Automata-based Representations of Arithmetic Sets

Bernard Boigelot

Université de Liège
Introduction

Motivation: Representing the sets of configurations handled during symbolic state-space exploration.

For systems using integer variables, these sets are often combinations of

- linear constraints, and
- periodicities;

Problem: Formula-based representations do not provide efficient algorithms for

- computing disjunctions, and
- testing inclusion.

Solution: Automata-based representations.
Automata-based Representations of Sets

Principles:

• Data values are encoded by words over a finite alphabet;
• The encoding of a set $S$ is thus a language $L(S)$;
• A finite-state machine that accepts $L(S)$ is a representation of $S$.

Advantages:

• Closed under Boolean operators, projection, Cartesian product, . . . ;
• Simple manipulation algorithms;
• Canonical form;
• Sufficiently expressive for many data domains.
Number Decision Diagrams

Principles:

- The domain is $\mathbb{Z}^n$, with $n > 0$;
- Integers are encoded in a base $r > 1$, most or least significant digit first;
- Negative numbers are encoded by their $r$'s complement;
- 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:

\[
\begin{align*}
  Enc_2(12) &= 0^+1100 \\
  Enc_2(-7) &= 1^+001.
\end{align*}
\]
• **Vectors** are encoded by reading repeatedly one digit for each component, in a fixed order;

• The component digits can be combined in several ways.

  **Synchronous encoding:**
  
  \[ Enc_2((-4, 6, 3)) = (1, 0, 0)^+ (1, 1, 0)(0, 1, 1)(0, 0, 1) \]

  **Serial encoding:**
  
  \[ Enc_2((-4, 6, 3)) = (100)^+ 110011001. \]

• An NDD representing a set \( S \subseteq \mathbb{Z}^n \) is an automaton accepting all the encodings of all the elements in \( S \).
An NDD representing the set
\[
\{ \vec{x} \in \mathbb{Z}^n \mid \vec{a} \cdot \vec{x} = b \}
\]
can be constructed by

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

\[
\beta(q) = \frac{\beta(q') - \vec{a} \cdot \vec{d}}{r}
\]
Example: $2x - y = -4$
NDDs: Expressiveness

**Theorem:** 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:** 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 $\langle \mathbb{Z}, +, \leq \rangle$ (i.e., Presburger arithmetic).

Automata-based representations of sets thus provide a simple algorithm for deciding Presburger arithmetic.
From Integers to Reals

Motivation: Representing the sets of configurations of systems with mixed integer and real variables (e.g., timed systems).

Idea: Real numbers can be encoded as infinite words over an alphabet composed of

- $r$ digits: 0, 1, ..., $r - 1$, and
- a separator $*$ between the integer and the fractional parts.

Examples:

$$Enc_2(3.5) = 0^+ 11*1(0)^\omega \cup 0^+ 11*0(1)^\omega$$
$$Enc_2(-4) = 1^+ 00* (0)^\omega \cup 1^+ 011* (1)^\omega.$$
Vectors can be encoded synchronously or serially, assigning the same number of digits to the integer part of each component;

A Real Vector Automaton (RVA) representing a set $S$ is a Büchi automaton that accepts $L(S)$.

The sets that are RVA-representable in a base $r > 1$ are those definable in the first-order theory $\langle \mathbb{R}, \mathbb{Z}, +, \leq, X_r \rangle$, where $X_r$ is a base-dependent predicate.

Example:
Implementing RVA

Problem: Manipulating Büchi automata is inefficient, especially if automata need to be complemented.

Solution: Use only weak deterministic Büchi automata.

Theorem: The sets that are definable in \( \langle \mathbb{R}, \mathbb{Z}, +, \leq \rangle \) can be represented by weak deterministic Büchi automata (in any base \( r > 1 \))

Advantages:

- In practical applications, weak deterministic RVA are as easy to handle as NDDs;
- There exists a canonical form for weak deterministic RVA.
Number Automata in Verification

Problem: Computing the set of reachable configurations of a model with integer and/or real variables.

Solution:

- Represent the sets to be handled by NDDs or RVA;
- Use acceleration methods for computing infinite sets of reachable configurations in finite time.

Two classes of acceleration methods have been developed:

- **specific techniques**, based on properties of
  - the data domain under study
  - the operations performed on variables, and
- **generic techniques**.
Specific Acceleration Techniques

Idea: Compute at once the set of configurations that can be reached by iterating a control cycle.

Definition: Given a control cycle $\sigma$, the meta-transition associated to $\sigma$ is a transformation equivalent to

$$\sigma^* = Id \cup \sigma \cup \sigma^2 \cup \sigma^3 \cup \ldots$$

By adding meta-transitions to the transition relation of a model, one speeds up its state-space exploration.

Problems:

- Given $\sigma$, deciding whether $\sigma^*$ can be applied to represented sets;
- Computing a representation of $\sigma^*(S)$ given $\sigma$ and a representation of $S$. 

Meta-Transitions and NDDs

Theorem: Given a transformation

$$
\sigma : \mathbb{Z}^n \rightarrow \mathbb{Z}^n : \vec{x} \mapsto A\vec{x} + \vec{b},
$$

where $A \in \mathbb{Z}^{n \times n}, b \in \mathbb{Z}^n$, one can decide whether its closure preserves the NDD-representable nature of sets (in a given base, or in any base).

Principles:

- The decision procedure relies on the eigenvalues of $A$;
- The criterion can be decided using simple integer arithmetic operations;
- The proof is constructive and can be translated into an algorithm for computing $\sigma^*(S)$;
- The same criterion becomes sufficient for transformations guarded by a system of linear constraints.
Application: The Lift Controller

The system is composed of

- a control panel that prompts the user for a floor number,
- a motor controller that moves the car in the appropriate direction.

The number of floors $N$ can be

- specified in the model (the number of reachable configurations is then $O(N^2)$),
- made infinite (i.e., there is no top floor),
- turned into a parameter (the initial set of configurations then contains all the values of $N$ greater than 1).
Lift Controller: Simple-PROMELA Model

int c = 1, g = 1, a = 0, N = 10;

process motor {
    do
        :: go_up: atomic { a == 1 -> up: a = 0; c = c + 1 }
        :: go_down: atomic { a == 2 -> down: a = 0; c = c - 1 }
    od
}

process control {
    do
        :: too_low: atomic { c < g -> a = 1 } ; raise: a == 0
        :: too_high: atomic { c > g -> a = 2 } ; lower: a == 0
        :: atomic { c == g ->
            do
                :: low: g < N -> incr: g = g + 1
                :: high: g > 1 -> decr: g = g - 1
                :: break
            od
        
        :: assert c >= 1
        :: assert c <= N
    od;
    meta (low), incr, low;
    meta (high), decr, high
}

meta (control.too_low, motor.go_up),
    control.raise, motor.up, motor.go_up,
    control.too_low;

meta (control.too_high, motor.go_down),
    control.lower, motor.down, motor.go_down,
    control.too_high;

## Lift Controller: Runtime Statistics

<table>
<thead>
<tr>
<th>$N$</th>
<th>Rel.</th>
<th>Reach.</th>
<th>$t_{\text{tot}}$ (s)</th>
<th>Mem. (b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1751</td>
<td>123</td>
<td>8.51</td>
<td>344124</td>
</tr>
<tr>
<td>3</td>
<td>1751</td>
<td>175</td>
<td>9.25</td>
<td>397672</td>
</tr>
<tr>
<td>4</td>
<td>1751</td>
<td>337</td>
<td>14.17</td>
<td>730296</td>
</tr>
<tr>
<td>5</td>
<td>1751</td>
<td>349</td>
<td>13.28</td>
<td>707100</td>
</tr>
<tr>
<td>6</td>
<td>1751</td>
<td>402</td>
<td>17.53</td>
<td>888602</td>
</tr>
<tr>
<td>7</td>
<td>1751</td>
<td>355</td>
<td>14.36</td>
<td>746420</td>
</tr>
<tr>
<td>8</td>
<td>1751</td>
<td>601</td>
<td>23.69</td>
<td>1231180</td>
</tr>
<tr>
<td>9</td>
<td>1751</td>
<td>609</td>
<td>20.75</td>
<td>1138864</td>
</tr>
<tr>
<td>10</td>
<td>1751</td>
<td>664</td>
<td>26.82</td>
<td>1323048</td>
</tr>
<tr>
<td>100</td>
<td>1751</td>
<td>1306</td>
<td>69.78</td>
<td>2579876</td>
</tr>
<tr>
<td>1000</td>
<td>1751</td>
<td>1913</td>
<td>140.02</td>
<td>3701240</td>
</tr>
<tr>
<td>10000</td>
<td>1751</td>
<td>2684</td>
<td>220.02</td>
<td>5072576</td>
</tr>
<tr>
<td>100000</td>
<td>1751</td>
<td>3329</td>
<td>325.06</td>
<td>6283068</td>
</tr>
<tr>
<td>1000000</td>
<td>1751</td>
<td>3938</td>
<td>462.04</td>
<td>7399076</td>
</tr>
<tr>
<td>$\infty$</td>
<td>849</td>
<td>181</td>
<td>9.91</td>
<td>578332</td>
</tr>
<tr>
<td>any</td>
<td>2295</td>
<td>231</td>
<td>57.83</td>
<td>2403828</td>
</tr>
</tbody>
</table>
Lift Controller: Sample LASH run ($N = 1000000000$)

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), 649999997499999999 states.
- interm. result: 7808 NDD state(s), 799999995000000000 states.
- interm. result: 8655 NDD state(s), 899999999400000000 states.
- interm. result: 8658 NDD state(s), 949999993500000000 states.
- interm. result: 8663 NDD state(s), 999999993000000000 states.

Fixpoint reached in 11 step(s).

*** Program validated.

Runtime statistics:
- residual memory: 0 byte(s).
- max memory: 4344928 byte(s).
Meta-Transitions and RVA

Goal: Adding meta-transitions to models with a both continuous and discrete transition semantics (such as Hybrid Automata).

Definition: A Linear Hybrid Transformation (LHT) \((P, \vec{q})\) is a transformation of the form

\[
\theta : 2^{\mathbb{R}^n} \to 2^{\mathbb{R}^n} : S \mapsto \left\{ \vec{x}' \in \mathbb{R}^n \mid (\exists \vec{x} \in S) \left( P \begin{bmatrix} \vec{x} \\ \vec{x}' \end{bmatrix} \leq \vec{q} \right) \right\},
\]

with \(n > 0\), \(P \in \mathbb{Z}^{m \times 2n}\), \(\vec{q} \in \mathbb{Z}^m\) and \(m \geq 0\).

Properties:

- Any path of a linear hybrid automaton is labeled by a LHT;
- LHT that satisfy a periodicity criterion can be turned into meta-transitions.
Illustration (behavior of a periodic LHT):
Example: the Leaking Gas Burner

\[ x = 0 \land y = 0 \land z = 0 \]

\[ \begin{align*}
  \dot{x} &= 1 \\
  \dot{y} &= 1 \\
  \dot{z} &= 1 \\
  x &\leq 1
\end{align*} \]

\[ \begin{align*}
  x \geq 30 &\implies x := 0 \\
  \neg L &\implies \begin{align*}
    \dot{x} &= 1 \\
    \dot{y} &= 1 \\
    \dot{z} &= 0
  \end{align*}
\]
Conclusions

- Automata-based representations of arithmetic sets have nice properties;
- They are well suited for several data domains;
- The main limit is the number of variables;
- An implementation is available (LASH).