

Representing Arithmetic Constraints with Finite Automata: An Overview

Bernard Boigelot

Pierre Wolper

Université de Liège

Motivation

- Linear numerical constraints are a very common and useful formalism (our particular motivation: verification of infinite-state systems).
- Many tools exist for dealing with such constraints. However,
 - nonconvex sets can be difficult to handle and usually do not have a normal form;
 - the full first-order theory is not always supported;
 - the combined use of integer and real variables (e.g. for representing periodic sets) is not handled.

Motivation (continued)

- Finite automata can represent sets of integer vectors, including all the Presburger definable sets.
- Sets of real vectors can be similarly represented by finite automata on infinite words.
- A good part of the theory is well known, the problem is to turn this approach into a usable technology.
- This means taking advantage of the special structure of the automata that are used, finding the best algorithms and producing an optimized implementation.

Representing sets of Integers and Reals with Finite Automata

- For a base $r > 1$, reals are encoded by infinite words (finite words for integers) built on the alphabet $\{0, \dots, r - 1, \star\}$.
- Negative numbers are encoded using r 's complement. The encoding of the integer part must be sufficiently long for the leading digit to be 0 for positive, $r - 1$ for negative number.
- A number has an infinite number of encodings since the leading digit can be repeated. Rationals x/y where all factors of y are factors of r have two encodings with the same integer length.

Examples : ($^\omega$ represents infinite repetition)

$$\begin{aligned}L_2(3.5) &= 0^+11\star 1(0)^\omega \cup 0^+11\star 0(1)^\omega \\L_2(-4) &= 1^+00\star (0)^\omega \cup 1^+011\star (1)^\omega \\L_{10}(11/2) &= 0^+5\star 5(0)^\omega \cup 0^+5\star 4(9)^\omega\end{aligned}$$

Representing sets of Integers and Reals with Finite Automata (continued)

- Vectors with n components are encoded by an n -tuple of words of identical integer length, or by a single word over the alphabet $\{0, \dots, r - 1\}^n \cup \{\star\}$.

Example : In base 2, the vector $(-2, 12.3)$ can be encoded as $(11110 \star 0^\omega, 01100 \star 01[1001]^\omega)$, or as $(1, 0)(1, 1)(1, 1)(1, 0)(0, 0) \star (0, 0)(0, 1)[(0, 1)(0, 0)(0, 0)(0, 1)]^\omega$.

- To limit the size of the alphabet, the digits of the various components of the vector are read serially, in a round robin way (*serial encoding*).

Example : In the serial encoding, $(-2, 12.3)$ can be encoded as $1011111000 \star 0001[01000001]^\omega$.

Representing sets of Integers and Reals with Finite Automata (continued 2)

- Finite automata are used to represent sets of n -component vectors.
- For real vectors, Büchi infinite word automata are used.
- To simplify operations, if a vector is in the set, all its encodings must be accepted.

Büchi Automata

- An infinite word (or ω -word) w over an alphabet Σ is a mapping $w : \mathbf{N} \rightarrow \Sigma$.
- A Büchi automaton is a five-tuple $A = (Q, \Sigma, \delta, Q_0, F)$, where
 - Q is a finite set of states;
 - Σ is the input alphabet;
 - δ is the transition function and is of the form $\delta : Q \times \Sigma \rightarrow 2^Q$ (nondeterministic) or $\delta : Q \times \Sigma \rightarrow Q$ (deterministic);
 - $Q_0 \subseteq Q$ is a set of initial states (a singleton for deterministic automata);
 - F is a set of accepting states.

Büchi Automata (continued)

- A run π of a Büchi automaton on an ω -word w is a mapping $\pi : \mathbf{N} \rightarrow Q$ that satisfies :
 - $\pi(0) \in Q_0$, i.e. the run starts in an initial state;
 - For all $i \geq 0$, $\pi(i + 1) \in \delta(\pi(i), w(i))$ (nondeterministic automata) or $\pi(i + 1) = \delta(\pi(i), w(i))$ (deterministic automata), i.e. the run respects the transition function.
- Let $inf(\pi)$ be the set of states that occur infinitely often in a run π . A run π is said to be accepting if $inf(\pi) \cap F \neq \emptyset$.
- A co-Büchi automaton is defined exactly as a Büchi automaton except that its accepting runs are those for which $inf(\pi) \cap F = \emptyset$.

The representation : Real Vector Automata

Let $n > 0$ and $r > 1$ be integers. A base- r n -dimension serial *Real Vector Automaton* (*RVA*) is a Büchi automaton $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ automaton over the alphabet $\Sigma = \{0, \dots, r-1\} \cup \{\star\}$, such that

- Every word accepted by \mathcal{A} is a serial encoding in base r of a vector in \mathbf{R}^n , and
- For every vector $\vec{x} \in \mathbf{R}^n$, \mathcal{A} accepts either all the encodings of \vec{x} in base r , or none of them.

Büchi Automata and Weak Automata

The automata corresponding to sets definable in the first-order theory of linear relations will be shown to be *weak*.

For a Büchi automaton $A = (Q, \Sigma, \delta, Q_0, F)$ to be *weak*, there has to be a partition of its state set Q into disjoint subsets Q_1, \dots, Q_m such that

- for each of the Q_i either $Q_i \subseteq F$ or $Q_i \cap F = \emptyset$; and
- there is a partial order \leq on the sets Q_1, \dots, Q_m such that for every $q \in Q_i$ and $q' \in Q_j$ for which, for some $a \in \Sigma$, $q' \in \delta(q, a)$ ($q' = \delta(q, a)$ in the deterministic case), $Q_j \leq Q_i$.

Constructing RVAs from linear equations

The problem is to construct an RVA representing the set S of all the solutions $\vec{x} \in \mathbf{R}^n$ of $\vec{a} \cdot \vec{x} = b$, given $n \geq 0$, $\vec{a} \in \mathbf{Z}^n$ and $b \in \mathbf{Z}$.

A Decomposition of the Problem

- Separate the encodings into their integer part w_I and fractional part w_F .
- For the corresponding vectors, we have that $\vec{x}_I \in \mathbf{Z}^n$, $\vec{x}_F \in [0, 1]^n$, $\vec{x} = \vec{x}_I + \vec{x}_F$, and $\vec{a} \cdot \vec{x}_I + \vec{a} \cdot \vec{x}_F = b$.
- It follows that the language of encodings is

$$\bigcup_{\varphi(\beta)} \{w_I \in \Sigma^+ \mid \vec{a} \cdot [w_I \star 0^\omega]_r^n = \beta\} \cdot \{\star\} \cdot \{w_F \in \Sigma^\omega \mid \vec{a} \cdot [0^n \star w_F]_r^n = b - \beta\},$$

with $\varphi(\beta) = b - \alpha' \leq \beta \leq b - \alpha \wedge (\exists m \in \mathbf{Z})(\beta = m \gcd(a_1, \dots, a_n))$
where $\vec{a} = (a_1, \dots, a_n)$, $\alpha = \sum_{a_i < 0} a_i$ and $\alpha' = \sum_{a_i > 0} a_i$.

Recognizing Integer Solutions

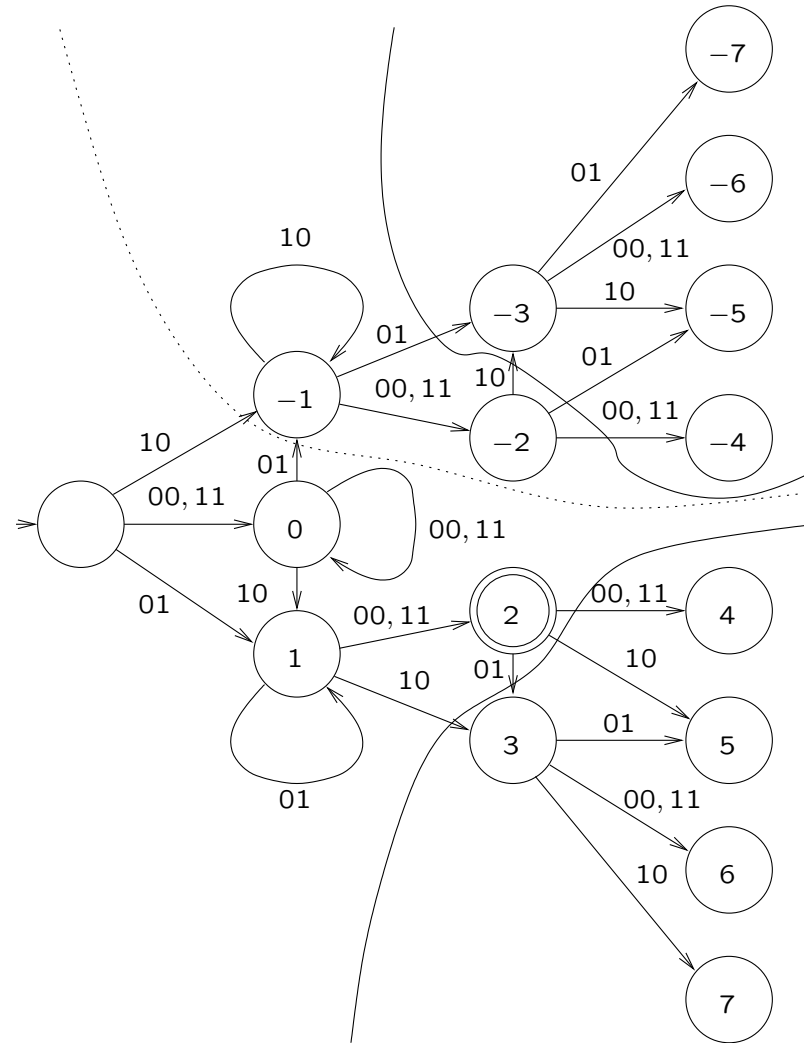
For an equation $\vec{a} \cdot \vec{x} = b$, construct a finite automaton $\mathcal{A}_{\vec{a}, b}$ that accepts all the finite words serially encoding in a given base r the integer solutions of the equation.

- The automaton has special states for reading sign digits;
- The other states are of the form (γ, i) , where γ is an integer and $0 \leq i \leq n - 1$ denotes a position in the serial reading of the vector digits;
- For a state s of the form $(\gamma, 0)$ we have that the path leading to it encode integer solutions to the equation $\vec{a} \cdot \vec{x} = \gamma$;
- The only accepting state s_F of $\mathcal{A}_{\vec{a}, b}$ is $(b, 0)$.
- From a state (γ, i) , an outgoing transition labeled d must lead to a state $s' = (\gamma', (i + 1) \bmod n)$ such that $\gamma' = r\gamma + a_1 d$ if $i = 0$ and $\gamma' = \gamma + a_{i+1} d$ if $i > 0$.

Recognizing Integer Solutions (continued)

- Only a finite number of states are needed : from a state $s = (\gamma, 0)$ such that $|\gamma| > (r - 1) \sum_{i=1}^n |a_i|$, one can only reach states s' such that $|\gamma'| > |\gamma|$, one can thus prune all such states with $|\gamma| > |b|$.
- The automaton is deterministic and essentially minimal (possible non minimality of sign states).
- It is convenient to construct the automaton backwards from its accepting state.
- One can merge the construction of automata for different values of the right-hand side of the equation.
- The number of states of the automaton is logarithmic in the value of b and linear in the value of the elements of \vec{a} .

An Example : the automaton for $x - y = 2$



Recognizing Fractional Solutions

- The other states are of the form (γ, i) , where γ is an integer and $0 \leq i \leq n - 1$ denotes a position in the serial reading of the digits;
- For a state s of the form $(\gamma, 0)$ we have that the infinite paths starting from s are exactly the solutions to the equation $\vec{a} \cdot \vec{x} = \gamma$;
- From a state (γ, i) , an outgoing transition labeled d must lead to a state $s' = (\gamma', (i + 1) \bmod n)$ such that $\gamma' = r\gamma - a_1d$ if $i = 0$ and $\gamma' = \gamma - a_{i+1}d$ if $i > 0$.
- Only a finite number of states are needed : only states $(\gamma, 0)$ with $\alpha = \sum_{a_i < 0} a_i \leq \gamma \leq \alpha' = \sum_{a_i > 0} a_i$ can appear in accepting computations.
- The automaton is deterministic and weak; all states within $[\alpha, \alpha']$ are accepting. It can be minimized (Löding's procedure for weak deterministic automata).

Constructing RVAs from linear inequations

- The starting point is an inequation $\vec{a}.\vec{x} \leq b$.
- One uses the same type of decomposition as for equations.
- For the integer part, all states $(\gamma, 0)$ with $\gamma \leq b$ are accepting.
- The construction can again be done backwards, but the result is no longer a deterministic automaton. However, it can be efficiently determinized since its states are *ordered*.

Computing with RVAs

- The algorithms we have given allow the construction of RVAs for atomic formulas.
- To construct RVAs for other first-order formulas, we need to be able to apply to RVAs Boolean operations as well as quantifiers.
- The Boolean combination of RVAs can quite directly be computed by a product construction. Note that complementation is not a problem since we are dealing with deterministic weak automata.
- Cartesian product (useful for combining RVAs operating on different variables) can similarly be computed by a product operation.
- Quantification is more delicate.

Applying Existential Quantification to RVAs

Given a set S represented by an RVA \mathcal{A} , the problem is to compute an RVA representing the set

$$\exists_i^{\mathbf{R}} S = \{(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) \in \mathbf{R}^{n-1} \mid (\exists x_i \in \mathbf{R})((x_1, \dots, x_n) \in S)\}.$$

- This is quite naturally done by a projection operation.
- The projection operation is fairly straightforward, but leads to two problems :
 - The resulting automaton is nondeterministic, which is problematic if further operations (especially complementation) have to be applied to it;
 - The projected out component might impose a now unnecessary minimum length on the the accepted encodings. This is not very difficult to fix, but is somewhat tricky to do efficiently.

Determinizing RVAs

- The RVA obtained after projection is a nondeterministic infinite word Büchi automaton.
- Determinization procedure for Büchi automata exist, but are not easily usable in practice.
- However, we are dealing with *weak* automata. These can be viewed both as Büchi and co-Büchi automata, and for the latter there is a simple determinization procedure.
- The question is then whether the resulting deterministic automaton will remain weak. For general automata it need not be, but for RVAs corresponding to linear arithmetic formulas, we will see that it is always so.

Determinizing co-Büchi automata

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a nondeterministic co-Büchi automaton. The deterministic co-Büchi automaton $A' = (Q', \Sigma, \delta', q'_0, F')$ defined as follows accepts the same ω -language.

- $Q' = 2^Q \times 2^Q$.
- $q'_0 = (\{q_0\}, \emptyset)$.
- For $(S, R) \in Q'$ and $a \in \Sigma$, δ' is defined by
 - if $R = \emptyset$, then $\delta'((S, R), a) = (T, T \setminus F)$ where $T = \{q \mid \exists p \in S \text{ and } q \in \delta(p, a)\}$;
 - if $R \neq \emptyset$, then $\delta'((S, R), a) = (T, U \setminus F)$ where $T = \{q \mid \exists p \in S \text{ and } q \in \delta(p, a)\}$, and $U = \{q \mid \exists p \in R \text{ and } q \in \delta(p, a)\}$.
- $F' = 2^Q \times \emptyset$.

Staying within weak automata

- The proof that the automaton obtained after projecting arithmetic sets remain weak goes through topology.
- A first result is that the sets defined in the linear first-order theory of the integers and reals is in the topological class $F_\sigma \cap G_\delta$ of the Borel hierarchy (topology based on Euclidean distance)..
- The interesting fact is that the languages accepted by weak deterministic automata have a similar characterization.

Automata and the Topology on Words

Consider the topology on infinite words induced by the distance

$$d(w, w') = \frac{1}{|\text{commonprefix}(w, w')| + 1}.$$

Theorem [SW74,MS97] : The ω -regular languages in the class $F_\sigma \cap G_\delta$ are exactly those accepted by *weak* deterministic automata.

Note the previous result does not guarantee that any automaton built for a set in $F_\sigma \cap G_\delta$ is weak, but it is not far from being so.

Automata and the Topology on Words (continued)

Definition: An automaton is *inherently weak* if none of its strongly connected components contains both accepting and nonaccepting cycles.

Theorem: Any deterministic Büchi automaton accepting a language in $F_\sigma \cap G_\delta$ is inherently weak.

Proof:

- For any language L accepted by a deterministic automaton that is not inherently weak, $(\exists w_1)(\forall \varepsilon_1 > 0)(\exists w_2)(\forall \varepsilon_2 > 0)(\exists w_3) \dots$
 - $d(w_i, w_{i+1}) < \varepsilon_i$ for $i = 1, 2, 3, \dots$,
 - $w_1, w_3, w_5, \dots \in L$, and
 - $w_2, w_4, w_6, \dots \notin L$.
- No language with this property can be accepted by a weak automaton.

Topology: from Vectors to Words

The topologies on vectors and words are different. To use the fact that we are dealing with sets in $F_\sigma \cap G_\delta$ in the automaton context, we need the following.

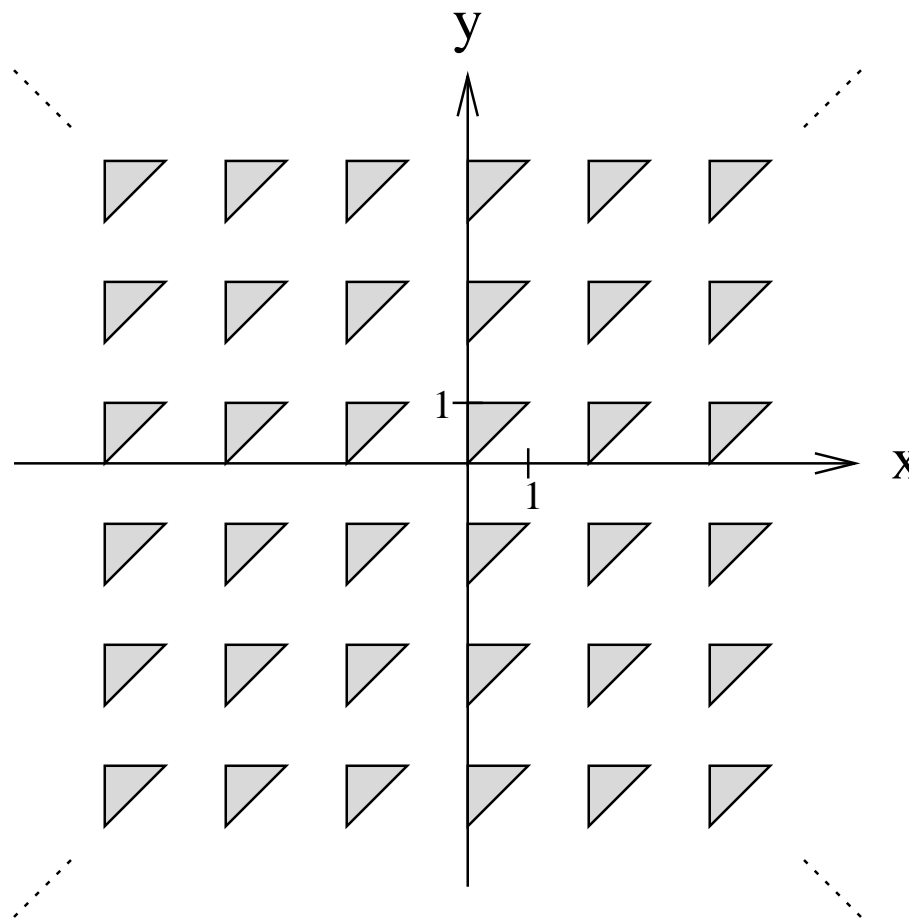
Theorem: If $S \subseteq \mathbf{R}^n$ is a set in $F_\sigma \cap G_\delta$ (wrt Euclidean distance), then $L_r(S)$ is a set in $F_\sigma \cap G_\delta$ (wrt distance on words).

Conclusion

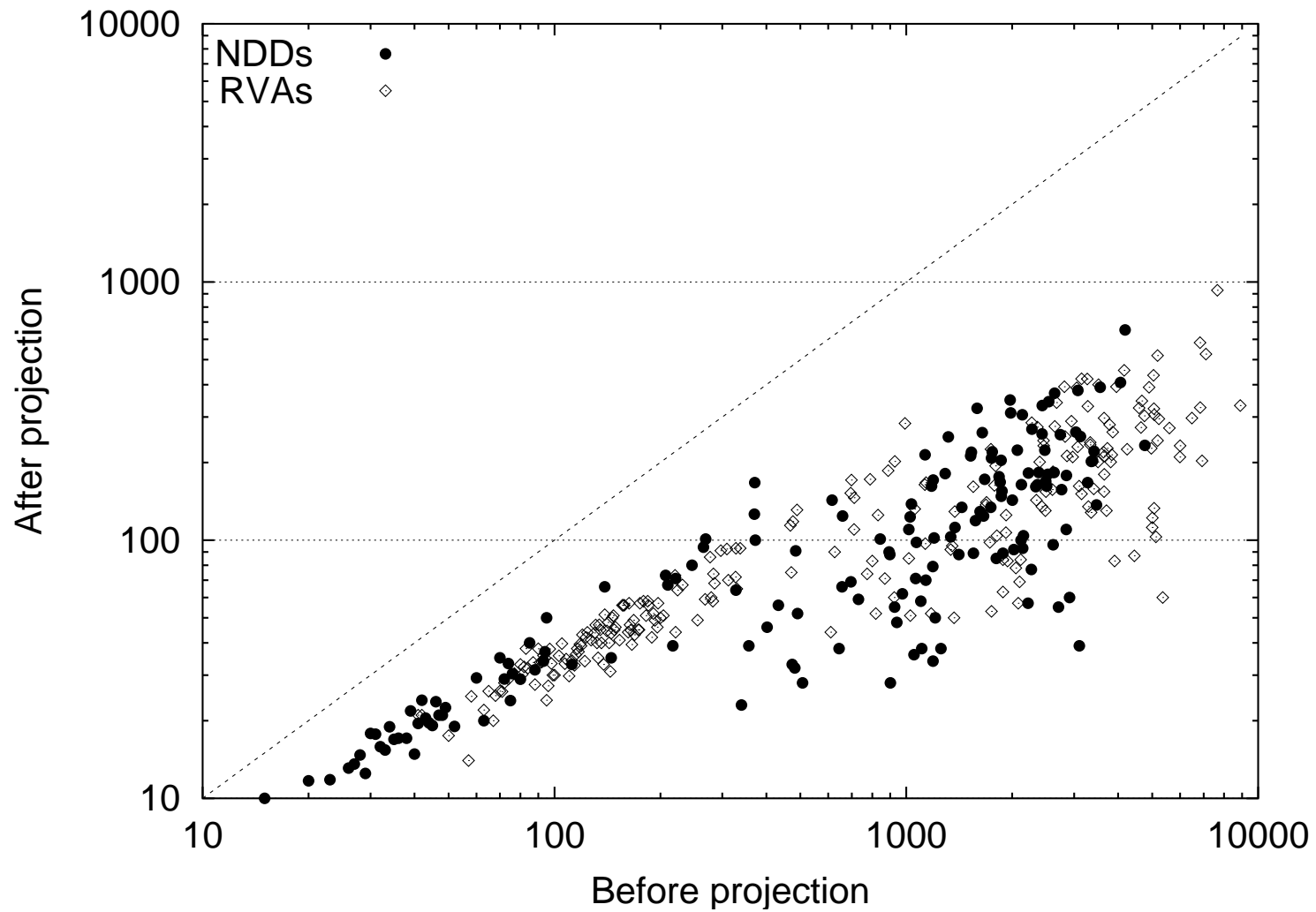
The automata obtained after determinizing RVAs obtained from linear arithmetic formulas are inherently weak and can easily be transformed into weak deterministic automata.

An Example

$$\left\{ (x_1, x_2) \in \mathbf{R}^2 \mid \begin{array}{l} (\exists x_3, x_4 \in \mathbf{R}) \\ (\exists x_5, x_6 \in \mathbf{Z}) \end{array} \left(\begin{array}{l} (x_1 = x_3 + 2 \cdot x_5) \wedge \\ (x_2 = x_4 + 2 \cdot x_6) \wedge \\ (x_3 \geq 0 \wedge x_4 \leq 1 \wedge x_4 \geq x_3) \end{array} \right) \right\}$$



Performance: the impact of projection and determinization



Conclusions

- Technique for dealing with linear arithmetic constraints that has valuable features :
 - Can handle reals and integers combined;
 - Can handle arbitrary first-order formulas;
 - Provides a normal form (minimized weak deterministic automaton) that is easily exploitable (satisfiability, equivalence, visualization, . . .);
 - Implemented in a tool (LASH).

Conclusions (continued)

- Experiments show that the technique is quite usable. It usually does not outperform other techniques but, once computed, the automaton representing a set provides a lot of information.
- From an automata-theoretic point of view, one conclusion is that arithmetic automata have a lot of special structure that influences the algorithms that can be used and their performance.
- There is still more structure to discover and exploit, for instance automata corresponding to formulas without integers are counter-free.