word w, the number of times the w_i symbol is traversed by the path is odd, whereas for a path from x^0 to y^0, or from x^m to y^m, this number is even. Hence the names of <u>even</u> and <u>odd automata</u>.

Given a difference bound relation R, the <u>even alphabet</u> of R, denoted as Σ_R^e, is the set of all graphs satisfying the following conditions, for each $G \in \Sigma_R^e$:

1. the set of nodes of G is $\mathbf{x} \cup \mathbf{x}'$,

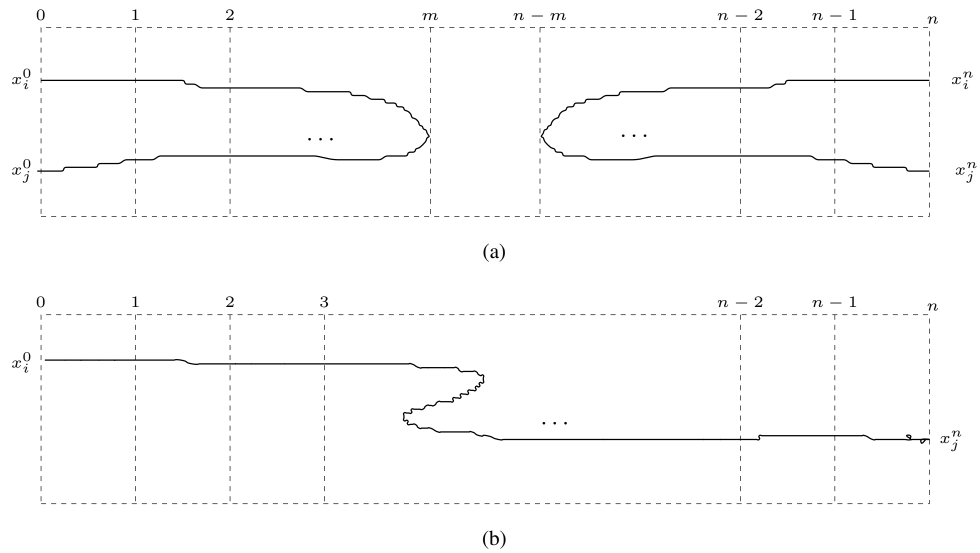


Fig. 4: Runs of Even and Odd Automata

2. for any $x, y \in \mathbf{x} \cup \mathbf{x}'$, there is an edge labeled α from x to y, only if $x - y \leq \alpha$ occurs in φ.
3. the in-degree and out-degree of each node are at most one.
4. the number of edges from $\mathbf{x}$ to $\mathbf{x}'$ equals the number of edges from $\mathbf{x}'$ to $\mathbf{x}$.

The odd alphabet of R, denoted by Σ_R^o, is defined in the same way, with the exception of the last condition, which becomes:

(4) the difference between the number of edges from $\mathbf{x}$ to $\mathbf{x}'$ and the number of edges from $\mathbf{x}'$ to $\mathbf{x}$ is either 1 or -1.

Let $\Sigma_R = \Sigma_R^e \cup \Sigma_R^o$. Notice that, the number of edges in all symbols of Σ_R^e is even, while the number of edges in all symbols of Σ_R^o is odd. The label of G is the sum of the weights that occur on its edges. Clearly, the weight of a path through $\mathcal{G}_R^m$ is the weight of the word it is represented by. We denote by $\omega(w)$ the weight of a word $w \in {\Sigma_R}^*$.

We are now ready for the definition of automata recognizing words that represent encodings of paths from $\mathcal{G}_R^m$. The even automaton recognizes paths that start and end on the same side of $\mathcal{G}_R^m$ i.e., either paths from x_i^0 to x_j^0, or from x_i^m to x_j^m, for some $1 \leq i, j \leq N$, respectively. We call the automata recognizing paths from x_i^0 to x_j^0 forward even automata, and the ones recognizing paths from x_i^m to x_j^m backward even automata (Figure 4 (a)). The odd automata recognize paths from one side of $\mathcal{G}_R^m$ to another. The automata recognizing paths from x_i^0 to x_j^m are called forward odd automata, whereas the ones recognizing paths from x_i^m to x_j^0 are called backward odd automata (Figure 4 (b)).

The even and odd automata share the same transition table, whereas the input alphabet is Σ_R^e for the former, and Σ_R^o for the latter. More precisely, we define the common transition table as $T_R = \langle Q, \Delta \rangle$, where:

$Q = \{l, r, lr, rl, \bot\}^N \cup \{I_\bullet(x) \mid x \in \mathbf{x}\} \cup \{F_\bullet(x) \mid x \in \mathbf{x}\}, \bullet \in \{of, ob, ef, eb\}$, and

$$\Delta = \Delta_g \cup \Delta_l \bigcup_{1 \le i,j \le N} (\Delta_{ij}^{ef} \cup \Delta_{ij}^{eb} \cup \Delta_{ij}^{of} \cup \Delta_{ij}^{ob})$$

We now define transition sets $\Delta_g, \Delta_l, \Delta_{ij}^{ef}, \Delta_{ij}^{eb}, \Delta_{ij}^{of}, \Delta_{ij}^{ob}$. There is a transition $\langle q_1 \dots q_N \rangle \xrightarrow{G} \langle q'_1, \dots, q'_N \rangle$ in Δ_g if and only if the following conditions hold, for all $1 \le i \le N$:

- $q_i = l$ iff G has one edge whose destination is x_i, and no other edge involving x_i.
- $q'_i = l$ iff G has one edge whose source is x'_i, and no other edge involving x'_i.
- $q_i = r$ iff G has one edge whose source is x_i, and no other edge involving x_i.
- $q'_i = r$ iff G has one edge whose destination is x'_i, and no other edge involving x'_i.
- $q_i = lr$ iff G has exactly two edges involving x_i, one having x_i as source, and another as destination.
- $q'_i = rl$ iff G has exactly two edges involving x'_i, one having x'_i as source, and another as destination.
- $q'_i \in \{lr, \bot\}$ iff G has no edge involving x'_i.
- $q_i \in \{rl, \bot\}$ iff G has no edge involving x_i.

Note that some even paths in $\mathcal{G}_R^m$ may be of length strictly less than m. Since we want to recognize these path by runs of length $m+2$, we need some extra self-loop transitions. These are defined in Δ_l:

$$\Delta_l = \{F_{ef}(x) \xrightarrow{\epsilon} F_{ef}(x) \mid x \in \mathbf{x}\} \cup \{I_{eb}(x) \xrightarrow{\epsilon} I_{eb}(x) \mid x \in \mathbf{x}\}, \text{ where}$$

ϵ is a special symbol of weight 0. Finally, we define Δ_{ij}^{ef}, Δ_{ij}^{eb}, Δ_{ij}^{of}, and Δ_{ij}^{ob}.

$$\Delta_{ij}^{ef} = \begin{cases} \{I_{ef}(x_i) \to q \mid q_i = r, q_j = l \text{ and } q_h \in \{lr, \bot\}, 1 \le h \le N, h \notin \{i,j\}\} & \text{if } i \ne j \\ \{I_{ef}(x_i) \to q \mid q_i = q_j = lr \text{ and } q_h \in \{lr, \bot\}, 1 \le h \le N, h \ne i\} & \text{otherwise} \end{cases}$$

$$\bigcup \{q \to F_{ef}(x_j) \mid q \in \{rl, \bot\}^N\}$$

$$\Delta_{ij}^{eb} = \begin{cases} \{q \to F_{eb}(x_i) \mid q_i = r, q_j = l \text{ and } q_h \in \{lr, \bot\}, 1 \le h \le N, h \notin \{i,j\}\} & \text{if } i \ne j \\ \{q \to F_{eb}(x_i) \mid q_i = q_j = lr \text{ and } q_h \in \{lr, \bot\}, 1 \le h \le N, h \ne i\} & \text{otherwise} \end{cases}$$

$$\bigcup \{I_{eb}(x_j) \to q \mid q \in \{rl, \bot\}^N\}$$

$$\begin{aligned} \Delta_{ij}^{of} = \quad & \{I_{of}(x_i) \to q \mid q_i = r \text{ and } q_h \in \{lr, \bot\}, 1 \le h \le N, h \ne i\} \\ & \bigcup \{q \to F_{of}(x_j) \mid q_j = r \text{ and } q_h \in \{rl, \bot\}, 1 \le h \le N, h \ne j\} \end{aligned}$$

$$\begin{aligned}\Delta_{ij}^{ob} = \quad & \{I_{ob}(x_i) \to q \mid q_i = l \text{ and } q_h \in \{lr, \perp\}, 1 \le h \le N, h \ne i\} \\ & \bigcup \{q \to F_{ob}(x_j) \mid q_j = l \text{ and } q_h \in \{rl, \perp\}, 1 \le h \le N, h \ne j\}\end{aligned}$$

With the above definitions, we can define the even forward, even backward, odd forward, and odd backward automata.

$$A_{ij}^{ef} = \langle T_R, I_{ef}(x_i), F_{ef}(x_j)\rangle$$

$$A_{ij}^{eb} = \langle T_R, I_{eb}(x_i), F_{eb}(x_j)\rangle$$

$$A_{ij}^{of} = \langle T_R, I_{of}(x_i), F_{of}(x_j)\rangle$$

$$A_{ij}^{ob} = \langle T_R, I_{ob}(x_i), F_{ob}(x_j)\rangle$$

Recall that $\mathcal{G}_R^m$ denotes the constraint graph corresponding to R^m, obtained by concatenating the constraint graph of R to itself $m > 0$ times. We say that a path in $\mathcal{G}_R^m$ stretches between k and l, for some $k \le l$, if the path contains at least one node from $\mathbf{x}^i$, for each $k \le i \le l$. Intuitively, all paths from x_i^0 to x_j^0 in $\mathcal{G}_R^m$ are recognized by the automaton A_{ij}^{ef}, paths from x_i^m to x_j^m by A_{ij}^{eb} (Figure 4 (a)), paths from x_i^0 to x_j^m by A_{ij}^{of}, and paths from x_i^m to x_j^0 by A_{ij}^{ob} (Figure 4 (b)). The following are proved in [8][4]:

Lemma 4 ([8]). *Suppose that $\mathcal{G}_R^m$ does not have cycles of negative weight, for some $m > 0$. Then, for any $1 \le i, j \le N$, $i \ne j$, the following hold:*

1. *A_{ij}^{ef} has an accepting run of length at most m if and only if there exists a path in $\mathcal{G}_R^m$, from x_i^0 to x_j^0, that stretches between 0 and some $n \le m$. Moreover, the minimal weight among all paths from x_i^0 to x_j^0, stretching from 0 to some $n \le m$, equals the minimal weight among all accepting runs of A_{ij}^{ef} of length at most m.*
2. *A_{ij}^{eb} has an accepting run of length at most m if and only if there exists a path in $\mathcal{G}_R^m$, from x_i^m to x_j^m, that stretches between $m - n$ and m, for some $1 \le n < m$. Moreover, the minimal weight among all paths from x_i^m to x_j^m, stretching from $m - n$ to m, equals the minimal weight among all accepting runs of A_{ij}^{eb}, of length at most m.*
3. *A_{ij}^{of} has an accepting run of length m if and only if there exists a path in $\mathcal{G}_R^m$, from x_i^0 to x_j^m. Moreover, the minimal weight among all paths from x_i^0 to x_j^m equals the minimal weight among all accepting runs of length m.*
4. *A_{ij}^{ob} has an accepting run of length n if and only if there exists a path in $\mathcal{G}_R^m$, from x_i^m to x_j^0. Moreover, the minimal weight among all paths from x_i^m to x_j^0 equals the minimal weight among all accepting runs of length m.*

[4] Lemmas 4.3, 4.4, 4.6 and 4.7.

Let $\mathcal{M}_R$ be the incidence matrix of T_R. By the construction of $\mathcal{M}_R$, for each variable $x \in \mathbf{x}$, there are eight indices, denoted as $I_{of}(x), I_{ob}(x), I_{ef}(x), I_{eb}(x), F_{of}(x)$, $F_{ob}(x), F_{ef}(x), F_{eb}(x) \in \{1, \ldots, 5^N + 8N\}$, such that all relations from Fig. 3 (b) hold, for all $1 \leq i, j \leq N$. It is easy to see (as an immediate consequence of the interpretation of the matrix product in $\mathbb{T}$) that, for any $m > 0$, the matrix $\mathcal{M}_R^{m+2}$ gives the minimal weight among all paths, of length m[5], between x_i^p and x_j^q, for any $1 \leq i, j \leq N$ and $p, q \in \{0, m\}$. But the sequence $\{\mathcal{M}_R^m\}_{m=0}^{\infty}$ is ultimately periodic, since every matrix is ultimately periodic in $\mathbb{T}$ [34]. By equating the relations from Fig. 3 (a) with the ones from Fig. 3 (b), we obtain that the sequence $\{\sigma(R^m)\}_{m=0}^{\infty}$ is ultimately periodic as well.

In conclusion, the algorithm from Fig. 1 will terminate on difference bounds relations. Moreover, the result is formula in Presburger arithmetic. This also simplifies the proof that transitive closures of difference bounds relations are Presburger definable, from [8], since the minimal paths of length m within the weighted automaton recognizing the paths of $\mathcal{G}_R^m$ correspond in fact to elements of the m-th power of $\mathcal{M}_R$ (the incidence matrix of the automaton) in $\mathbb{T}$.

Running Example Let us show the construction of the zigzag automaton for the relation $R \equiv (x - x' \leq 1) \wedge (x - y' \leq -1) \wedge (y - x' \leq -2) \wedge (y - y' \leq 2)$. Notice that there are only forward odd paths, i.e. paths from $\mathbf{x_0}$ to $\mathbf{x}_m$ in $\mathcal{G}_R^m$ for any $m \geq 1$. The transition table $T_R = \langle Q, \Delta \rangle$ of the zigzag automaton is depicted in Figure 5 (isolated states, such as (r, l), have been removed). For instance, the automaton $A_{xy}^{ef} = \langle T_R, I_{of}(x), O_{of}(x) \rangle$ recognizes a run of length $m+2$ with weight w if and only if there is a path from x_0 to x^m in $\mathcal{G}_R^m$ of length m and with weight w. There are four such paths in $\mathcal{G}_R^3$ and the Figure 6 shows the corresponding runs of the zigzag automaton.

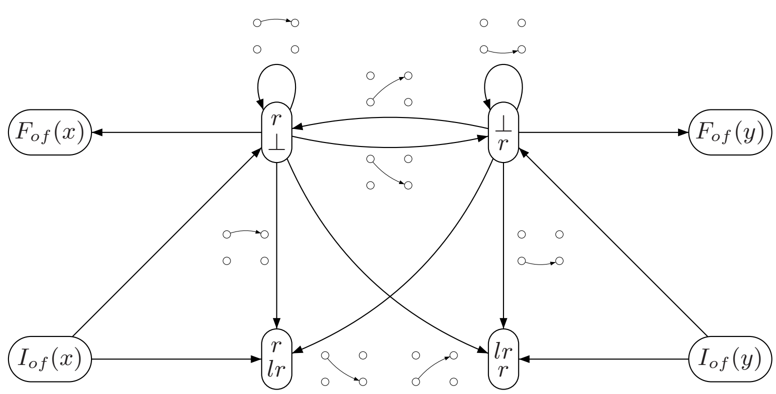


Fig. 5: Zigzag automaton

Checking ω-Consistency and Inconsistency of Difference Bounds Relations For a difference bounds relation $R(\mathbf{x}, \mathbf{x}') \in \mathcal{R}_{db}$ and a matrix $\Lambda \in \mathbb{T}^{2N \times 2N}$, we give

[5] The offset of 2 is needed due to use of the special initial and final transitions.

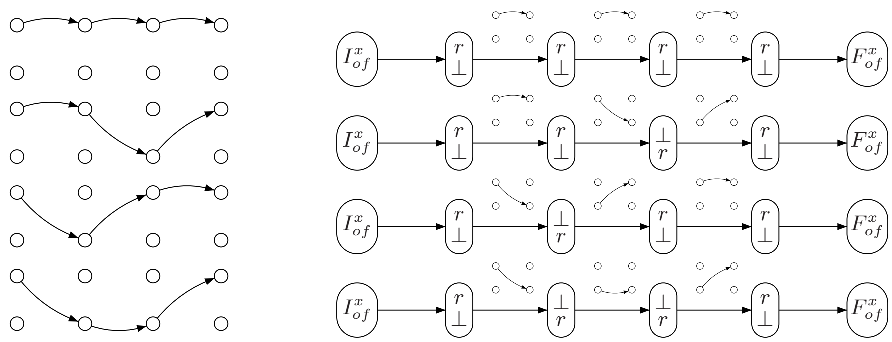


Fig. 6: Runs

methods to decide the queries $\mathcal{Q}_1$ and $\mathcal{Q}_2$ (lines 7 and 9 in Fig. 1) efficiently. To this end, we consider the class of parametric difference bounds relations. From now on, let $k \notin \mathbf{x}$ be a variable interpreted over $\mathbb{N}_+$.

Definition 5. *A formula $\phi(\mathbf{x}, k)$ is a* <u>*parametric difference bounds constraint*</u> *if it is equivalent to a finite conjunction of atomic propositions of the form $x_i - x_j \leq a_{ij} \cdot k + b_{ij}$, for some $1 \leq i, j \leq N$, $i \neq j$, where $a_{ij}, b_{ij} \in \mathbb{Z}$.*

The class of parametric difference bounds relations with parameter k is denoted as $\mathcal{R}_{db}(k)$. A parametric difference bounds constraint $\phi(k)$ can be represented by a matrix $M_\phi[k]$ of linear terms, where $(M_\phi[k])_{ij} = a_{ij} \cdot k + b_{ij}$ if $x_i - x_j \leq a_{ij} \cdot k + b_{ij}$ occurs in ϕ, and ∞ otherwise. Dually, a matrix $M[k]$ of linear terms corresponds to the formula $\Delta_M(k) \equiv \bigwedge_{M[k]_{ij} \neq \infty} x_i - x_j \leq M[k]_{ij}$. With these considerations, we define $\pi(M[k]) = \Delta_M(k)$. Clearly, we have $\pi(k \cdot \Lambda + \sigma(R^b)) \in \mathcal{R}_{db}(k)$, for $R \in \mathcal{R}_{db}$, $b \geq 0$ and $\Lambda \in \mathbb{T}^{2N \times 2N}$.

Parametric DBMs do not have a closed form, since in general, the minimum of two linear terms in k (for all valuations of k) cannot be expressed again as a linear term. According to (1), one can define the closed form of a parametric DBM as a matrix of terms of the form $\min\{a_i \cdot k + b_i\}_{i=1}^m$, for some $a_i, b_i \in \mathbb{Z}$ and $m > 0$. Then the query $\mathcal{Q}_1$ can be written as a conjunction of formulae of the form $\forall k > 0 \,.\, \min\{a_i \cdot k + b_i\}_{i=1}^m = a_0 \cdot k + b_0$. The following lemma gives a way to decide the validity of such formulae:

Lemma 5. *Given $\ell, a_0, a_1, \ldots, a_m,\ b_0, b_1, \ldots, b_m \in \mathbb{Z}$, the following are equivalent:*

1. $\forall k \geq \ell \,.\, \min\{a_i \cdot k + b_i\}_{i=1}^m = a_0 \cdot k + b_0$
2. $\bigvee_{i=1}^m (a_i = a_0 \wedge b_i = b_0) \ \wedge\ \bigwedge_{j=1}^m (a_0 \leq a_j \wedge a_0 \cdot \ell + b_0 \leq a_j \cdot \ell + b_j)$

Proof. We assume w.l.o.g. that all terms $a_i \cdot k + b_i,\ i = 1, \ldots, m$ are distinct.
"1 ⇒ 2" For infinitely many $k \geq K$ we have $\min\{a_i \cdot k + b_i\}_{i=1}^m = a_0 \cdot k + b_0$. Since the set of terms $\{a_i \cdot k + b_i\}_{i=1}^m$ is finite, there exists $k_1 < k_2$ and some $i = 1, \ldots, m$ such that $a_i \cdot k_1 + b_i = a_0 \cdot k_1 + b_0$ and $a_i \cdot k_2 + b_i = a_0 \cdot k_2 + b_0$. Hence we have:

$$(a_i - a_0) \cdot k_1 = b_0 - b_i$$
$$(a_i - a_0) \cdot k_2 = b_0 - b_i$$

and the only possibility is when $a_i = a_0$ and $b_i = b_0$. For the second part, we have $a_0 \cdot k + b_0 \leq a_i \cdot k + b_i$, for all $k \geq K$, therefore:

- $a_0 \cdot K + b_0 \leq a_i \cdot K + b_i$ (the case $k = K$)
- $b_0 - b_i \leq (a_i - a_0) \cdot k$, for all $k \geq K$, therefore $a_i - a_0 \geq 0$ (the term $(a_i - a_0) \cdot k$ is bounded from below, hence it cannot decrease infinitely often)

"$2 \Rightarrow 1$" Since $\bigvee_{i=1}^{m} a_i = a_0 \wedge b_i = b_0$, we have $a_0 \cdot k + b_0 \in \{a_i \cdot k + b_i\}_{i=1}^{m}$, for all $k \geq K$. By $a_0 \leq a_i \wedge a_0 \cdot K + b_0 \leq a_i \cdot K + b_i$ we obtain $a_0 \cdot k + b_0 \leq a_i \cdot k + b_i$, for all $k \geq K$. Therefore $\min\{a_i \cdot k + b_i\}_{i=1}^{m} = a_0 \cdot k + b_0$, for all $k \geq K$.

□

In analogy to the non-parametric case, the inconsistency of a parametric difference bounds constraint $\phi(k)$ amounts to the existence of a strictly negative elementary cycle in $M_\phi[k]$, for some valuation $k \in \mathbb{N}_+$. We are also interested in finding the smallest value for which such a cycle exists. The following lemma gives this value.

Lemma 6. *Let $\phi(\mathbf{x}, k)$ be a parametric difference bounds constraint and $M_\phi[k]$ be its associated matrix. For some $a_{ij}, b_{ij} \in \mathbb{Z}$, let $\{a_{ij} \cdot k + b_{ij}\}_{j=1}^{m_i},\ i = 1, \ldots, 2N$ be the set of terms denoting weights of elementary cycles going through i. Then ϕ is inconsistent for some $\ell \in \mathbb{N}$ and $k \geq \ell$ if and only if there exists $1 \leq i \leq 2N$ and $1 \leq j \leq m_i$ such that either (i) $a_{ij} < 0$ or (ii) $a_{ij} \geq 0 \wedge a_{ij} \cdot \ell + b_{ij} < 0$ holds. Moreover, the smallest value for which ϕ becomes inconsistent is $\min_{i=1}^{2N}\{\min_{j=1}^{m_i} \gamma_{ij}\}$, where $\gamma_{ij} = \max(\ell, \lfloor -\frac{b_{ij}}{a_{ij}} \rfloor + 1)$, if $a_{ij} < 0$, $\gamma_{ij} = \ell$, if $a_{ij} \geq 0\ \wedge\ a_{ij} \cdot \ell + b_{ij} < 0$, and $\gamma_{ij} = \infty$, otherwise.*

Proof. ϕ is inconsistent iff there exists $k \geq K$ such that $a_{ij} \cdot k + b_{ij} < 0$, for some $1 \leq i \leq 2n$ and $1 \leq j \leq m_i$. If $a_{ij} < 0$, then $k > -\frac{b_{ij}}{a_{ij}}$, hence $k \geq \lfloor -\frac{b_{ij}}{a_{ij}} \rfloor + 1$. Since $k \geq K$, we have $k_0 = \max(K, \lfloor -\frac{b_{ij}}{a_{ij}} \rfloor + 1)$. Else, if $a_{ij} \geq 0$ and $a_{ij} \cdot K + b_{ij} \geq 0$, we have $a_{ij} \cdot k + b_{ij} \geq 0$, for all $k \geq K$, contradiction. The only remaining case is $a_{ij} \geq 0 \wedge a_{ij} \cdot K + b_{ij} < 0$, where we chose $k_0 = K$.

□

Running Example We demonstrate the main steps of the algorithm 1 for the difference bounds relation $R \equiv (x - x' \leq 1) \wedge (x - y' \leq -1) \wedge (y - x' \leq -2) \wedge (y - y' \leq 2)$. The condition at line 6 is first satisfied for $(b, c) = (2, 2)$ and hence the candidate rate Λ_0 is the one shown in Figure 7.

In order to answer the universal query at line 7,

$$k \geq 0\ :\ \rho(k \cdot \Lambda + \sigma(R^b)) \circ R^c \Leftrightarrow \rho((k+1) \cdot \Lambda + \sigma(R^b)) \not\Leftrightarrow \text{false}\ (\mathcal{Q}_1)$$

we need to compute relational composition of the parametric DB relation $\rho(k \cdot \Lambda + \sigma(R^b))$, whose matrix and graph encodings are in figures 8a and 8b, with DB relation R^c (figures 7 and 8c) and check whether the result of the composition is equivalent to the right-hand side (figure 9). The composition is done by the classical Floyd-Warshall algorithm, which computes shortest paths between extremal vertices of the graph 10 (obtained by concatenating graph $\mathcal{G}_{\sigma(k \cdot \Lambda_0 + M_{R^b})}$ with $\mathcal{G}_{R^c}$). For instance, the shortest path between x and y' is $min\{-3–3k–0, -1–3k–3\} = -4–3k$, hence $x' - y' \leq -4–3k$.

When computing the min operation during Floyd-Warshall, we minimize the number of linear terms in each cell by erasing terms which are comparable and greater than some term in the cell. By doing so, we obtain a matrix which is identical to the one in figure 9a and hence the condition $\mathcal{Q}_1$ is satisfied.

The algorithm then returns the transitive closure at line 8:

$$\begin{aligned}
R^* &\equiv \bigvee_{i=0}^{1} R^i \ \vee\ \exists k \geq 0\ .\ \bigvee_{i=0}^{1} \pi(k \cdot \Lambda + \sigma(R^2)) \circ R^i \\
&\equiv (x' = x \wedge y' = y) \vee (x - x' \leq 1 \wedge x - y' \leq -1 \wedge y - x' \leq -2 \wedge y - y' \leq -2) \vee \\
&\quad \exists k \geq 0\ .\ (x - x' \leq -3k - 3 \wedge x - y' \leq -3k \wedge y - x' \leq -3k - 1 \wedge y - y' \leq -3k - 3) \vee \\
&\quad (x - x' \leq -3k - 2 \wedge x - y' \leq -3k - 4 \wedge y - x' \leq -3k - 5 \wedge y - y' \leq -3k - 2)
\end{aligned}$$

After quantifier elimination, we obtain

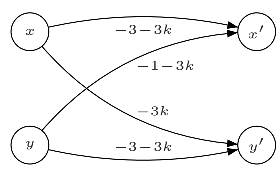
$$\begin{aligned}
R^* \equiv\ & (x' = x \wedge y' = y) \vee (x - x' \leq 1 \wedge x - y' \leq -1 \wedge y - x' \leq -2 \wedge y - y' \leq -2) \vee \\
& (x - x' \leq -3 \wedge x - y' \leq 0 \wedge y - x' \leq -1 \wedge y - y' \leq -3) \vee \\
& (x - x' \leq -2 \wedge x - y' \leq -4 \wedge y - x' \leq -5 \wedge y - y' \leq -2)
\end{aligned}$$

$$M_{R^b} = M_{R^c} = \begin{pmatrix} 0 & \infty & -3 & 0 \\ \infty & 0 & -1 & -3 \\ \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 0 \end{pmatrix} M_{R^{b+c}} = \begin{pmatrix} 0 & \infty & -6 & 3 \\ \infty & 0 & -4 & -6 \\ \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 0 \end{pmatrix} M_{R^{b+2c}} = \begin{pmatrix} 0 & \infty & -9 & 6 \\ \infty & 0 & -7 & -9 \\ \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 0 \end{pmatrix} \Lambda_0 = \begin{pmatrix} 0 & \infty & -3 & -3 \\ \infty & 0 & -3 & -3 \\ \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

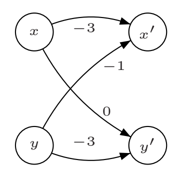
Fig. 7: Candidate rate Λ_0 for $(b, c) = (2, 2)$.

$$\begin{pmatrix} 0 & \infty & (-3 - 3k) & -3k \\ \infty & 0 & (-1 - 3k) & (-3 - 3k) \\ \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

(a) $k \cdot \Lambda_0 + M_{R^b}$



(b) $\mathcal{G}_{\sigma(k \cdot \Lambda_0 + M_{R^b})}$



(c) $\mathcal{G}_{R^c}$

Fig. 8: Left-hand side of $\mathcal{Q}_1$

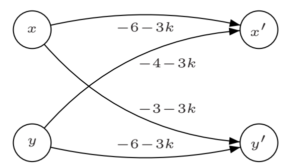
4.2 Octagons

Let $\mathbf{x} = \{x_1, x_2, ..., x_N\}$ be a set of variables ranging over $\mathbb{Z}$.

Definition 6. *A formula $\phi(\mathbf{x})$ is an* <u>*octagonal constraint*</u> *if it is equivalent to a finite conjunction of terms of the form $\pm x_i \pm x_j \leq a_{ij}$, $2x_i \leq b_i$, or $-2x_i \leq c_i$, where $a_{ij}, b_i, c_i \in \mathbb{Z}$ and $1 \leq i, j \leq N,\ i \neq j$.*

$$\begin{pmatrix} 0 & \infty & (-6-3k) & (-3-3k) \\ \infty & 0 & (-4-3k) & (-6-3k) \\ \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

(a) $(k+1) \cdot \Lambda_0 + M_{R^b}$



(b) $\mathcal{G}_{\sigma((k+1)\cdot\Lambda_0+M_{R^b})}$

Fig. 9: Right-hand side of $\mathcal{Q}_1$

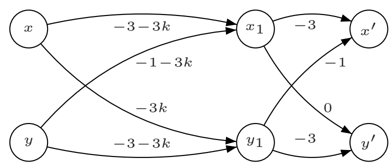


Fig. 10: Computing parametric composition $\sigma(k \cdot \Lambda_0 + M_{R^b}) \circ R^c$.

The class of octagonal relations is denoted by $\mathcal{R}_{oct}$ in the following. We represent octagons as difference bounds constraints over the set of variables $\mathbf{y} = \{y_1, y_2, \ldots, y_{2N}\}$, with the convention that y_{2i-1} stands for x_i and y_{2i} for $-x_i$, respectively. For example, the octagonal constraint $x_1 + x_2 = 3$ is represented as $y_1 - y_4 \leq 3 \wedge y_2 - y_3 \leq -3$. To handle the $\mathbf{y}$ variables in the following, we define $\bar{\imath} = i - 1$, if i is even, and $\bar{\imath} = i + 1$ if i is odd. Obviously, we have $\bar{\bar{\imath}} = i$, for all $i \in \mathbb{Z},\ i \geq 0$. We denote by $\overline{\phi}$ the difference bounds constraint $\phi[y_1/x_1, y_2/-x_1, \ldots, y_{2N-1}/x_N, y_{2N}/-x_N]$ over $\mathbf{y}$. The following equivalence relates ϕ and $\overline{\phi}$:

$$\phi(\mathbf{x}) \Leftrightarrow (\exists y_2, y_4, \ldots, y_{2N} \,.\, \overline{\phi} \wedge \bigwedge_{i=1}^{N} y_{2i-1} + y_{2i} = 0)[x_1/y_1, \ldots, x_N/y_{2N-1}] \quad (2)$$

An octagonal constraint ϕ is equivalently represented by the DBM $M_{\overline{\phi}} \in \mathbb{T}^{2N \times 2N}$, corresponding to $\overline{\phi}$. We say that a DBM $M \in \mathbb{T}^{2N \times 2N}$ is <u>coherent</u> iff $M_{ij} = M_{\bar{\jmath}\bar{\imath}}$ for all $1 \leq i, j \leq 2N$. This property is needed since any atomic proposition $x_i - x_j \leq a$, in ϕ can be represented as both $y_{2i-1} - y_{2j-1} \leq a$ and $y_{2j} - y_{2i} \leq a$, $1 \leq i, j \leq N$. Dually, a coherent DBM $M \in \mathbb{T}^{2N \times 2N}$ corresponds to the octagonal constraint Ω_M:

$$\bigwedge_{1 \leq i,j \leq N} (x_i - x_j \leq M_{2i-1,2j-1} \wedge x_i + x_j \leq M_{2i-1,2j} \wedge -x_i - x_j \leq M_{2i,2j-1}) \quad (3)$$

A coherent DBM M is said to be <u>octagonal-consistent</u> if and only if Ω_M is consistent.

Definition 7. *An octagonal-consistent coherent DBM $M \in \mathbb{T}^{2N \times 2N}$ is said to be <u>tightly closed</u> if and only if the following hold:*

1. $M_{ii} = 0,\ \forall 1 \leq i \leq 2N$
2. $M_{i\bar{\imath}}$ *is even*, $\forall 1 \leq i \leq 2N$
3. $M_{ij} \leq M_{ik} + M_{kj},\ \forall 1 \leq i, j, k \leq 2N$
4. $M_{ij} \leq \lfloor \frac{M_{i\bar{\imath}}}{2} \rfloor + \lfloor \frac{M_{\bar{\jmath}j}}{2} \rfloor,\ \forall 1 \leq i, j \leq 2N$

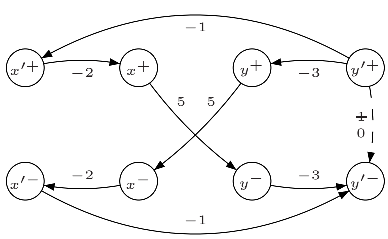
The following theorem from [3] provides an effective way of testing consistency and computing the tight closure of a coherent DBM. Moreover, it shows that the tight closure of a given DBM is unique and can also be computed in time $\mathcal{O}(N^3)$.

Theorem 2. *[3] Let $M \in \mathbb{T}^{2N\times 2N}$ be a coherent DBM. Then M is octagonal-consistent if and only if M is consistent and $\lfloor \frac{M_{i\bar{\imath}}}{2} \rfloor + \lfloor \frac{M_{\bar{\imath}i}}{2} \rfloor \geq 0$, for all $1 \leq i \leq 2N$. Moreover, the tight closure of M is the DBM $M^t \in \mathbb{T}^{2N\times 2N}$ defined as $M^t_{ij} = \min\left\{ M^*_{ij}, \left\lfloor \frac{M^*_{i\bar{\imath}}}{2} \right\rfloor + \left\lfloor \frac{M^*_{\bar{\jmath}j}}{2} \right\rfloor \right\}$, for all $1 \leq i,j \leq 2N$, where $M^* \in \mathbb{T}^{2N\times 2N}$ is the closure of M.*

The tight closure of an octagonal constraint is needed for existential quantifier elimination, and ultimately, for proving that the class of octagonal relations is closed under composition. The latter is proved in [7]:

Theorem 3. *Let $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$ be a set of variables ranging over $\mathbb{Z}$, $\phi(\mathbf{x})$ a consistent octagonal constraint, and M^t_ϕ be the tight closure of the corresponding DBM. Then the formula $\exists x_k \,.\, \phi(\mathbf{x})$ is equivalent to erasing the $(2k)$-th and $(2k-1)$-th lines and columns from M^t_ϕ. Moreover, the resulting DBM is also coherent and tightly closed.*

Running Example We illustrate the main steps of the transitive closure algorithm for the relation $R(x,y,x',y') (\equiv x+y \leq 5) \wedge (x'-x \leq -2) \wedge (y'-y \leq -3) \wedge (y'-x' \leq 1)$. Figure 11a shows the graph representation $\mathcal{G}_R$. Note that the implicit constraint (crossed in the figure) $x'^+ - x'^- \leq 1$ is not tight. The tightening step replaces the bound 1 with 0. Figure 11b shows the canonic matrix representation $M^*_R = \sigma(R)^*$ of R.



(a) $\mathcal{G}_R$

$$
\begin{array}{c}
 \\ x^+ \\ x^- \\ y^+ \\ y^- \\ x'^+ \\ x'^- \\ y'^+ \\ y'^-
\end{array}
\begin{array}{c}
\begin{array}{cccccccc} x^+ & x^- & y^+ & y^- & x'^+ & x'^- & y'^+ & y'^- \end{array} \\
\begin{pmatrix}
0 & \infty & \infty & 5 & \infty & \infty & \infty & 2 \\
\infty & 0 & \infty & \infty & \infty & -2 & \infty & -1 \\
\infty & 5 & 0 & \infty & \infty & 3 & \infty & 4 \\
\infty & \infty & \infty & 0 & \infty & \infty & \infty & -3 \\
-2 & \infty & \infty & 3 & 0 & \infty & \infty & 0 \\
\infty & \infty & \infty & \infty & \infty & 0 & \infty & 1 \\
-1 & 2 & -3 & 4 & 1 & 0 & 0 & 0 \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & 0
\end{pmatrix}
\end{array}
$$

(b) M^*_R

Fig. 11: Graph and matrix representation of a relation.

Octagonal Relations are Ultimately Periodic Given a consistent octagonal relation $R(\mathbf{x}, \mathbf{x}')$ let $\sigma(R) = M_{\overline{R}}$. Dually, for any coherent DBM $M \in \mathbb{T}^{4N\times 4N}$, let $\rho(M) = \Omega_M$. Clearly, $\rho(\sigma(R)) \Leftrightarrow R$, as required by Def. 2.

In order to prove that the class $\mathcal{R}_{oct}$ of octagonal relations is ultimately periodic, we need to prove that the sequence $\{\sigma(R^m)\}_{m=0}^{\infty}$ is ultimately periodic, for an arbitrary

$$
\begin{array}{c}
\\
x^{+} \\ x^{-} \\ y^{+} \\ y^{-} \\ x''^{+} \\ x''^{-} \\ y''^{+} \\ y''^{-} \\ x'^{+} \\ x'^{-} \\ y'^{+} \\ y'^{-}
\end{array}
\begin{array}{c}
\begin{array}{cccccccccccc}
x^{+} & x^{-} & y^{+} & y^{-} & x''^{+} & x''^{-} & y''^{+} & y''^{-} & x'^{+} & x'^{-} & y'^{+} & y'^{-}
\end{array} \\
\begin{pmatrix}
0 & \infty & \infty & 5 & \infty & \infty & \infty & 2-3k & \infty & \infty & \infty & -1-3k \\
\infty & 0 & \infty & \infty & \infty & -2-2k & \infty & -1-3k & \infty & -4-2k & \infty & -4-3k \\
\infty & 5 & 0 & \infty & \infty & 3-2k & \infty & 4-3k & \infty & 1-2k & \infty & 1-3k \\
\infty & \infty & \infty & 0 & \infty & \infty & \infty & -3-3k & \infty & \infty & \infty & -6-3k \\
-2-2k & \infty & \infty & 3-2k & 0 & \infty & \infty & -5k & \infty & \infty & \infty & -3-5k \\
\infty & \infty & \infty & \infty & \infty & 0 & \infty & 1 & \infty & -2 & \infty & -2 \\
-1-3k & 2-3k & -3-3k & 4-3k & 1 & -5k & 0 & -6k & \infty & -2-5k & \infty & -3-6k \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty & \infty & \infty & -3 \\
-4-2k & \infty & \infty & 1-2k & -2 & \infty & \infty & -2-5k & 0 & \infty & \infty & -5-5k \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty & 1 \\
-4-3k & -1-3k & -6-3k & 1-3k & -2 & -3-5k & -3 & -3-6k & 1 & -5-5k & 0 & -6-6k \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0
\end{pmatrix}
\end{array}
$$

Fig. 12: Left-hand side before tightening

relation $R \in \mathcal{R}_{oct}$. It is sufficient to consider only the case where R is ω-consistent, hence $\sigma(R^m) = M_{\overline{R^m}}$, for all $m \geq 0$. We rely in the following on the main result of [7], which establishes a relation between $M_{\overline{R^m}}$ (the octagonal DBM corresponding to the m-th iteration of R) and $M_{\overline{R}^m}$ (the DBM corresponding to the m-th iteration of $\overline{R} \in \mathcal{R}_{db}$), for $m > 0$:

$$(M_{\overline{R^m}})_{ij} = \min\left\{(M_{\overline{R}^m})_{ij}, \left\lfloor \frac{(M_{\overline{R}^m})_{i\bar{\imath}}}{2} \right\rfloor + \left\lfloor \frac{(M_{\overline{R}^m})_{\bar{\jmath}j}}{2} \right\rfloor\right\}, \text{ for all } 1 \leq i, j \leq 4N \quad (*)$$

This relation is in fact a generalization of the tight closure expression from theorem 2, from $m = 1$ to any $m > 0$.

In Section 4.1 it was shown that difference bounds relations are ultimately periodic. In particular, this means that the sequence $\{M_{\overline{R}^m}\}_{m=0}^{\infty}$, corresponding to the iteration of the difference bounds relation $\overline{R}$, is ultimately periodic. To prove that the sequence $\{M_{\overline{R^m}}\}_{m=0}^{\infty}$ is also ultimately periodic, it is sufficient to show that: the minimum and the sum of two ultimately periodic sequences are ultimately periodic, and also that the integer half of an ultimately periodic sequence is also ultimately periodic.

Lemma 7. *Let $\{s_m\}_{m=0}^{\infty}$ and $\{t_m\}_{m=0}^{\infty}$ be two ultimately periodic sequences. Then the sequences $\{\min(s_m, t_m)\}_{m=0}^{\infty}$, $\{s_m + t_m\}_{m=0}^{\infty}$ and $\left\{\left\lfloor \frac{s_m}{2} \right\rfloor\right\}_{m=0}^{\infty}$ are ultimately periodic as well.*

Proof. For the sequences $\{\min(s_m, t_m)\}_{m=0}^{\infty}$ and $\{s_m + t_m\}_{m=0}^{\infty}$ we assume w.l.o.g. that the two sequences $\{s_m\}_{m=0}^{\infty}$ and $\{t_m\}_{m=0}^{\infty}$ are ultimately periodic starting at the same index K, have the same period c and rates $\lambda_0^{(s)}, ..., \lambda_{c-1}^{(s)}$ respectively $\lambda_0^{(t)}, ..., \lambda_{c-1}^{(t)}$.

We can show that the sum sequence $\{s_m + t_m\}_{m=0}^{\infty}$ is periodic as well starting at K, with period c and rates $\lambda_0^{(s)} + \lambda_0^{(t)}, ..., \lambda_{c-1}^{(s)} + \lambda_{c-1}^{(t)}$. In fact, for every $k \geq K$ and $i = 0, ..., c-1$ we have successively:

$$
\begin{aligned}
(s+t)_{(k+1)c+i} &= s_{(k+1)c+i} + t_{(k+1)c+i} && (4) \\
&= \lambda_i^{(s)} + s_{kc+i} + \lambda_i^{(t)} + t_{kc+i} && (5) \\
&= \lambda_i^{(s)} + \lambda_i^{(t)} + s_{kc+i} + t_{kc+i} && (6) \\
&= (\lambda_i^{(s)} + \lambda_i^{(t)}) + (s+t)_{kc+i} && (7)
\end{aligned}
$$

Similarly, for the min sequence $\{\min(s_m, t_m)\}_{m=0}^{\infty}$ consider the points $(b_i)_{i=0,c-1}$ defined by

$$b_i = \begin{cases} \left\lceil \frac{s_{Kc+i} - t_{Kc+i}}{\lambda_i^{(t)} - \lambda_i^{(s)}} \right\rceil & \text{if } \lambda_i^{(s)} < \lambda_i^{(t)} \text{ and } t_{K+i} < s_{K+i} \\ \left\lceil \frac{t_{Kc+i} - s_{Kc+i}}{\lambda_i^{(s)} - \lambda_i^{(t)}} \right\rceil & \text{if } \lambda_i^{(t)} < \lambda_i^{(s)} \text{ and } s_{K+i} < t_{K+i} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

It can be shown that, for each $i = 0, ..., c-1$ precisely one of the following assertions hold:

1. $(\lambda_i^{(s)} < \lambda_i^{(t)}$ or $\lambda_i^{(s)} = \lambda_i^{(t)}$ and $s_{Kc+i} < t_{Kc+i})$ and $\forall k \geq K + b_i.\ s_{kc+i} \leq t_{kc+i}$
2. $(\lambda_i^{(t)} < \lambda_i^{(s)}$ or $\lambda_i^{(t)} = \lambda_i^{(s)}$ and $t_{Kc+i} \leq s_{Kc+i})$ and $\forall k \geq K + b_i.\ t_{kc+i} \leq s_{kc+i}$

Intuitively, starting from the position $K + b_i$, on every period c, the minimum amongst the two sequences is always defined by the same sequence i.e., the one having the minimal rate on index i, or if the rates are equal, the one having the smaller starting value.

We can show now that the min sequence $\{\min(s_m, t_m)\}_{m=0}^{\infty}$ is periodic starting at $K + \max_{i=0}^{c-1} b_i$, with period c and rates $\min(\lambda_0^{(s)}, \lambda_0^{(t)}), ..., \min(\lambda_{c-1}^{(s)}, \lambda_{c-1}^{(t)})$. That is, we have successively, for every $k \geq K + \max_{i=0}^{c-1} b_i$ and $i = 0, ..., c-1$, and whenever i satisfies the condition (1) above (the case when i satisfies the condition (2) being similar):

$$\begin{aligned} \min(s,t)_{(k+1)c+i} &= \min(s_{(k+1)c+i}, t_{(k+1)c+i}) \\ &= s_{(k+1)c+i} \\ &= \lambda_i^{(s)} + s_{kc+i} \\ &= \min(\lambda_i^{(s)}, \lambda_i^{(t)}) + \min(s_{kc+i}, t_{kc+i}) \\ &= \min(\lambda_i^{(s)}, \lambda_i^{(t)}) + \min(s,t)_{kc+i} \end{aligned}$$

For the sequence $\left\{ \left\lfloor \frac{s_m}{2} \right\rfloor \right\}_{m=0}^{\infty}$, assume that the sequence $\{s_m\}_{m=0}^{\infty}$ is ultimately periodic starting at K, with period c and rates $\lambda_0, ..., \lambda_{c-1}$. It can be easily shown that the sequence $\lfloor \frac{s_m}{2} \rfloor$ is ultimately periodic as well starting at K, with period $2c$, and rates $\lambda_0, ..., \lambda_{c-1}, \lambda_0, ..., \lambda_{c-1}$.

We have successively for any $k \geq K$, and for any $i = 0, ..., c-1$:

$$\left\lfloor \frac{s_{(k+1)2c+i}}{2} \right\rfloor = \left\lfloor \frac{2\lambda_i + s_{k \cdot 2c+i}}{2} \right\rfloor = \lambda_i + \left\lfloor \frac{s_{k \cdot 2c+i}}{2} \right\rfloor$$

Similarly, for any $k \geq K$ and for any $i = 0, ..., c-1$ we have

$$\left\lfloor \frac{s_{(k+1)2c+c+i}}{2} \right\rfloor = \left\lfloor \frac{2\lambda_i + s_{k\cdot 2c+c+i}}{2} \right\rfloor = \lambda_i + \left\lfloor \frac{s_{k\cdot 2c+c+i}}{2} \right\rfloor$$

□

Together with the above relation $(*)$, lemma 7 proves that $\mathcal{R}_{oct}$ is ultimately periodic.

Checking ω-Consistency and Inconsistency of Octagonal Relations This section describes an efficient method of deciding the queries $\mathcal{Q}_1$ and $\mathcal{Q}_2$ (lines 7 and 9 in Fig. 1) for the class of octagonal relations. In order to deal with these queries symbolically,we need to consider first the class $\mathcal{R}_{oct}(k)$ of octagonal relations with parameter k. In the rest of this section, let $k \notin \mathbf{x}$ be a variable ranging over $\mathbb{N}_+$.

Definition 8. *Then a formula $\phi(\mathbf{x}, z)$ is a parametric octagonal constraint if it is equivalent to a finite conjunction of terms of the form $\pm x_i \pm x_j \le a_{ij} \cdot k + b_{ij}$, $2x_i \le c_i \cdot k + d_i$, or $-2x_i \le c'_i \cdot k + d'_i$, where $a_{ij}, b_{ij}, c_i, d_i, c'_i, d'_i \in \mathbb{Z}$ and $1 \le i, j \le N,\ i \ne j$.*

A parametric octagon $\phi(\mathbf{x}, k)$ is represented by a matrix $M_{\overline{\phi}}[k]\mathbb{T}[k]^{2N\times 2N}$ of linear terms over k, and viceversa, a matrix $M[k] \in \mathbb{T}[k]^{2N\times 2N}$ corresponds to a parametric octagon $\Omega_M(k)$. We define $\pi(M[k]) = \Omega_M(k)$. As in the case of difference bounds constraints, one notices that $\pi(k \cdot \Lambda + \sigma(R^b)) \in \mathcal{R}_{oct}(k)$, for $R \in \mathcal{R}_{oct}$, $b \ge 0$ and $\Lambda \in \mathbb{T}^{4N\times 4N}$.

The composition of parametric octagonal relations (from e.g. $\mathcal{Q}_1$) requires the computation of the tight closure in the presence of parameters. According to theorem 2, the parametric tight closure can be expressed as a matrix of elements of the form $\min\{t_i(k)\}_{i=1}^m$, where $t_i(k)$ are either: (i) linear terms, i.e. $t_i(k) = a_i \cdot k + b_i$, or (ii) sums of halved linear terms, i.e. $t_i(k) = \lfloor \frac{a_i \cdot k + b_i}{2} \rfloor + \lfloor \frac{c_i \cdot k + d_i}{2} \rfloor$.

The main idea is to split a halved linear term of the form $\lfloor \frac{a \cdot k + b}{2} \rfloor,\ k > 0$ into two linear terms $a \cdot k + \lfloor \frac{b}{2} \rfloor$ and $a \cdot k + \lfloor \frac{b-a}{2} \rfloor$, corresponding to the cases of $k > 0$ being even or odd, respectively. This is justified by the following equivalence:

$$\{\lfloor \tfrac{a \cdot k + b}{2} \rfloor \mid k > 0\} = \{a \cdot k + \lfloor \tfrac{b}{2} \rfloor \mid k > 0\} \cup \{a \cdot k + \lfloor \tfrac{b-a}{2} \rfloor \mid k > 0\}$$

Hence, an expression of the form $\min\{t_i(k)\}_{i=1}^m$ yields two expressions $\min\{t_i^e(k)\}_{i=1}^m$, for even k, and $\min\{t_i^o(k)\}_{i=1}^m$, for odd k, where t_i^e and t_i^o, $1 \le i \le m$, are effectively computable linear terms. With these considerations, $\mathcal{Q}_1$ (for octagonal relations) is equivalent to a conjunction of equalities of the form $\forall k > 0\ .\ \min\{t_i^\bullet(k)\}_{i=1}^m = t_0^\bullet(k),\ \bullet \in \{e, o\}$. Now we can apply lemma 5 to the right-hand sides of the equivalences above, to give efficient equivalent conditions for deciding $\mathcal{Q}_1$.

The query $\mathcal{Q}_2$ is, according to Theorem 2, equivalent to finding either (i) a strictly negative cycle in a parametric octagonal DBM $M[k]$, or (ii) an index $1 \le i \le 4N$ such that $\lfloor \frac{M[k]_{i\bar{i}}}{2} \rfloor + \lfloor \frac{M[k]_{\bar{i}i}}{2} \rfloor < 0$. Considering that the set of terms corresponding to the two cases above is $T = \{a_i \cdot k + b_i\}_{i=1}^m \cup \{\lfloor \frac{c_i \cdot k + d_i}{2} \rfloor + \lfloor \frac{e_i \cdot k + f_i}{2} \rfloor\}_{i=1}^p$, we split each term $t \in T$ into two matching linear terms, and obtain, equivalently:

$$T_{e,o} = \{\alpha_i^e \cdot k + \beta_i^e\}_{i=1}^{m+p} \cup \{\alpha_i^o \cdot k + \beta_i^o\}_{i=1}^{m+p}$$

Now we can apply lemma 6, and compute the minimal value for which a term $t \in T_{e,o}$ becomes negative, i.e. $n_0 = \min_{i=1}^{m+p} \min(2\gamma_i^e, 2\gamma_i^o - 1)$, where $\gamma_i^\bullet = \max(1, \lfloor -\frac{\beta_i^\bullet}{\alpha_i^\bullet} \rfloor + 1)$, if $\alpha_i^\bullet < 0$, 1 if $\alpha_i^\bullet \geq 0 \ \wedge \ \alpha_i^\bullet + \beta_i^\bullet < 0$, and ∞, otherwise, for $\bullet \in \{e, o\}$.

Running Example We show below how the universal query $\mathcal{Q}_1$ is checked for the octagonal relation $R(x, y, x', y')(\equiv x+y \leq 5) \wedge (x' - x \leq -2) \wedge (y' - y \leq -3) \wedge (y' - x' \leq 1)$. The check at line 6 of the transitive closure algorithm succeeds for $(b, c) = (1, 1)$. Then, in order to check the universal query $\mathcal{Q}_1$ at line 7, one needs to compose parametric difference bound matrices, similarly as in the case of difference bound relations. Moreover, the tightening step must be performed. Figure 12 shows the parametric matrix, which is closed, but not tightly closed. We illustrate the tightening for the constraint $y''^{+} - y'^{-} \leq -6 - 6k$. The term $-6 - 6k$ can be split into even part $-3 - 12k$ and odd part $3 - 12k$. y''^{+} represents y'' and is stored as $2y'' = y''^{+} - y''^{-} \leq -6k$. $-y'^{-}$ represents $-(-y') = y'$ and is stored as $2y' = y'^{+} - y'^{-} \leq -6 - 6k$. Hence, the tightening amounts to computation of $\left\lfloor \frac{-6k}{2} \right\rfloor + \left\lfloor \frac{-6-6k}{2} \right\rfloor$. We split $\left\lfloor \frac{-6k}{2} \right\rfloor$ into $-6k$ (even) and $3 - 6k$ (odd) and similarly, we split $\left\lfloor \frac{-6-6k}{2} \right\rfloor$ into $-3 - 6k$ (even) and $-6k$ (odd). Then we tighten even and odd parts separately:

$$min\{-3 - 12k, (-6k) + (-3 - 6k)\} = -3 - 12k \text{ (even)}$$
$$min\{3 - 12k, (3 - 6k) + (-6k)\} = 3 - 12k \text{ (odd)}$$

The query $\mathcal{Q}_1$ is checked similarly to the case of difference bound relations, the check is just performed twice, both for the even and odd parts. The algorithm returns the transitive closure at line 8:

$$\begin{aligned}
R^* &\equiv \bigvee_{i=0}^{0} R^i \ \vee \ \exists k \geq 0 \,.\, \bigvee_{i=0}^{0} \pi(k \cdot \Lambda + \sigma(R)) \circ R^i \\
&\equiv (x' = x \wedge y' = y) \vee \\
&\quad \exists k \geq 0 \,.\, (y' \leq -3k \wedge x' - x \leq -2k - 2 \wedge y' - x' \leq 1 \wedge y' - x \leq -3k - 1 \wedge \\
&\quad y' - y \leq -3k - 3 \wedge x + y \leq 5 \wedge x + y' \leq -3k + 2 \wedge \\
&\quad x' + y \leq -2k + 3 \wedge x' + y' \leq -5k \wedge y' + y \leq -3k + 4)
\end{aligned}$$

After quantifier elimination, we obtain

$$\begin{aligned}
R^* \equiv\ & (x' = x \wedge y' = y) \vee \\
& (y' \leq 0 x' - x \leq -2 \wedge y' - x \leq -1 \wedge y' - x' \leq 1 \wedge \\
& y' - y \leq -3 \wedge x + y \leq 5 \wedge x + y' \leq 2 \wedge \\
& x' + y \leq 3 \wedge x' + y' \leq 0 \wedge y + y' \leq 4)
\end{aligned}$$

4.3 Finite Monoid Affine Transformations

The class of affine transformations is one of the most general classes of deterministic transition relations involving integer variables. If $\mathbf{x} = \langle x_1, \ldots, x_N \rangle$ is a vector of variables ranging over $\mathbb{Z}$, an affine transformation is a relation of the form:

$$T \ \equiv \ \mathbf{x}' = A \otimes \mathbf{x} + \mathbf{b} \ \wedge \ \phi(\mathbf{x}) \tag{9}$$

where $A \in \mathbb{Z}^{N \times N}$, $\mathbf{b} \in \mathbb{Z}^N$, ϕ is a Presburger formula, and $\otimes$ stands for the standard matrix multiplication in $\mathbb{Z}$.

The affine transformation is said to have the finite monoid property [5, 20] if the monoid $\langle \mathcal{M}_A, \otimes \rangle$, where $\mathcal{M}_A = \{A^{\otimes^i} \mid i \geq 0\}$ is finite. In this case, we also say that A is finite monoid. Here $A^{\otimes^0} = I_N$ and $A^{\otimes^i} = A \otimes A^{\otimes^{i-1}}$, for $i > 0$. Intuitively, the finite monoid property is equivalent to the fact that A has finitely many powers (for the standard integer multiplication) that repeat periodically. It is easy to see that A is finite monoid if and only if there exists $p \geq 0$ and $l > 0$ such that $A^{\otimes^p} = A^{\otimes^{p+l}}$, i.e. $\mathcal{M}_A = \{A^{\otimes^0}, \ldots, A^{\otimes^p}, \ldots, A^{\otimes^{p+l-1}}\}$.

If A is finite monoid, it can be shown that T^* can be defined in Presburger arithmetic [5, 20]. We achieve the same result below, by showing that finite monoid affine transformations are ultimately periodic relations. As a byproduct, the transitive closure of such relations can also be computed by the algorithm in Fig. 1.

An affine transformation T (9) can be equivalently written in the homogeneous form:

$$T \equiv \mathbf{x}'_h = A_h \otimes \mathbf{x}_h \ \wedge \ \phi_h(\mathbf{x}_h) \text{ where } A_h \equiv \left(\begin{array}{c|c} A & \mathbf{b} \\ \hline 0 \ldots 0 & 1 \end{array} \right)$$

where $x_h = \langle x_1, \ldots x_N, x_{N+1} \rangle$ with $x_{N+1} \notin \mathbf{x}$ being a fresh variable and $\phi_h(\mathbf{x}_h) \equiv \phi(\mathbf{x}) \wedge x_{N+1} = 1$. In general, the k-th iteration of an affine transformation can be expressed as:

$$T^k \ \equiv \ \mathbf{x}'_h = A_h{}^{\otimes^k} \otimes \mathbf{x}_h \ \wedge \ \forall 0 \leq \ell < k \, . \, \phi_h(A_h{}^{\otimes^\ell} \otimes \mathbf{x}_h) \tag{10}$$

Notice that, if $\mathbf{x}_h^{(0)}$ denotes the initial values of $\mathbf{x}_h$, the values of $\mathbf{x}_h$ at the ℓ-th iteration are $\mathbf{x}_h^{(\ell)} = A_h{}^{\otimes^\ell} \otimes \mathbf{x}_h^{(0)}$. Moreover, we need to ensure that all guards up to (and including) the $(k-1)$-th step are satisfied, i.e. $\phi_h(A_h{}^{\otimes^\ell} \otimes \mathbf{x}_h)$, for all $0 \leq \ell < k$.

For the rest of the section we fix A and $\mathbf{b}$, as in (9). The encoding of a consistent affine transformation T is defined as $\sigma(T) = A_h \in \mathbb{T}^{(N+1) \times (N+1)}$. Dually, for some $M \in \mathbb{T}[k]^{(N+1) \times (N+1)}$, we define:

$$\pi(M) : \exists x_{N+1}, x'_{N+1} \, . \, \mathbf{x}'_h = M \otimes \mathbf{x}_h \ \wedge \ \forall 0 \leq \ell < k \, . \, \phi_h(M[\ell/k] \otimes \mathbf{x}_h)$$

where $M[\ell/k]$ denotes the matrix M in which each occurrence of k is replaced by ℓ. In contrast with the previous cases (Section 4.1 and Section 4.2), only M is not sufficient here to recover the relation $\pi(M)$ – ϕ needs to be remembered as well[6].

With these definitions, we have $\sigma(T^k) = A_h^{\otimes^k}$, for all $k > 0$ – as an immediate consequence of (10). The next lemma proves that the class of finite monoid affine relations is ultimately periodic.

Lemma 8. *Given a finite monoid matrix $A \in \mathbb{Z}^{N \times N}$ and a vector $\mathbf{b} \in \mathbb{Z}^N$, the sequence $\{A_h^{\otimes^k}\}_{k=0}^{\infty}$ is ultimately periodic.*

[6] This incurs a slight modification of the algorithm presented in Fig. 1.

Proof. Let $A \in \mathbb{Z}^{N \times N}$ be a matrix, $\mathbf{b} \in \mathbb{Z}^N$ be a vector, and

$$A_h \equiv \left(\begin{array}{c|c} A & \mathbf{b} \\ \hline 0 \ldots 0 & 1 \end{array} \right)$$

Then we have, for all $k \geq 0$:

$$(A_h)^{\otimes k} = \left(\begin{array}{c|c} A^{\otimes k} & \sum_{i=0}^{k-1} A^{\otimes i} \otimes \mathbf{b} \\ \hline 0 \ldots 0 & 1 \end{array} \right)$$

For $i = N+1,\ 1 \leq j \leq N+1$, $\{(A_h^{\otimes k})_{ij}\}_{k=0}^{\infty}$ is trivially ultimately periodic. For $1 \leq i, j \leq N$, $\{(A_h^{\otimes k})_{ij}\}_{k=0}^{\infty}$ is ultimately periodic due to the fact that A is finite monoid. It remains to be proven that, for all $1 \leq j \leq N$, the sequence $\{(\sum_{i=0}^{k-1} A^{\otimes i} \otimes \mathbf{b})_j\}_{k=0}^{\infty}$ is ultimately periodic. W.l.o.g. assume that the monoid of A is $\mathcal{M}_A = \{M^{\otimes 0}, M^{\otimes 1}, \ldots, M^{\otimes p}, \ldots M^{\otimes p+l-1}\}$, where $M^{\otimes p} = M^{\otimes p+l}$. Then, for $k \geq p$ we have:

$$\sum_{i=0}^{k-1} M^{\otimes i} = \sum_{i=0}^{p-1} M^{\otimes i} + \lfloor \tfrac{k-p+1}{l} \rfloor \cdot \sum_{i=p}^{p+l-1} M^{\otimes i} + \sum_{i=p}^{p+((k-p+1) \mod l)} M^{\otimes i}$$

Hence the sequence $\{\sum_{i=0}^{k-1} M^{\otimes i}\}_{k=0}^{\infty}$ is ultimately periodic with period l and rates

$$\Lambda_j = \sum_{i=p}^{p+l-1} M^{\otimes i}$$

for all $j = 0, 1, \ldots, l-1$.

□

The queries $\mathcal{Q}_1$ and $\mathcal{Q}_2$ (lines 7 and 9 in Fig. 1) in the case of finite monoid affine transformations, are expressible in Presburger arithmetic. These problems could be simplified in the case of affine transformations without guards, i.e $T \equiv \mathbf{x}' = A\mathbf{x} + \mathbf{b}$. The transformation is, in this case, ω-consistent. Consequently, $\mathcal{Q}_1$ reduces to an equivalence between two homogeneous systems $\mathbf{x}'_h = A_{1h} \otimes \mathbf{x}_h$ and $\mathbf{x}'_h = A_{2h} \otimes \mathbf{x}_h$. This is true if and only if $A_{1h} = A_{2h}$. The query $\mathcal{Q}_2$ becomes trivially false in this case.

5 Reachability Analysis of Counter Automata

This section describes how the transitive closure algorithm from the previous is used to tackle the control state reachability problem (or, equivalently, the emptiness problem) of counter automata. The algorithms described in this section are implemented in a tool called FLATA [22], which was successfully used to verify several hardware and software systems (see Section 6).

In this section, $\mathcal{R}(\mathbf{x}, \mathbf{x}')$ denotes the class of arithmetic relations, which are conjunctions of atomic formulae of these forms:

- $\sum_{i=1}^{n} a_i x_i \leq a_0$, where $a_i \in \mathbb{Z}$ for all $1 \leq i \leq n$ (linear constraint)
- $\sum_{i=1}^{n} a_i x_i \equiv_k a_0$, where $a_i \in \mathbb{Z}$, $k \in \mathbb{Z}^+$ for all $1 \leq i \leq n$ (modulo constraint)

We denote by $\mathcal{R}_{db}(\mathbf{x}, \mathbf{x}')$ the class of difference bound relations, by $\mathcal{R}_{oct}(\mathbf{x}, \mathbf{x}')$ the class of octagons, by $\mathcal{R}_{fm}(\mathbf{x}, \mathbf{x}')$ the class of finite monoid affine relations, and by

$\mathcal{R}_{poly}(\mathbf{x}, \mathbf{x}')$ the class of polyhedra (a polyhedron is a conjunction of linear constraints defined above). All these classes are subclasses of $\mathcal{R}(\mathbf{x}, \mathbf{x}')$ and moreover

$$\mathcal{R}_{db}(\mathbf{x}, \mathbf{x}') \subsetneq \mathcal{R}_{oct}(\mathbf{x}, \mathbf{x}') \subsetneq \mathcal{R}_{poly}(\mathbf{x}, \mathbf{x}') \subsetneq \mathcal{R}(\mathbf{x}, \mathbf{x}')$$

and

$$\mathcal{R}_{fm}(\mathbf{x}, \mathbf{x}') \subsetneq \mathcal{R}_{poly}(\mathbf{x}, \mathbf{x}')$$

5.1 Counter Automata

A counter automaton (CA) is a tuple $A = \langle \mathbf{x}, Q, I, F, T \rangle$, where $\mathbf{x} = \{x_1, \ldots, x_n\}$ is a finite, ordered set of counters ranging over $\mathbb{Z}$, Q is a finite set of control states, $I \subseteq Q$ is a set of initial states, $F \subseteq Q$ is a set of final states, and T is a transition relation given by a set of rules $q \xrightarrow{R} q'$, where $R \in \mathcal{R}(\mathbf{x}, \mathbf{x}')$ is an arithmetic relation relating current values of counters $\mathbf{x}$ to their future values $\mathbf{x}' = \{x' \mid x \in \mathbf{x}\}$. Let $\delta_S, \delta_T : T \to Q$ and $\delta_R : T \to \mathcal{R}(\mathbf{x}, \mathbf{x}')$ be mappings indicating source control state, target control state, and relational labeling of a transition, respectively. Given a transition $t : q \xrightarrow{R} q'$, let $\delta_S(t) = q$, $\delta_T(t) = q'$, and $\delta_R(t) = R$. Furthermore, let $In, Out, Incid, SLoops : Q \to 2^T$ be mappings indicating incoming, outgoing, incident transitions, and self-loops of a control state: $In(q) = \{t \in T \mid \delta_T(t) = q\}$, $Out(q) = \{t \in T \mid \delta_S(t) = q\}$, $Incid(q) = In(q) \cup Out(q)$, $SLoops(q) = In(q) \cap Out(q)$.

A control path in a counter automaton A is a finite sequence $q_1 q_2 \ldots q_n$ of control states such that, for all $1 \leq i < n$, there exists a transition rule $q_i \xrightarrow{\varphi_i} q_{i+1}$. A cycle is a control path starting and ending in the same control state. An elementary cycle is a cycle in which each state appears only once, except for the first one, which appears both at the beginning and at the end. A counter automaton is said to be flat iff each control state belongs to at most one elementary cycle.

A configuration of a CA A is a pair $\langle q, \nu \rangle$ where $q \in Q$ is a control state, and $\nu \in \mathbb{Z}^n$ is a valuation of counters. A configuration $\langle q', \nu' \rangle$ is an immediate successor of $\langle q, \nu \rangle$ if and only if A has a transition rule $q \xrightarrow{R} q'$ such that $\models R[\nu/\mathbf{x}, \nu'/\mathbf{x}']$. Given two control states $q, q' \in Q$, a run of A from q to q' is a finite sequence of configurations $\rho = c_1 c_2 \ldots c_m$, where $c_1 = \langle q, \nu \rangle$, $c_m = \langle q', \nu' \rangle$ for some $\nu, \nu' \in \mathbb{Z}^n$, and c_{i+1} is an immediate successor of c_i, for all $1 \leq i < m$. Now we define the run, trace and input-output semantics of a CA A.

$$Run(A) = \{\rho \mid \rho = \langle q_1, \nu_1 \rangle \ldots \langle q_m, \nu_m \rangle \text{is a run}, q_1 \in I, q_m \in F\}$$

$$Tr(A) = \{\nu_1 \ldots \nu_m \mid \langle q_1, \nu_1 \rangle \ldots \langle q_m, \nu_m \rangle \in Run(A)\}$$

$$IO(A) = \{\langle \nu_1, \nu_m \rangle \mid \nu_1 \ldots \nu_m \in Tr(A)\}$$

With the above definitions, the emptiness problem for a given CA A, $\textsc{Empty}(A)$ for short, can be formulated in any of the following equivalent ways:

$$Run(A) \stackrel{?}{=} \emptyset \qquad Tr(A) \stackrel{?}{=} \emptyset \qquad IO(A) \stackrel{?}{=} \emptyset$$

The reachability problem for a given a CA $A = \langle \mathbf{x}, Q, I, F, T\rangle$ and $q \in Q$, $\text{REACH}(A,q)$ for short, can be formulated as

is there a run from some $q_i \in I$ to q ?

Note that the reachability and the emptiness problems are equivalent, since one can be formulated by the other.

$$\text{REACH}(A{=}\langle \mathbf{x}, Q, I, F, T\rangle, q{\in}Q) \textbf{ iff } \neg\text{EMPTY}(A'{=}\langle \mathbf{x}, Q, I, \{q\}, T\rangle)$$

$$\text{EMPTY}(A{=}\langle \mathbf{x}, Q, I, F, T\rangle) \textbf{ iff } \bigwedge_{q\in F} \neg\text{REACH}(A,q)$$

The rest of this section describes a procedure for deciding the emptiness of a counter automaton, in general. The idea of the procedure is to eliminate the control states of the automaton, while preserving the input-output semantics. The two main ingredients of this method are:

– a semi-algorithm for the computation of transitive closures of disjunctive relations
– a state elimination procedure, based on the composition of incoming with outgoing transitions

An important class of counter automata, for which the emptiness decision method presented in this section is guaranteed to terminate, are the flat counter automata with ultimately periodic relations on loops, i.e. each transition belonging to an elementary loop is labeled by an ultimately periodic relation. For more information on flat counter automata, the interested reader is pointed to e.g. [8, **?**].

5.2 Computing Transitive Closures of Disjunctive Relations

The transitive closure algorithms introduced in Sections 3 work typically on conjunctive relations – notice that the difference bounds and octagonal relation as well as the finite monoid affine transformations are all defined as conjunctions of atomic propositions. In practice we are often confronted with disjunctive relations, corresponding to the case of nested loops. It is however well known that disjunctive transition relations are Turing-complete, as one can encode the semantics of a 2-counter machine [30] into a loop labeled by a disjunction of difference bounds relations.

In this section we present a semi-algorithm that attempts to define the transitive closure of disjunctive relations as a formula of Presburger arithmetic, whenever this is possible. More precisely, we aim at computing $(\bigvee_{L\in\mathcal{L}} L)^*$, where $\mathcal{L}$ is a finite set of relations, i.e. the effects of all "interleavings" of relations in $\mathcal{L}$ – we refer to an interleaving as to a sequence of relation symbols, representing the composition of the relations along the way. One could attempt to compute the transitive closure by successively enumerating all interleavings of length $0, 1, 2, \ldots, MAX$ with the hope that at some point $0 \leq i < MAX$, we find out that the interleavings of length $i{+}1$ are subsumed by those of length $1, \ldots, i$. When MAX is reached, the semi-algorithm gives up.

A naive enumeration of interleavings would clearly not terminate and return the precise answer even in the case where $\mathcal{L}$ contains a single increment relation ($x' = x{+}1$).

To tackle this problem, we first compute the transitive closures of the ultimately periodic relations in $\mathcal{L}$, and explore the interleavings between these transitive closures instead. Clearly, for any relation R, we have $R^+ \circ R^+ \rightarrow R^+$, hence we only explore the interleavings in which only successive occurrences of different transitive closures are considered.

Algorithm 1 Semi-algorithm computing $(\bigvee_{L\in\mathcal{L}} L)^*$. Function arguments: $\mathcal{L} \subseteq \mathcal{R}_{oct}(\mathbf{x},\mathbf{x}') \cup \mathcal{R}_{fm}(\mathbf{x},\mathbf{x}')$, $\mathcal{R},\mathcal{R}',\mathcal{R}_p \subseteq \mathcal{R}(\mathbf{x},\mathbf{x}')$, $R \in \mathcal{R}(\mathbf{x},\mathbf{x}')$.

procedure DISJCLOSURE($\mathcal{L}$, $\mathcal{R}_p$)
  INIT
  $\mathcal{R} \leftarrow \mathcal{R}_1$
  ADDCHECK($\mathcal{R}$, $\mathcal{R}_0$)
  $\mathcal{R}' \leftarrow \emptyset$, $i \leftarrow 1$
  while $i < MAX$ **do**
    $\mathcal{R}' \leftarrow \mathcal{R}$
    for all $R_1 \in \mathcal{R}_i$ **do**
      for all $R_2 \in \mathcal{R}_1$ **do**
        $R \leftarrow R_1 \circ R_2$
        if $(\bigvee_{R_p\in\mathcal{R}_p} R_p \circ R) \not\Leftrightarrow \bot$ **then**
          ADDCHECK($\mathcal{R}_{i+1}$, R)
    ADDCHECK($\mathcal{R}$, $\mathcal{R}_{i+1}$)
    if $\mathcal{R} = \mathcal{R}'$ **then**
      return $(\top, \mathcal{R})$
    $i \leftarrow i+1$
  return $(\bot)$

procedure INIT
  for all $1 \leq i \leq MAX$ **do**
    $\mathcal{R}_i \leftarrow \emptyset$
  $\mathcal{R}_0 \leftarrow \{Id\}$
  for all $L \in \mathcal{L}$ **do**
    ADDCHECK($\mathcal{R}_1$, L^+)

procedure ADDCHECK($\mathcal{R}, R$))
  if $R \Rightarrow (\bigvee_{R'\in\mathcal{R}} R')$ **then**
    return
  for all $R' \in \mathcal{R}$ **do**
    if $R' \Rightarrow R$ **then**
      remove R' **from** $\mathcal{R}$
  add R **to** $\mathcal{R}$

procedure ADDCHECK($\mathcal{R}, \mathcal{R}'$))
  for all $R \in \mathcal{R}'$ **do**
    ADDCHECK($\mathcal{R}, R$)

The semi-algorithm given in 1 computes the transitive closure of a relation in DNF which is passed by the parameter $\mathcal{L}$ as a set of its disjuncts. It uses sets $\mathcal{R}_i$, $0 \leq i \leq MAX$ to store computed interleavings of length i. $\mathcal{R}$ cumulates all interleavings of already computed lengths. Procedure ADDCHECK($\mathcal{R}$,R) adds the relation R to the set $\mathcal{R}$ only if the computed interleavings $\mathcal{R}$ subsume it and removes all relations in $\mathcal{R}$ which are subsumed by R. INIT clears all the sets of relations and fills the sets corresponding to interleavings of length 0 and 1. The main procedure DISJCLOSURE uses auxiliary set $\mathcal{R}'$ to check if the fixpoint has been reached. Interleavings of length $i+1$ are computed by composing relations in $\mathcal{R}_i$ with relations in $\mathcal{R}_1$. The second argument of DISJCLOSURE $\mathcal{R}_p$ is the prefix-context of the disjunctive relation, which is used to prune interleavings—whenever the composition of the context with an interleaving is equivalent to $\bot$, the interleaving is not considered. Hence, a computation without pruning can be achieved by giving the context equivalent to $\top$, i.e. DISJCLOSURE($\mathcal{L}$, $\{\top\}$).

5.3 Emptiness Procedure

We attempt to answer emptiness queries by application of three basic transformations preserving the input-output semantics of a CA, which are shown in the right column of the Algorithm 2. $\textsc{Transf}_0$ transforms the automaton into the form $A = \langle Q, \{q_i\}, \{q_f\}, T\rangle$, where $In(q_i) = \emptyset$ and $Out(q_f) = \emptyset$. $\textsc{Transf}_1$ eliminates a control state with no self-loops. $\textsc{Transf}_2$ replaces self-loops of a control state with meta-transitions (passed in the argument $\mathcal{C}$) which express precise effects of iterating these self-loops any number of times in any interleaving. The obvious condition for application of $\textsc{Transf}_2$ is that these meta-transitions are indeed disjuncts of this transitive closure.

The left column of the Algorithm 2 describes a strategy for answering the emptiness query. The main procedure $\textsc{Empty}$ first transforms the automaton by $\textsc{Transf}_0$. The purpose is that the emptiness can be then answered by attempting to eliminate all states but q_i, q_f by the other two transformations $\textsc{Transf}_1$ and $\textsc{Transf}_2$. Then, by calling $\textsc{Transformation}_1$, it eliminates all the states without loops, in an order where states generating less transitions are processed first. Finally, by calling $\textsc{Transformation}_2$, it attempts to eliminate the states with loops. If the transitive closure of the disjunctive relation corresponding to loops at a state q is computable, it is processed by $\textsc{Transf}_2$. Subsequently, states q and q', now without loops, are eliminated by $\textsc{Transf}_1$.

We now describe the reason behind the while loop in $\textsc{Transformation}_2$. Suppose that $t_1 : q_1 \xrightarrow{R_1} q_2$, $t_2 : q_2 \xrightarrow{R_2} q_3$. Then, $\exists \mathbf{x} \,.\, R_1 \circ R_2 \Rightarrow \exists \mathbf{x} \,.\, R_2$. This means that as a result of relational composition, the constraint on $\mathbf{x}'$ may be stronger. In other words, elimination of q_2 increases the chance for success of $\textsc{DisjClosure}$, since pruning via the prefix may be more effective after elimination of q_2. Hence, $\textsc{Transformation}_2$ exits only when no elimination was possible during the last while-loop iteration. Depending on the success of $\textsc{Transformation}_2$, the main procedure $\textsc{Empty}$ returns ? for don't know, $\top$ for empty, and $\bot$ for non-empty.

We following lemmas lead to a theorem which proves that the emptiness procedure preserves the input-output semantics.

Lemma 9. *Let A, A' be arbitrary counter automata. Then,*

$$P_{AA'} \Leftrightarrow IO(A) \subseteq IO(A')$$

$$P_{A'A} \Leftrightarrow IO(A') \subseteq IO(A),$$

where

$$P_{AA'} \equiv \forall \langle q_1, \nu_1\rangle \ldots \langle q_m, \nu_m\rangle \in Tr(A) \,.\, \exists \langle q'_1, \nu'_1\rangle \ldots \langle q'_{m'}, \nu'_{m'}\rangle \in Tr(A') \,.\, \nu_1 = \nu'_1 \wedge \nu_m = \nu'_{m'}$$

$$P_{A'A} \equiv \forall \langle q_1, \nu_1\rangle \ldots \langle q_m, \nu_m\rangle \in Tr(A') \,.\, \exists \langle q'_1, \nu'_1\rangle \ldots \langle q'_{m'}, \nu'_{m'}\rangle \in Tr(A) \,.\, \nu_1 = \nu'_1 \wedge \nu_m = \nu'_{m'}$$

Proof. Follows immediately from the definition of the input-output semantics.

By Lemma 9, proving $IO(A) = IO(A')$ amounts to proving $P_{AA'}$ and $P_{A'A}$.

Lemma 10. *Let $A' = \textsc{Transf}_0(A)$. Then, $IO(A) = IO(A')$.*

Proof. ($\subseteq$) Let $(\rho = c_1 \ldots c_m) \in Tr(A)$ and let $c_1 = \langle q_1, \nu_1\rangle$, $c_m = \langle q_m, \nu_m\rangle$. By construction, A' has a trace $\rho' = c_0 c_1 \ldots c_m c_{m+1}$, where $c_0 = \langle q_i, \nu_1\rangle$, $c_{m+1} = \langle q_f, \nu_m\rangle$.

($\supseteq$) Similar to ($\subseteq$)

Lemma 11. *Let $A' = \textsc{Transf}_1(A, q)$. Then, $IO(A) = IO(A')$.*

Algorithm 2 Emptiness Algorithm

procedure EMPTY(A)
  $A \leftarrow$ TRANSF$_0(A)$
  $A \leftarrow$ TRANSFORMATION_1(A)
  $A \leftarrow$ TRANSFORMATION_2(A)
  if $Q \neq F \cup I$ **then**
    return $(A, ?)$
  if $T = \emptyset$ **then**
    return $(A, \top)$
  else
    return $(A, \bot)$

procedure TRANSFORMATION_1(A)
  /* $A = \langle Q, I, F, T\rangle$ */
  $X \leftarrow \{q \mid SLoops(q) = \emptyset\} \setminus (I \cup F)$
  while $X \neq \emptyset$ **do**
    $q \leftarrow \arg\min_{q \in X} |In(q)| * |Out(q)|$
    $A \leftarrow$ TRANSF$_1(A, q)$
    $X \leftarrow \{q \in Q \mid SLoops(q) = \emptyset\}$
  return A

procedure TRANSFORMATION_2(A)
  /* $A = \langle Q, I, F, T\rangle$ */
  $s \leftarrow |Q|$, $s' \leftarrow 0$
  while $s \neq s'$ **do**
    $s' \leftarrow s$
    for all $q \in Q \setminus (I \cup F)$ **do**
      $Y \leftarrow$ TRANSF$_2(A, q)$
      if $Y = (\top, A')$ **then**
        $A \leftarrow$ TRANSF$_1(A', q)$
        $A \leftarrow$ TRANSF$_1(A, q')$
    $s \leftarrow |Q|$
  return A

procedure TRANSF$_0(A)$
  /* $A = \langle Q, I, F, T\rangle$ */
  $Q' \leftarrow Q \cup \{q_i, q_f\}$, where $q_i, q_f \notin Q$
  $T'_i \leftarrow \{q_i \xrightarrow{Id(\mathbf{x})} q \mid q \in I\}$
  $T'_f \leftarrow \{q \xrightarrow{Id(\mathbf{x})} q_f \mid q \in F\}$
  $T' \leftarrow T \cup T'_i \cup T'_f$
  $A' \leftarrow \langle Q', \{q_i\}, \{q_f\}, T'\rangle$
  return A'

precondition $A = \langle Q, I, F, T\rangle$,
$q \in Q \setminus (I \cup F)$, $SLoops(q) = \emptyset$
procedure TRANSF$_1(A, q)$
  $Q' \leftarrow Q \setminus \{q\}$
  $T_n \leftarrow \{q_1 \xrightarrow{R_1 \circ R_2} q_2 \mid (q_1 \xrightarrow{R_1} q) \in T,$
    $(q \xrightarrow{R_2} q_2) \in T\}$
  $T' \leftarrow (T \setminus Incid(q)) \cup T_n$
  return $\langle Q', I, F, T'\rangle$

precondition $A = \langle Q, I, F, T\rangle$,
$q \in Q \setminus (I \cup F)$, $|SLoops(q)| \geq 1$
procedure TRANSF$_2(A, q)$
  $\mathcal{L} \leftarrow \{L \mid (q \xrightarrow{L} q) \in SLoops(q)\}$
  $\mathcal{R}_p \leftarrow \{L \mid (q' \xrightarrow{L} q) \in In(q), q' \neq q\}$
  $Y \leftarrow$ DISJCLOSURE($\mathcal{L}, \mathcal{R}_p$)
  if $Y = (\top, \mathcal{C})$ **then**
    $Q' \leftarrow Q \cup \{q'\}$
    $T_n \leftarrow \{q \xrightarrow{C} q' \mid C \in \mathcal{C}\}$
    $T_c \leftarrow \{q' \xrightarrow{R} q'' \mid (q \xrightarrow{R} q'') \in T, q'' \neq q\}$
    $T' \leftarrow (T \setminus Out(q)) \cup T_c \cup T_n$
    return $(\top, \langle Q', I, F, T'\rangle)$
  else
    return $(\bot)$

Proof. *($\subseteq$) Let $\rho \in Tr(A)$. Consider its segments of the form $\langle q_1, \nu_1\rangle\langle q_2, \nu_2\rangle\langle q_3, \nu_3\rangle$ s.t. $q_2 = q$. Then, there are transitions $t_1, t_2 \in T$, $t_1 : q_1 \xrightarrow{R_1} q_2$, $t_2 : q_2 \xrightarrow{R_2} q_3$, s.t. $\models R_1[\nu_1/\mathbf{x}, \nu_2/\mathbf{x}']$, and $\models R_2[\nu_2/\mathbf{x}, \nu_3/\mathbf{x}']$. By construction of A', there is a transition $t\colon q_1 \xrightarrow{R_1 \circ R_2} q_2 \in T'$. By definition of the $\circ$ operator, $\models (R_1 \circ R_2)[\nu_1/\mathbf{x}, \nu_3/\mathbf{x}']$. Hence, by rewriting each such segment of ρ with $\langle q_1, \nu_1\rangle\langle q_3, \nu_3\rangle$, we obtain $\rho' \in Tr(A')$. Let $\rho = c_1 \dots c_m$, and $\rho' = c'_1 \dots c'_{m'}$. Clearly, $c_1 = c'_1 \wedge c_m = c'_{m'}$.*

($\supseteq$) Similar to ($\subseteq$)

Lemma 12. *Let $(\top, A') = \text{T}\small{\text{RANSF}}_2(A, q)$. Then, $IO(A) = IO(A')$.*

Proof. *($\subseteq$) Let $\rho \in Tr(A)$. Consider its segments of the form $\langle q_1, \nu_1\rangle\langle q_2, \nu_2\rangle\langle q_3, \nu_3\rangle$ s.t. $q_2 = q$, and $q_1, q_3 \neq q$ (segments of type 1). By construction of A', there is a transition $t\colon q \xrightarrow{R} q' \in T'$ s.t. $Id(\mathbf{x}) \Rightarrow R$.*

Further, consider segments of the form $\langle q_1, \nu_1\rangle \dots \langle q_m, \nu_m\rangle$ s.t. $m \geq 4$, $q_1, q_m \neq q$, and $q_i = q$ for all $2 \leq i \leq m-1$ (segments of type 2). Then, there are transitions in A: $t_i\colon q_i \xrightarrow{R_i} q_{i+1}$ s.t. $\models R_i[\nu_i/\mathbf{x}, \nu_{i+1}/\mathbf{x}']$ for all $1 \leq i \leq m-1$. Firstly, we show that the algorithm DISJCLOSURE *is correct. Clearly, the algorithm* DISJCLOSURE *correctly computes the transitive closure if $\mathcal{R}_p = \{\top\}$. For $\mathcal{R}_p$ as defined in* TRANSF_2*, we show that pruning doesn't cause any loss in the input-output semantics: Suppose that the interleaving $R_2 \circ \dots \circ R_{m-2}$ has been pruned, i.e. that $R_p \circ (R_2 \circ \dots \circ R_{m-2}) \Leftrightarrow \perp$ for all $R_p \in \mathcal{R}_p$. Since $R_1 \in \mathcal{R}_p$, the segment is infeasible—contradiction. Hence, $R_2 \circ \dots \circ R_{m-2}$ hasn't been pruned. Secondly, we show that there is a corresponding segment in A'. By construction of A' and by correctness of* DISJCLOSURE*, there is $t\colon q \xrightarrow{R} q'$ s.t. $R_2 \circ \dots \circ R_{m-2} \Rightarrow R$, and hence $\models R[\nu_2/\mathbf{x}, \nu_{m-1}/\mathbf{x}']$.*

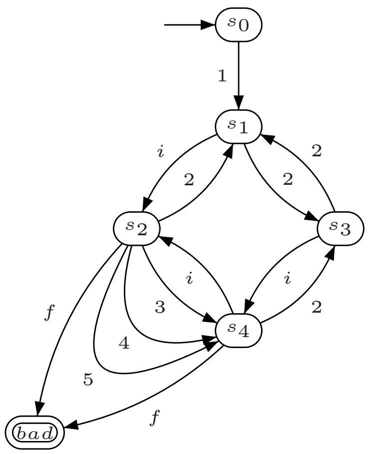
By rewriting each segment of ρ of type 1 with $\langle q_1, \nu_1\rangle\langle q, \nu_2\rangle\langle q', \nu_2\rangle\langle q_3, \nu_3\rangle$ and each segment of type 2 with $\langle q_1, \nu_1\rangle\langle q, \nu_2\rangle\langle q', \nu_{m-1}\rangle\langle q_m, \nu_m\rangle$, we obtain $\rho' \in Tr(A')$. Let $\rho = c_1 \dots c_m$, and $\rho' = c'_1 \dots c'_{m'}$. Clearly, $c_1 = c'_1 \wedge c_m = c'_{m'}$.

($\supseteq$) Similar to ($\subseteq$)

Theorem 4. *Let $(A', _) = $ EMPTY(A). Then, $IO(A) = IO(A')$.*

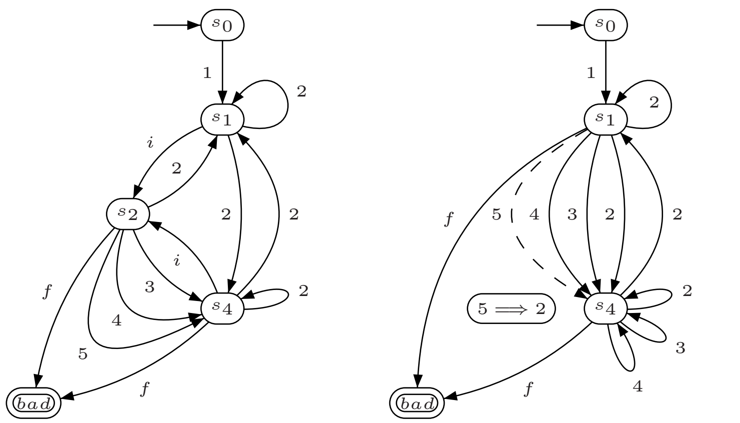
Proof. *A direct consequence of Lemmas 10, 11, and 12.*

Example Figure 13 (a) gives an example of a counter automaton which can be reduced by our method. Since the initial state s_0 has no incoming edges, and the final state bad has no outgoing edges, the procedure TRANSF_0 is not needed. The result of procedure TRANSF_1 is the elimination of states s_3 and s_2 (in this order), the result being depicted in Figure 13 (b). Notice that the relation 5 is subsumed by 2, and hence the corresponding transition is not added to the resulting automaton. The next step is the elimination of state s_1, which occurs on a self-loop. To this end, the procedure TRANSF_2 is applied, the resulting automaton being showed in Figure 13 (c). The last state to be eliminated is s_4, which requires the computation of the transitive closure of a disjunctive relation. The resulting automaton is empty, as all the relations obtained by composition with the incoming and outgoing transitions of state s_4 are inconsistent.

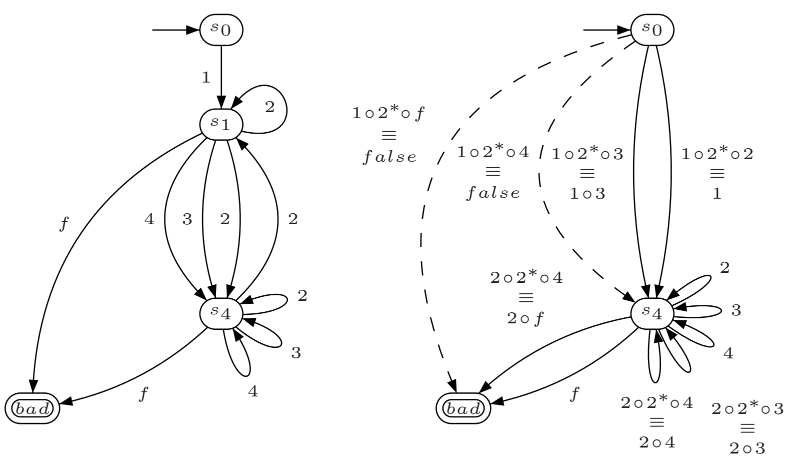


$$
\begin{array}{lll}
i: & & \{a' = a, \quad m' = m\} \\
1: & & \{a' = 0, \quad m' \geq 2\} \\
2: & & \{a' = 0, \quad m' = m\} \\
3: & [m \geq a+2] & \{a' = a+1, m' = m\} \\
4: & [m \leq a] & \{a' = a+1, m' = m\} \\
5: & [m = a+1] & \{a' = 0, \quad m' = m\} \\
f: & [a = m] & \{a' = a, \quad m' = m\}
\end{array}
$$

(a) Counter Automaton Example



(b) After Elimination of States without Self Loops



(c) After Elimination with a State on a Self-loop

Fig. 13: Counter Automaton Reduction Example

6 Experimental Results

In this section, we give the experimental evidence first on the transitive closure algorithm and second on its application—the reachability analysis. We have implemented the techniques described in the previous sections, the sole exception being the transitive closure for the class of finite-monoid affine transformations, in our tool FLATA [22], a framework we develop for the analysis of counter systems. We use subclass-specific algorithms for operations such as composition, satisfiability, and transitive closure. We resort to an external SMT solver YICES [18] only for checking satisfiability of polyhedra and modulo relations.

6.1 Transitive Closure Algorithm

We have implemented the transitive closure algorithm from Fig. 1 within the FLATA toolset [22]. We compared the performance of this algorithm with our older transitive closure computation methods for difference bounds [8] and octagonal relations [7]. We currently lack experimental data for finite monoid relations (namely, a comparison with existing tools such as FAST [4], LASH [36] or TReX [2] on this class), as our implementation of finite monoid affine transformation class is still underway.

Table 1 shows the results of the comparison between the older algorithms described in [8, 7] (denoted as **old**) and the algorithm in Fig. 1 for difference bounds relations $d_{1,...,6}$ and octagonal relations $o_{1,...,6}$. The tests have been performed on both **compact** (minimum number of constraints) and **canonical** (i.e. closed, for difference bounds and tightly closed, for octagons) relations. The **speedup** column gives the ratio between the **old** and **new** execution times. The experiments were performed on a 2.53GHz machine with 2.9GB of memory.

As shown in Table 1, the maximum observed speedup is almost 10^5 for difference bounds (d_4 in canonical form) and of the order of four for octagons. For the relations d_5 (canonical form), d_6 and o_6 the computation using older methods took longer than 10^6 msec. It is also worth noticing that the highest execution time with the new method was of 2.5 msec.

Table 2 compares FLATA with the FAST tool [4] on counter systems with one self loop labeled with a randomly generated deterministic difference bound relation. We generated 50 such relations for each size $N = 10, 15, 20, 25, 50, 100$. Notice that FAST usually runs out of memory for more than 25 variables, whereas FLATA can handle 100 variables in reasonable time (less than 8 seconds on average).

6.2 Reachability in Counter Automata

This section reports on experiments we have performed with the FLATA toolset, to decide reachability in several counter automata, modeling real-life systems. We have experimented on three sets of counter automata. The first set of models we considered is taken from [35], where an approach for verification of generic VHDL circuit designs based on translation to counter automata is presented. Traditional verification techniques for hardware systems usually assume that the state space of these systems is finite. The approach presented in [35] aims at verification of parameterized VHDL

Relation		new	compact		canonical	
			old	**speedup**	**old**	**speedup**
d_0	$(x - x' = -1) \wedge (x = y')$	0.18	0.7	3.89	38.77	215.39
d_1	$(x - x' = -1) \wedge (x' = y')$	0.18	18.18	101.0	38.77	215.39
d_2	$(x - x' = -1) \wedge (x = y') \wedge (x - z' \leq 5) \wedge (z = z')$	1.2	26.5	22.1	33431.2	27859.3
d_3	$(x - x' = -1) \wedge (x = y') \wedge (x - z \leq 5) \wedge (z = z')$	0.6	32.7	54.5	33505.5	55841.7
d_4	$(x - x' = -1) \wedge (x = y) \wedge (x - z \leq 5) \wedge (z = z')$	0.5	702.3	1404.6	48913.8	97827.6
d_5	$(a = c) \wedge (b = a') \wedge (b = b') \wedge (c = c')$	1.8	5556.6	3087.0	$> 10^6$	∞
d_6	$(a - b' \leq -1) \wedge (a - e' \leq -2) \wedge (b - a' \leq -2)$ $\wedge (b - c' \leq -1) \wedge (c - b' \leq -2) \wedge (c - d' \leq -1)$ $\wedge (d - c' \leq -2) \wedge (d - e' \leq -1 \wedge e - a' \leq -1)$ $\wedge (e - d' \leq -2) \wedge (a' - b \leq 4) \wedge (a' - c \leq 3)$ $\wedge (b' - c \leq 4 \wedge b' - d \leq 3) \wedge (c' - d \leq 4) \wedge (c' - e \leq 3)$ $\wedge (d' - a \leq 3 \wedge d' - e \leq 4) \wedge (e' - a \leq 4) \wedge (e' - b \leq 3)$	5.6	$> 10^6$	∞	$> 10^6$	∞
o_1	$(x + x' = 1)$	0.21	0.91	4.33	0.91	4.33
o_2	$(x + y' \leq -1) \wedge (-y - x' \leq -2)$	0.29	0.85	2.93	0.84	2.9
o_3	$(x \leq x') \wedge (x + y' \leq -1) \wedge (-y - x' \leq -2)$	0.32	0.93	2.91	0.94	2.94
o_4	$(x + y \leq 5) \wedge (-x + x' \leq -2) \wedge (-y + y' \leq -3)$	0.21	3.67	17.48	13.52	64.38
o_5	$(x + y \leq 1) \wedge (-x \leq 0) \wedge (-y \leq 0)$	1.2	20050.9	16709.1	$> 10^6$	∞
o_6	$(x \geq 0) \wedge (y \geq 0) \wedge (x' \geq 0) \wedge (y' \geq 0)$ $\wedge (x + y \leq 1) \wedge (x' + y' \leq 1) \wedge (x - 1 \leq x')$ $\wedge (x' \leq x + 1) \wedge (y - 1 \leq y') \wedge (y' \leq y + 1)$	2.5	$> 10^6$	∞	$> 10^6$	∞

Table 1: Comparison with older algorithms on difference bounds and octagons. Times are in milliseconds.

vars	FLATA			FAST				
	done	av.	E_T	done	av.	E_T	E_M	E_B
10	50	1.5	0	49	0.6	0	0	1
15	50	1.6	0	31	10.5	17	0	2
20	50	1.6	0	4	3.4	9	8	29
25	50	1.6	0	2	4.2	2	10	36
50	50	1.6	0	0	-	0	0	50
100	49	7.7	1	0	-	0	0	50

(a) – matrix density 3%

vars	FLATA			FAST				
	done	av.	E_T	done	av.	E_T	E_M	E_B
10	50	1.5	0	22	6.9	23	1	4
15	50	1.5	0	1	20.6	4	3	42
20	50	1.6	0	0	-	1	0	49
25	43	1.7	7	0	-	0	0	50
50	50	2.3	0	0	-	0	0	50
100	42	5.5	8	0	-	0	0	50

(b) – matrix density 10%

Table 2: Comparison with FAST (MONA plugin) on deterministic difference bounds. Times are in seconds. E_T – timeout 30 s, E_B – BDD too large, E_M – out of memory

components with infinite state space. These generic components allow for the creation of libraries of reusable hardware, and are therefore commonplace in practice.

The translation to counter automata described in [35] maps bit variables to control locations, and integer variables to counters. Various safety properties are encoded as bit variables whose values are equivalent to propositional logic formulae representing the bad (unsafe) states.

We report on three out of the total five models from [35] (the remaining two are currently beyond the capabilities of FLATA and are a source of motivation for our future work). For the component of a hardware `counter`, we check whether overflow of a parametric bound is possible. For the `register` component, we check if the reset works correctly. The `synlifo` is a synchronous LIFO component with push and pop operations, which implements signals empty and full. The property checks if these signals are set correctly for a LIFO container of arbitrary size. Additionally, we have created buggy versions of these models (`counter-bug`, `register-bug`, and `synlifo-bug`).

The second set of examples are counter automata generated from programs with singly-linked lists, using the approach described in [11]. The main idea is that the set of heaps generated by a program with a finite number of local variables can be represented by a finite number of shape graphs, and the (unbounded) lengths of various list segments can be tracked by counters. The result of the translation of a program with lists is a counter automaton whose transition semantics is in bisimulation with the original program.

For all singly-linked list programs, we check that there are no null pointer dereferences. The `InsDel` program inserts elements to an empty list non-deterministically many times while counting the number of insertions and then traverses this number of elements and deletes each traversed element. Here we also check whether the list is empty when the program ends. The `ListCounter` program counts the length of a list during a first traversal of the list, which increments a counter. Here we also check that the length of the list is unchanged, and equal to the final value of the counter. The `ListReversal` is a textbook program that returns a list containing the same elements as the input list, in the reversed order. The reversal is done in place, by changing the links between the cells, instead of creating a copy of the input list. Here we also check that the lengths of the input list equals the length of the output list.

The third set of counter automata models are given by the decision procedure of the array logic SIL (Singly Indexed Logic), described in [25]. The decidability of the satisfiability problem for SIL encodes the set of models of a formula as the union of sets of traces of a set of flat counter automata with difference bounds constraints, whose emptiness is known to be decidable e.g., [14, 20]. Since FLATA is guaranteed to terminate on flat models with ultimately periodic relations on loops, we can use it as a solver for the SIL logic.

We report on two SIL formulae which arise as verification conditions for loop invariants of array manipulating programs. The <u>array rotation</u> program rotates an array by one element to the left and the <u>array split</u> program splits an array to negative and non-negative parts. Both formulae are translated to a pair of counter automata, hence Table 3 reports the results of analyses as pairs $x; y$ (lines `split` and `rotation`).

We have compared the performance of FLATA with other existing tools for the analysis of counter automata. The FAST tool [4] is based on the acceleration of loops with finite monoid affine transformations – we have used FAST with four different plugins for solving Presburger queries: MONA [27] (finite automata), Prestaf [17] (shared automata), Omega [19] (quantifier elimination) and LASH [36] (numeric decision diagrams). The ARMC tool [33] uses predicate abstraction and interpolation-based abstraction refinement. The ASPIC tool [23] uses widening-based abstract interpretation. Table 3 gives the performance comparison of various tools for the reachability analysis of counter automata on the set of examples described above.

One can notice from the times reported in Table 3 that tools based on abstraction (ARMC, ASPIC) are in general faster than the ones based on precise acceleration of loops (FAST, FLATA), the reason being that abstraction greatly simplifies the models used in the analysis. On the other hand, the imprecision introduced often leads to "don't know" answers (ASPIC) or extra iterations of the abstraction-refinement loop (ARMC).

Another problem of the methods purely based on abstraction is the `modulo` counter automaton depicted in Figure 14. In order to prove that this automaton is empty, one must prove that "x is even" is an invariant of the system. In our experience, FAST and FLATA could prove this property, whereas ARMC and ASPIC could not (cf. Table 1).

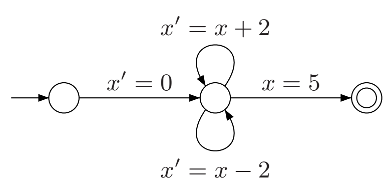


Fig. 14: The modulo counter automaton.

7 Conclusion and Future Work

We presented a new, scalable algorithm for computing the transitive closure of ultimately periodic relations. We show that this algorithm is applicable to difference bounds, octagonal and finite monoid affine relations, as all three classes are shown to be ultimately periodic. Experimental results show great improvement in the time needed to compute transitive closures of difference bounds and octagonal relations, as well as in the number of variables which can be handled.

This algorithm, together with its semi-algorithmic extension for computing the transitive closure of disjunctive relations, is the core of a new method for the reachability analysis of counter automata. We have implemented this method in a reachability analysis tool for counter automata, called FLATA, and we have experimented on several test cases, stemming from real applications. Experimental evidence and the comparison with existing tools show promising results.

model	$\|Q\|$	$\|T\|$	FLATA	ARMC	FAST		ASPIC
					MONA	PRESTAF	
counter	6	13	0.8	0.1	0.7	0.7	0.1
counter-bug	6	13	0.9	0.1	0.7	0.7	D
register	10	49	0.9	0.1	0.14	0.5	0.1
register-bug	10	49	0.9	0.1	0.14	0.5	D
synlifo	43	1006	25	1.5	500	112	1.7
synlifo-bug	43	1006	15	1.5	419	101	D
insdel-bug	31	37	0.4	0.2	225.3	21.4	D
listreversal	97	107	6.2	42.6	T	E	0.2
listreversal-bug	99	107	2.4	0.2	T	T	D
listcounter	31	35	1.0	2.9	178.7	14	0.1
listcounter-bug	31	34	0.5	0.1	T	T	D
split	61	329	9.6	1.2	C	C	C
rotation	33	147	5.7	0.6	C	C	C
modulo	3	4	0.7	T	0.1	0.4	D

Table 3: Reachability comparison (time in seconds). T – time-out 14min, E – segmentation fault, D – ”don’t know”, C – incompatible (non-deterministic) relations

So far, the reachability analysis implemented in FLATA is precise: we reduce the input counter automaton while preserving its input-output semantics. For this reason, the tool is sound and complete. In the future, we plan to combine our algorithm with abstraction techniques, in order to be able to treat larger systems. We plan on investigating both over-approximation (incompleteness) and under-approximation (unsoundness) techniques. Another future direction consists in enriching the input model by adding procedures and concurrency (hierarchical parallel counter automata) and developping compositional analysis methods for the new extensions of the model.

References

1. A. Annichini, E. Asarin, and A. Bouajjani. Symbolic techniques for parametric reasoning about counter and clock systems. In CAV ’00, pages 419–434. Springer, 2000.
2. A. Annichini, A. Bouajjani, and M. Sighireanu. Trex: A tool for reachability analysis of complex systems. In CAV ’01, pages 368–372. Springer, 2001.
3. R. Bagnara, P. M. Hill, and E. Zaffanella. An improved tight closure algorithm for integer octagonal constraints. In VMCAI ’08, 2008.
4. S. Bardin, J. Leroux, and G. Point. Fast extended release. In CAV’06, volume 4144, pages 63–66. Springer Verlag, 2006.
5. B. Boigelot. Symbolic Methods for Exploring Infinite State Spaces, volume PhD Thesis, Vol. 189. Collection des Publications de l’Université de Liège, 1999.
6. Ahmed Bouajjani, Marius Bozga, Peter Habermehl, Radu Iosif, Pierre Moro, and Tomás Vojnar. Programs with lists are counter automata. In CAV, pages 517–531, 2006.
7. M. Bozga, C. Gîrlea, and R. Iosif. Iterating octagons. In TACAS ’09, pages 337–351. Springer, 2009.
8. M. Bozga, R. Iosif, and Y. Lakhnech. Flat parametric counter automata. Fundamenta Informaticae, 91:275–303, 2009.

9. Marius Bozga, Peter Habermehl, Radu Iosif, Filip Konecný, and Tomás Vojnar. Automatic verification of integer array programs. In CAV, pages 157–172, 2009.
10. Marius Bozga, Radu Iosif, and Filip Konen. Fast acceleration of ultimately periodic relations. In Computer Aided Verification, Lecture Notes in Computer Science 6174, pages 227–242. Springer Verlag, 2010.
11. Marius Bozga, Radu Iosif, and Swann Perarnau. L2ca. `http://www-verimag.imag.fr/L2CA.html`.
12. Marius Bozga, Radu Iosif, and Swann Perarnau. Quantitative separation logic and programs with lists. In IJCAR, pages 34–49, 2008.
13. Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement for symbolic model checking. J. ACM, 50(5):752–794, 2003.
14. H. Comon and Y. Jurski. Multiple Counters Automata, Safety Analysis and Presburger Arithmetic. In CAV '98, volume 1427 of LNCS, pages 268 – 279. Springer, 1998.
15. Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. Introduction to Algorithms. McGraw-Hill Higher Education, 2nd edition, 2001.
16. P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In Conference Record of the Fifth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pages 84–97, Tucson, Arizona, 1978. ACM Press, New York, NY.
17. J.M. Couvreur. Prestaf. `http://altarica.labri.fr/forge/projects/3/wiki/PresTAF`.
18. B. Dutertre and L. de Moura. The yices smt solver. `http://yices.csl.sri.com/`.
19. William Pugh et. al. Omega. `http://www.cs.umd.edu/projects/omega/`.
20. A. Finkel and J. Leroux. How to compose presburger-accelerations: Applications to broadcast protocols. In FST TCS '02, pages 145–156. Springer, 2002.
21. Alain Finkel, Étienne Lozes, and Arnaud Sangnier. Towards model-checking programs with lists. In ILC, pages 56–86, 2007.
22. http://www-verinew.imag.fr/FLATA.html.
23. Laure Gonnord. Aspic. `http://laure.gonnord.org/pro/aspic/aspic.html`.
24. Peter Habermehl, Radu Iosif, Adam Rogalewicz, and Tomás Vojnar. Proving termination of tree manipulating programs. In ATVA, pages 145–161, 2007.
25. Peter Habermehl, Radu Iosif, and Tomáš Vojnar. A logic of singly indexed arrays. In Proceedings of the 15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning, LPAR '08, pages 558–573, 2008.
26. O. H. Ibarra. Reversal-bounded multicounter machines and their decision problems. Journal of the Association for Computing Machinery, 25(1):116 – 133, January 1978.
27. Nils Klarlund and Anders Møller. Mona. `http://www.brics.dk/mona/`.
28. J. Leroux and G. Sutre. Flat counter automata almost everywhere! In Proc. ATVA, volume 3707 of LNCS, pages 489–503. Springer, 2005.
29. A. Miné. The octagon abstract domain. Higher-Order and Symbolic Computation, 19(1):31–100, 2006.
30. M. Minsky. Computation: Finite and Infinite Machines. Prentice-Hall, 1967.
31. M. Presburger. Über die Vollstandigkeit eines gewissen Systems der Arithmetik. Comptes rendus du I Congrés des Pays Slaves, Warsaw 1929.
32. Christophe Reutenauer. The mathematics of Petri nets. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1990.
33. Andrey Rybalchenko. Armc. `http://www7.in.tum.de/~rybal/armc/`.
34. B. De Schutter. On the ultimate behavior of the sequence of consecutive powers of a matrix in the max-plus algebra. Linear Algebra and its Applications, 307:103–117, 2000.

35. Ales Smrcka and Tomás Vojnar. Verifying parametrised hardware designs via counter automata. In Haifa Verification Conference, pages 51–68, 2007.
36. P. Wolper and B. Boigelot. Verifying systems with infinite but regular state spaces. In CAV '98, pages 88–97. Springer, 1998.