Automata-based Data Structures for Representing Sets of Vectors

Bernard Boigelot
Université de Liège

Joint work with: Pierre Wolper, Louis Bronne, Stéphane Rassart, Sébastien Jodogne, Julien Brusten, Véronique Bruyère, Jérôme Leroux, Jean-François Degbomont, ...
**Goal:** Obtaining **good data structures** for representing the sets handled during state-space exploration of programs.

**Example:** “Leaking gas burner”.
Requirements

- Domain $\mathbb{Z}^n$ or $\mathbb{R}^n$.

- **Sufficient expressive power** for handling linear constraints and discrete periodicities.

- **Algorithms** for:
  - computing Boolean combinations, set differences, projections, Cartesian products, . . . .
  - checking emptiness, inclusion and equality of sets.

- **Efficiency** and conciseness.
1st Approach: Formula-based Representations

Example:

\[
\{ (x_1, x_2) \in \mathbb{R}^2 \mid (\exists x_3, x_4 \in \mathbb{R}) (\exists x_5, x_6 \in \mathbb{Z}) 
\left( (x_1 = x_3 + 2 \cdot x_5) \land 
(x_2 = x_4 + 2 \cdot x_6) \land 
(x_3 \geq 0 \land x_4 \leq 1 \land x_4 \geq x_3) \right) \}
\]
Advantages:

- The first-order theories $\langle \mathbb{Z}, + \leq \rangle$ (Presburger arithmetic) and $\langle \mathbb{R}, \mathbb{Z}, + \leq \rangle$ are
  - sufficiently expressive, and
  - decidable.

- Computing Boolean combinations, projections, complement of sets is immediate.

Drawbacks:

- Formulas cannot be easily simplified into a canonical form.
- Comparing sets (i.e., checking equality, inclusion or emptiness) is a costly operation.
2nd Approach: Automata-based Representations

Principles:

- Data values are encoded as words over some finite alphabet.
- This encoding relation maps sets of values onto languages.
- If the language $L$ encoding a set $S$ is regular, then any finite-state automaton that accepts $L$ forms a representation of $S$.

Advantages:

- Expressive enough for many applications.
- Large class of simple, efficient, and well-studied manipulation algorithms.
- Deterministic automata admit an easily computable canonical form.
Application to the Domain \( \mathbb{Z} \)

Principles:

- Integers are encoded in an integer base \( r > 1 \), most significant digit first.
- Numbers are encoded by the \( r \)'s complement method:

A word

\[ d_{p-1}d_{p-2}\cdots d_1d_0 \]

encodes the number

\[
\sum_{i=0}^{p-1} d_i r^i \geq 0 \quad \text{if } d_{p-1} = 0.
\]

\[
-r^p + \sum_{i=0}^{p-1} d_i r^i < 0 \quad \text{if } d_{p-1} = r - 1.
\]
The number of digits $p$ in the encodings of $z$ is not fixed, but must satisfy

$$-r^{p-1} \leq z < r^{p-1}.$$ 

Examples:

$$Enc_2(12) = 0^+1100$$
$$Enc_2(-7) = 1^+001.$$
Generalization to $\mathbb{Z}^n$

- **Vectors** can be encoded by reading repeatedly one digit for each component, in a fixed order.

- The component digits can be combined in several ways.

  **Synchronous encoding:**

  \[
  \text{Enc}_2((-4, 6, 3)) = (1, 0, 0)^+ (1, 1, 0)(0, 1, 1)(0, 0, 1).
  \]

  **Serial encoding:**

  \[
  \text{Enc}_2((-4, 6, 3)) = (100)^+ 110011001.
  \]
Number Decision Diagrams

Definition: A Number Decision Diagram (NDD) representing a set $S \subseteq \mathbb{Z}^n$ is a finite-state automaton accepting all the encodings of all the elements in $S$.

Advantages:

- Computing Boolean combinations, differences, Cartesian products of represented sets amounts to simple automata constructions.
- Deterministic NDDs admit an easily computable canonical form.

Question: Expressive enough?
Characterizing the Expressiveness of NDDs

Theorem [Büchi, Bruyère]: A set $S \subseteq \mathbb{Z}^n$ is representable by an NDD in a base $r > 1$ iff it can be defined in the first-order theory $\langle \mathbb{Z}, +, \leq, V_r \rangle$, where $V_r(z)$ is the greatest power of $r$ that divides $z$.

Theorem [Cobham, Semenov]: A set $S \subseteq \mathbb{Z}^n$ is representable by an NDD in any base $r > 1$ iff it can be defined in the first-order theory $\langle \mathbb{Z}, +, \leq \rangle$.

Question: Do NDDs lead to a practical decision procedure for Presburger arithmetic?
From Linear Equations to NDDs

An NDD representing the set

\[ \{ \vec{x} \in \mathbb{Z}^n \mid \vec{a}.\vec{x} = b \} \]

can be constructed by

- associating to each state \( q \) an integer \( \beta(q) \) such that any path ending in \( q \) reads a solution of \( \vec{a}.\vec{x} = \beta(q) \).
- starting the construction from an accepting state \( q_F \) such that \( \beta(q_F) = b \).
- applying a backward propagation rule:

\[ \beta(q') = r\beta(q) + \vec{a}.\vec{d} \]
\[ \beta(q) = \frac{\beta(q') - \vec{a}.\vec{d}}{r} \]
• creating an initial state $q_0$ connected to the other states by the rule

\[ \beta(q) = -\frac{\vec{a} \cdot \vec{d}}{r - 1} \]
Example: \(2x - y = -4\)
Generalization to Inequalities

A similar procedure can be used for obtaining a NDD representing the set

$$\{ \vec{x} \in \mathbb{Z}^n \mid \vec{a}.\vec{x} \leq b \}.$$  

- The propagation rule becomes

$$\beta(q) = \left\lfloor \frac{\beta(q') - \vec{a}.\vec{d}}{r} \right\rfloor.$$  

- The accepting states $q_F$ are those that satisfy

$$\beta(q_F) \leq b.$$  

- The resulting automaton is generally non deterministic (but can be efficiently determinized).
Projecting Sets Represented by NDDs

Let $A$ be a NDD representing a set $S \subseteq \mathbb{Z}^n$, and $i \in [1, n]$.

**Definition:** The projection of $S$ over the vector components that differ from $i$ is the set

$$\exists_i S = \{(z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n) \mid (z_1, \ldots, z_n) \in S\}.$$

**Problem:** In order to obtain a NDD representing $\exists_i S$, it is not sufficient to delete from transition labels the symbols corresponding to the $i$-th vector component.

Indeed, the resulting automaton does not always accept all the encodings of the vectors in $\exists_i S$. 
Illustration:

\[ S = \{(1, 5, 2)\} \]
\[ Enc_2(S) = (000)^+ 010 001 110 \]
\[ Enc_2(S)|_{1,3} = (00)^+ 00 01 10 \]
\[ \not= Enc_2[\exists_2 S] = (00)^+ 01 10 \]
Projecting Sets Represented by NDDs: Solution

Idea: For each tuple \( u \in \{0, r - 1\}^n \) of sign digits, if the automaton accepts a word of the form \( u^k w \), then it should also accept the words \( u^i w \) for all \( 0 < i < k \).

This property can be enforced by constructing a prefix automaton \( A_P \) that recognizes sign digits, and linking \( A_P \) to \( A \) by means of \( \varepsilon \) transitions.
Problem: This method is exponentially costly in \( n \).

Solution: Use a serial encoding of vectors, and build the prefix automaton in such a way that the tuple of sign digits that are not distinguished by \( \mathcal{A} \) are not separated.

This is done by the following procedure:

1. An initial prefix automaton is obtained by intersecting \( \mathcal{A} \) with an automaton accepting the language \( \{0, r - 1\}^n \).

2. During the exploration of \( \mathcal{A} \), upon finding two sign tuples that are handled identically by \( \mathcal{A}_P \) but lead to different states of \( \mathcal{A} \), the automaton \( \mathcal{A}_P \) is refined so as to separate them.
Deciding Presburger Arithmetic

NDDs thus provide a simple procedure for deciding Presburger arithmetic:

1. Using the constructions developed for equations and inequations, build NDDs representing the solutions of atomic formulas.

2. Apply Boolean connectives and negations by performing the corresponding operations on automata.

3. Handle quantifications by computing projections (existential), or complement/projection/complements (universal).

4. Check whether the resulting automaton accepts a nonempty language.

Note:

- Each universal quantifier requires a determinization step, which may potentially incur an exponential blowup.

- It has however been established that the size of the automata stays elementary [Klaedtke].
Overview of Other Operations

Algorithms have also been developed for applying the following operations to NDD-represented sets:

- Counting efficiently the number of vectors in a set.
- Deciding whether the closure of an affine transformation $\vec{z} \leftarrow A\vec{z} + \vec{b}$ preserves the representability of a set.
- Computing the image of a set by such a closure.
- Extracting from a NDD representing a Presburger set $S$ a formula describing $S$ [Leroux, Latour].
Example of Verification Run

Compilation statistics:
  number of gates : 0.
  number of processes : 3.
  number of variables : 4.
  total number of control locations : 11.
  number of synchronized transitions : 0.
  number of meta-transitions : 4.

Translating the transition relation...
  with transitions : 1647 NDD state(s).
  with synchronised transitions : 1647 NDD state(s).
  with transitions & meta-transitions : 4017 NDD state(s).

Translating the set of initial states...
  initial set : 218 NDD state(s).

Starting state-space exploration...
  interm. result : 638 NDD state(s), 3 states.
  interm. result : 1044 NDD state(s), 1000000003 states.
  interm. result : 1461 NDD state(s), 3999999999 states.
  interm. result : 2709 NDD state(s), 500000005499999997 states.
  interm. result : 4596 NDD state(s), 1500000006499999995 states.
  interm. result : 6409 NDD state(s), 3500000004499999994 states.
  interm. result : 7020 NDD state(s), 6499999997499999999 states.
  interm. result : 7808 NDD state(s), 7999999995000000000 states.
  interm. result : 8655 NDD state(s), 8999999994000000000 states.
  interm. result : 8658 NDD state(s), 9499999993500000000 states.
  interm. result : 8663 NDD state(s), 9999999993000000000 states.

Fixpoint reached in 11 step(s).
*** Program validated.

Runtime statistics:
  residual memory : 0 byte(s).
  max memory : 4344928 byte(s).
Extension to the domain $\mathbb{R}$

Principles:

• Real numbers are encoded as infinite words over $r$ digits $0, 1, \ldots, r - 1$, and a separator $\star$.

  Example: $Enc_2(3.5) = 0^+11 \star 1(0)^\omega \cup 0^+11 \star 0(1)^\omega$.

• Vectors from $\mathbb{R}^n$ are encoded as infinite words over either
  $\{0, 1, \ldots, r - 1\}^n \cup \{\star\}$ (synchronous encoding), or $\{0, 1, \ldots, r - 1, \star\}$
  (serial encoding).

• A Real Vector Automaton (RVA) representing a set $S \subseteq \mathbb{R}^n$ is a Büchi automaton accepting all the base-$r$ encodings of all the vectors in $S$.

Question: Expressive power?
From Linear Equations to RVA

Problem: Construct a RVA representing the set

\[ \{ \vec{x} \in \mathbb{R}^n \mid \vec{a} \cdot \vec{x} = b \} \].

Solution:

Consider a word \( w_I \ast w_F \in Enc_r(\vec{x}) \).

\[ \begin{align*}
   w_I \ast \vec{0}^\omega & \text{ encodes } \vec{x}_I \in \mathbb{Z}^n, \\
   \vec{0} \ast w_F & \text{ encodes } \vec{x}_F \in [0, 1]^n.
\end{align*} \]

\[ \vec{x}_I + \vec{x}_F = \vec{x} \]

\[ \Rightarrow \vec{a} \cdot \vec{x}_I + \vec{a} \cdot \vec{x}_F = b. \]
This equation decomposes into

\[ \vec{a}.\vec{x}_I = \beta_1 \quad \vec{a}.\vec{x}_F = \beta_2, \]

with \( \beta_1, \beta_2 \in \mathbb{Z} \) and \( \beta_1 + \beta_2 = b \).

In addition, \( \sum_{a_i < 0} a_i \leq \beta_2 \leq \sum_{a_i > 0} a_i \), and \( (\exists m \in \mathbb{Z})(\beta_1 = m. \gcd(a_1, \ldots, a_n)) \).

**Construction:** For each \( \beta_2 \in [\sum_{a_i < 0} a_i, \sum_{a_i > 0} a_i] \) such that \( b - \beta_2 \) is a multiple of \( \gcd(a_1, \ldots, a_n) \):

1. Build a NDD accepting the integer solutions of \( \vec{a}.\vec{x}_I = b - \beta_2 \).
2. Build an automaton accepting the solutions in \([0, 1]^n\) of \( \vec{a}.\vec{x}_F = \beta_2 \).
3. Connect these automata by transitions labeled by “\(*\)”. 

25
Building the Automaton for the Fractional Part

Problem: Construct a deterministic Büchi automaton representing in a base \( r > 1 \) the set

\[
\{ \vec{x} \in [0, 1]^n \mid \vec{a}.\vec{x} = b \}.
\]

Solution:

- Associate each state \( q \) with an integer \( \beta'(q) \), such that the language accepted from \( q \) encodes the solutions in \([0, 1]^n\) of \( \vec{a}.\vec{x} = \beta'(q) \).

- Mark all the states of the constructed automaton as accepting.
• Apply a forward propagation rule:

\[
\beta'(q) = \frac{\beta'(q') + \vec{a} \cdot \vec{d}}{r}
\]

\[
\beta'(q') = r \beta'(q) - \vec{a} \cdot \vec{d}
\]

Notes:

• The automaton is built forwards.

• If \( \beta'(q) \) is outside of \([\sum_{a_i<0} a_i, \sum_{a_i>0} a_i]\), then the state \( q \) can be discarded.

• The automata built for several values of \( b \) can share a part of their transition relation.
Example: $4x - 2y = -5$, $r = 2$. 
Cost of the Construction

- The size of the constructed RVA is

\[ O(f \sum_{1 \leq i \leq n} |a_i| \log_r |b|), \]

where \( f \) is equal to

- \( r^n \) with a synchronous encoding;
- \( n \) with a serial encoding.

- Time and memory costs are linear in this size.

Note: The construction algorithm can easily be adapted for generating a representation of

\[ \{ \vec{x} \in \mathbb{R}^n \mid \vec{a} \cdot \vec{x} \leq b \}. \]
The Expressive Power of RVA

- RVA representing the following sets can be built:
  - linear equations \( \{ \vec{x} \in \mathbb{R}^n \mid \vec{a}.\vec{x} = b \} \) and inequations \( \{ \vec{x} \in \mathbb{R}^n \mid \vec{a}.\vec{x} \leq b \} \).
  - the sets \( \mathbb{Z} \) and \( \mathbb{R} \).

- Algorithms are available for constructing, given RVA representing the sets \( S_1, S_2 \subseteq \mathbb{R}^n \), RVA representing:
  - \( S_1 \cup S_2 \), \( S_1 \cap S_2 \), \( S_1 \times S_2 \),
  - \( \overline{S_1} = \mathbb{R}^n \setminus S_1 \),
  - \( \exists_i S_1 = \{(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \mid (\exists x_i \in \mathbb{R})(x_1, \ldots, x_n) \in S_1)\} \).

- One can easily check whether a RVA represents the empty set.
Corollaries:

- The sets definable in the first-order theory \( \langle \mathbb{R}, \mathbb{Z}, +, \leq \rangle \) are representable by RVA.

- RVA provide an effective procedure for deciding this theory.

Note: It is also known that:

- the expressiveness of RVA in a base \( r > 1 \) corresponds to the first-order theory \( \langle \mathbb{R}, \mathbb{Z}, +, \leq, X_r \rangle \), where \( X_r \) is a base-dependent predicate.

- the sets that are representable by RVA in all bases \( r > 1 \) are exactly those that are definable in \( \langle \mathbb{R}, \mathbb{Z}, +, \leq \rangle \) [B., Brusten, Bruyère, Leroux].

Question: Can we make this decision procedure usable in practice?
Some Thoughts on the Problem

- Using finite automata as data structures for representing sets of integers is an interesting and potentially practical approach.
- Extending this representation to reals can be done quite naturally and yields a tool for handling the combined theory of integers and reals.
- Handling the reals is done by moving to automata on infinite words, which from a practical algorithmic point of view can be quite problematic. (In particular, complementing an automaton is difficult.)
- This is surprising since the additive theory of the reals is easier to handle than the corresponding theory over the integers.
Overview of Solution

We will show that:

• The sets definable in $\langle \mathbb{R}, \mathbb{Z}, +, \leq \rangle$ satisfy some topological properties.

• Automata representing such sets have a special structure.

• This special structure makes the use of much simpler algorithms possible.
Properties of Arithmetic Sets

• In $\mathbb{R}^n$, Boolean combinations of linear (in)equalities define Boolean combinations of open and closed sets.

• The theory $\langle \mathbb{R}, +, \leq, 1 \rangle$ admits quantifier elimination.

• Thus, only Boolean combinations of open and closed sets can be defined in this theory.

• This should translate to properties of the automata accepting the encodings of these sets.

• However, we are looking at the first-order additive theory $\langle \mathbb{R}, \mathbb{Z}, +, \leq \rangle$ of the reals and integers. Can we say something of the topology of the sets defined in this theory?
A Little Topological Background

Let $S$ be a set and $d(x, y)$ be a distance defined on the elements of $S$.

- A neighborhood of a point $x \in S$ is a set $N_\varepsilon(x) = \{y \in S \mid d(x, y) < \varepsilon\}$, with $\varepsilon > 0$.

- A set $U \subseteq S$ is open if for every $x \in U$, there exists $\varepsilon > 0$ such that $N_\varepsilon(x) \subseteq U$.

- A set $U \subseteq S$ is closed if the set $S \setminus U$ is open.
- The Borel hierarchy defines a collection of classes of sets, that starts with the following.
  - The closed sets: \( F \).
  - The open sets: \( G \).
  - The countable unions of closed sets: \( F_\sigma \).
  - The countable intersections of open sets: \( G_\delta \).
  - The countable intersections of sets in \( F_\sigma \): \( F_{\sigma \delta} \).
  - ...
Illustration:

\[ F \cap G \]

\[ F \sigma \cap G \delta \]

\[ B(F) = B(G) \]

\[ X \rightarrow Y \]: \( X \subset Y \); \( B(X) \): Boolean combinations of sets in \( X \).
Topological Properties of Arithmetic Sets

We consider the topology induced by the Euclidean distance

\[ d(\vec{x}, \vec{y}) = \left( \sum_{i=1}^{n} |x_i - y_i|^2 \right)^{1/2} \]

on the vectors of \( \mathbb{R}^n \).

**Theorem:** The sets definable in the first-order theory \( \langle \mathbb{R}, \mathbb{Z}, +, \leq \rangle \) belong to the topological class \( F_\sigma \cap G_\delta \).

**Proof:** If \( \varphi \) is a formula of \( \langle \mathbb{R}, \mathbb{Z}, +, \leq \rangle \) then so is \( \neg \varphi \). It is thus sufficient to prove that every definable set is in \( F_\sigma \).

Let \( \varphi \) be a formula of \( \langle \mathbb{R}, \mathbb{Z}, +, \leq \rangle \).
1. We replace each variable $x$ appearing in $\varphi$ by $x_I + x_F$, with

- $x_I$ the integer part of $x$.
- $x_F$ the fractional part of $x$.

Example:

$$(\exists x \in \mathbb{R}) \varphi \quad \longrightarrow \quad (\exists x_I \in \mathbb{Z})(\exists x_F \in \mathbb{R})$$

$$\quad (0 \leq x_F < 1 \land \phi[x/x_I + x_F])$$
2. Integer and fractional variables are then **separated in the atomic formulas**.

Example:

\[(x_I + x_F) = (y_I + y_F) + (z_I + z_F) \rightarrow (x_I = y_I + z_I \land x_F = y_F + z_F) \lor (x_I = y_I + z_I + 1 \land x_F = y_F + z_F - 1)\]

3. The quantifiers are then **distributed over the Boolean operators**, and unnecessary ones are eliminated.

Example:

\[(Qx_I \in \mathbb{Z})(\phi_I \alpha \phi_F) \rightarrow (Qx_I \in \mathbb{Z})(\phi_I) \alpha \phi_F,\]

where

- \(Q \in \{\exists, \forall\}, \ \alpha \in \{\land, \lor\},\)
- \(\phi_I\) only contains integer variables,
- \(\phi_F\) only contains fractional variables.
4. One then obtains a formula $\varphi$ of the form

$$B(\phi_{I}^{(1)}, \phi_{I}^{(2)}, \ldots, \phi_{I}^{(m)}, \phi_{F}^{(1)}, \phi_{F}^{(2)}, \ldots, \phi_{F}^{(m')}).$$

For each value $(a_1, a_2, \ldots, a_k) \in \mathbb{Z}^k$ of the free integer variable of this formula, each subformula $\phi_{I}^{(i)}$ is identically true or false. One thus has

$$\varphi \equiv \bigvee_{\vec{a} \in \mathbb{Z}^k} \left( (x_{I}^{(1)}, \ldots, x_{I}^{(k)}) = (a_1, \ldots, a_k) \right) \wedge B(a_1, \ldots, a_k)(\phi_{F}^{(1)}, \ldots, \phi_{F}^{(m')}).$$

The formula $\varphi$ hence defines a countable union of Boolean combinations of open and closed sets, thus a set in $F_{\sigma}$. 
Consider the topology on infinite words induced by the distance

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

**Theorem [Staiger, Wagner, Maler]:** The \( \omega \)-regular languages in the class \( F_\sigma \cap G_\delta \) are exactly those accepted by weak deterministic automata.

A **weak automaton** is a Büchi automaton whose set of states can be partitioned into sets \( Q_1, Q_2, \ldots, Q_m \) such that

- There exists a partial order \( \leq \) among these sets with the property that
  \[ (\forall q \in Q_i, q' \in Q_j)(q \rightarrow^* q' \Rightarrow Q_j \leq Q_i). \]

- Each \( Q_i \) contains only accepting or nonaccepting states.
The previous result does not guarantee that any automaton built for a set in $F_\sigma \cap G_\delta$ is weak, but we have the following.

**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, $(\exists w_1)(\forall \varepsilon_1 > 0)(\exists w_2)(\forall \varepsilon_2 > 0)(\exists w_3) \cdots$
  - $d(w_i, w_{i+1}) < \varepsilon_i$ for $i = 1, 2, 3, \ldots$,
  - $w_1, w_3, w_5, \ldots \in L$, and
  - $w_2, w_4, w_6, \ldots \notin L$.
- 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 $S \subseteq \mathbb{R}^n$ is a set in $F_\sigma \cap G_\delta$ (w.r.t. Euclidean distance), then $Enc_r(S)$ is a set in $F_\sigma \cap G_\delta$ (w.r.t. distance on words).

- The proof has to take into account the fact that every word is not necessarily an encoding of a vector.
- **Dual encodings** also prevent a direct mapping between the topologies.
- Nevertheless, the proof goes through for the class $F_\sigma \cap G_\delta$. 
Computing with RVA

From the results we have just seen, it follows that:

**Theorem:** Any deterministic RVA representing a set defined by a formula of the theory $\langle \mathbb{R}, \mathbb{Z}, +, \leq \rangle$ is inherently weak.

This property allows us to work with RVA that are weak automata and makes possible to use algorithms that are specific to this class of automata.

- **Linear equations and inequations:** The algorithms that we have studied produce weak automata.

- **Intersection, union, Cartesian product, projection:** One uses the corresponding operations on languages. The weak nature of the automata is preserved.
- **Complementation:**

  1. The weak RVA is viewed as a *co-Büchi automaton* (a word is accepted if there is an execution of the automaton on that word that *does not go infinitely often* through accepting states).

  2. For co-Büchi automata, there is a simple determinization procedure (see next slide).

  3. The resulting deterministic automaton is complemented into a Büchi automaton.

  4. The resulting automaton must be inherently weak and hence can easily be transformed into a weak automaton.

- **Satisfiability:** One checks whether the RVA has a *reachable accepting strongly connected component*. 
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 \( \omega \)-language.

- \( Q' = 2^Q \times 2^Q \).
- \( q'_0 = (\{q_0\}, \emptyset) \).
- For \( (S, R) \in Q' \) and \( a \in \Sigma \), \( \delta' \) 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 \( U = \{q \mid \exists p \in R \text{ and } q \in \delta(p, a)\} \).
- \( F' = 2^Q \times \emptyset \).
An Example

\[ \left\{(x_1, x_2) \in \mathbb{R}^2 \mid \left( \exists x_3, x_4 \in \mathbb{R} \right) \left( \exists x_5, x_6 \in \mathbb{Z} \right) \left( \begin{array}{l} x_1 = x_3 + 2 \cdot x_5 \\ x_2 = x_4 + 2 \cdot x_6 \\ x_3 \geq 0 \land x_4 \leq 1 \land x_4 \geq x_3 \end{array} \right) \right\} \]
Performance: the Impact of Projection and Determinization

![Graph showing the impact of projection and determinization](image-url)
Summary of Results

• These results do not introduce new algorithms, but show that known algorithms can be used in situations where this was a priori impossible.

• Weak deterministic automata have an easily computable canonical minimized form [Löding]. There is thus a canonical form for RVA.

• From a practical point of view, RVA are just as usable as automata representations of sets of integers.

• Experiments with our implementation (LASH) confirm this.