

# 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  $\mathbf{Z}^n$ , 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:

$$\begin{aligned} Enc_2(12) &= 0^+1100 \\ Enc_2(-7) &= 1^+001. \end{aligned}$$

- **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  $S \subseteq \mathbf{Z}^n$  is an automaton accepting all the encodings of all the elements in  $S$ .

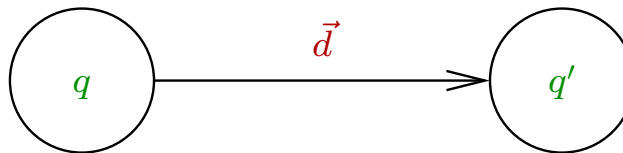
# From Linear Constraints to NDDs

An NDD representing the set

$$\{\vec{x} \in \mathbf{Z}^n \mid \vec{a} \cdot \vec{x} = b\}$$

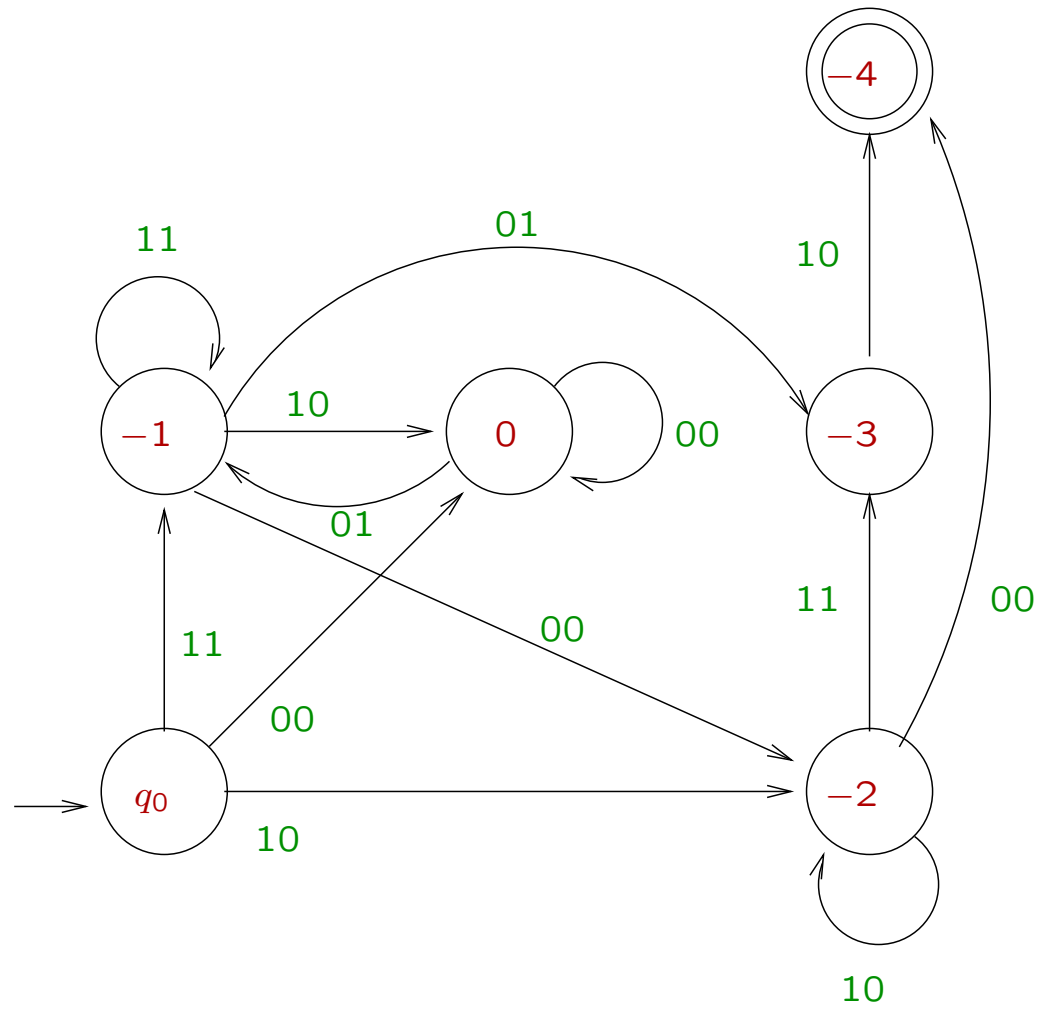
can be constructed by

- 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:  $0, 1, \dots, r - 1$ , and
- a separator  $\star$  between the integer and the fractional parts.

**Examples:**

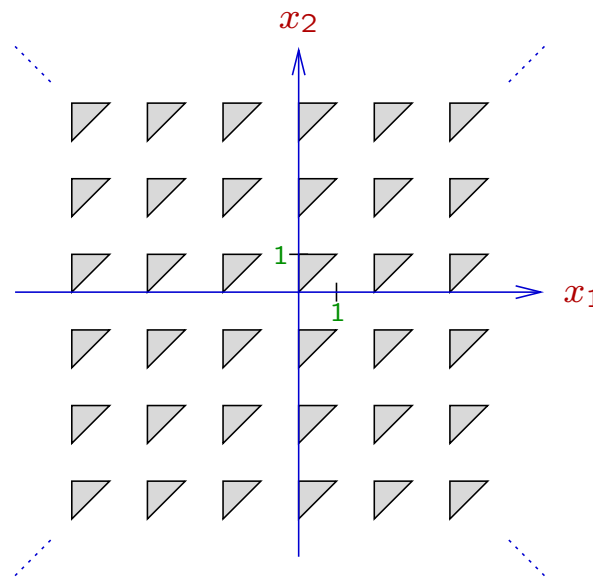
$$Enc_2(3.5) = 0^+11\star 1(0)^\omega \cup 0^+11\star 0(1)^\omega$$

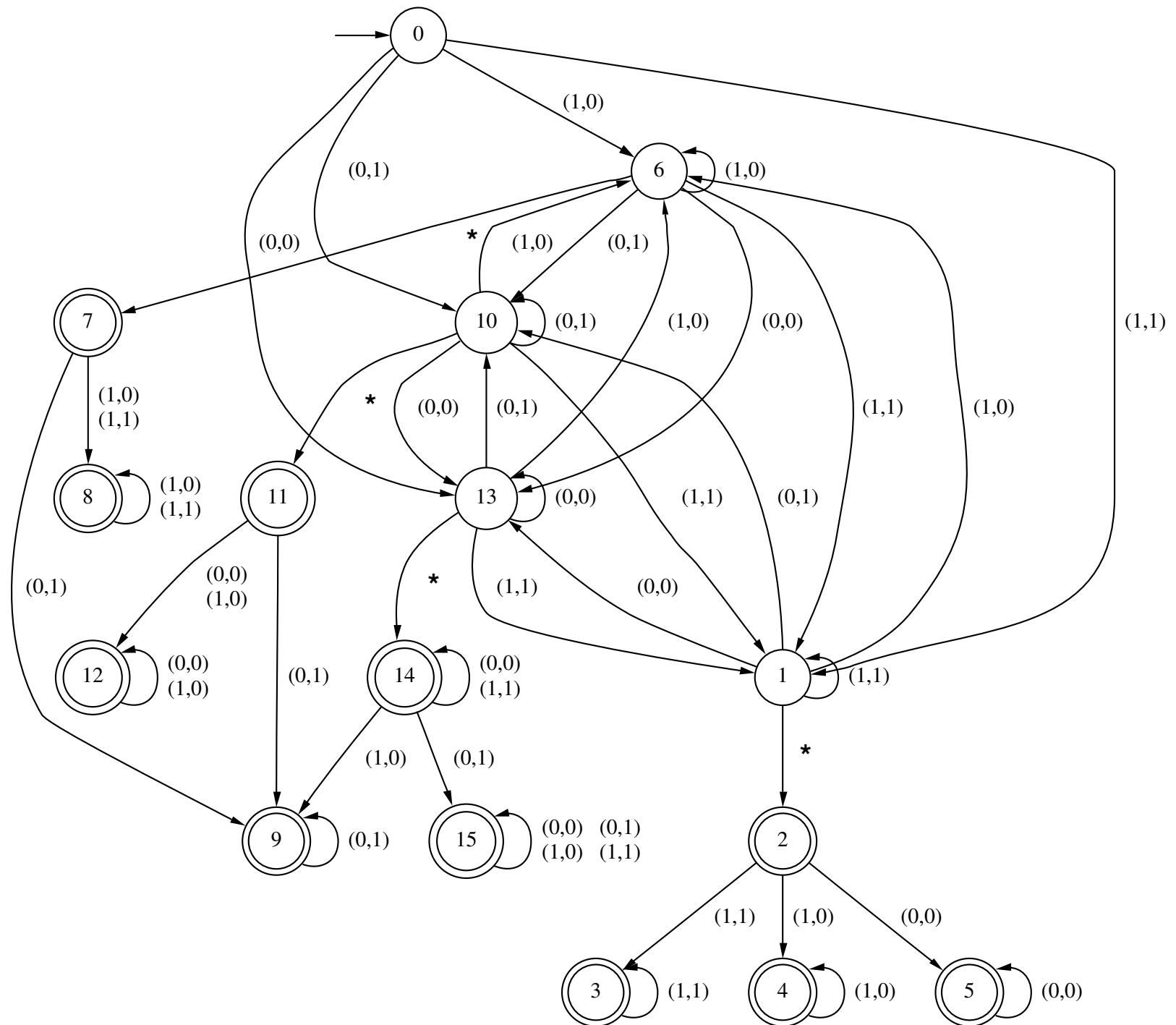
$$Enc_2(-4) = 1^+00\star (0)^\omega \cup 1^+011\star (1)^\omega.$$

# Real Vector Automata

- 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  $\langle \mathbf{R}, \mathbf{Z}, +, \leq, X_r \rangle$ , 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 can be represented by a weak deterministic Büchi automaton (in an arbitrary base  $r > 1$ ) are exactly those that are definable in  $\langle \mathbf{R}, \mathbf{Z}, +, \leq \rangle$ .

**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

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

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

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

where  $A \in \mathbf{Z}^{n \times n}$ ,  $b \in \mathbf{Z}^n$ , 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.

## 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

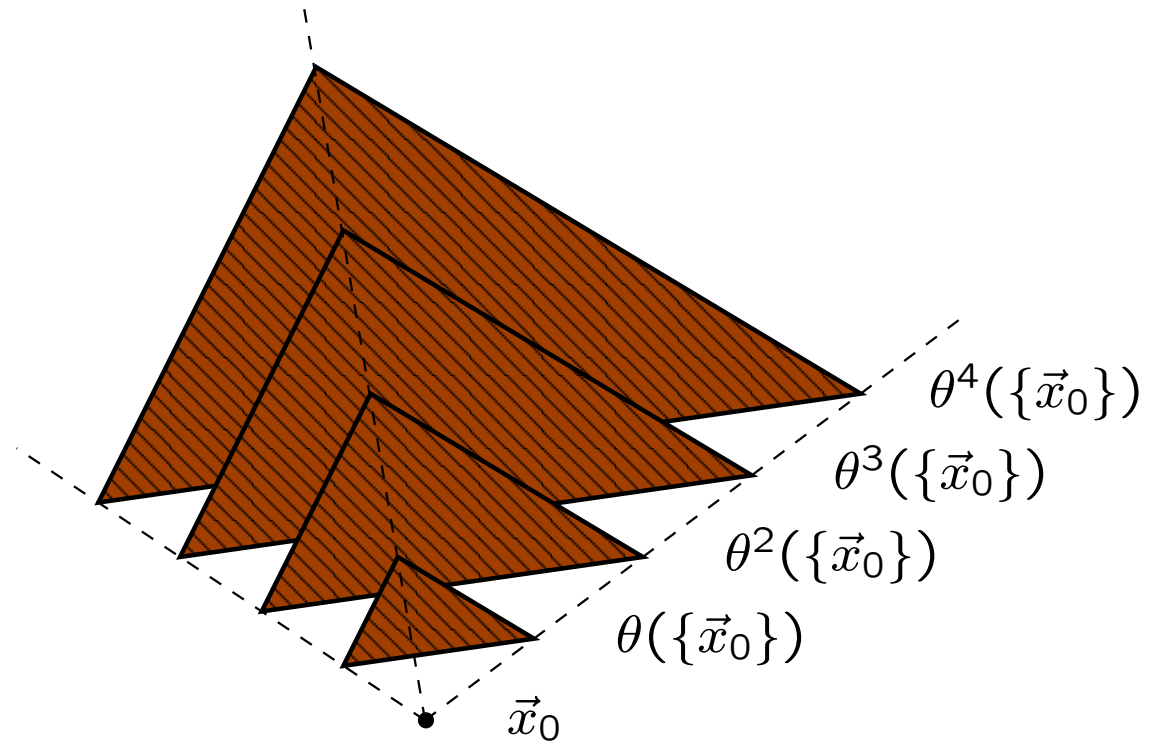
$$\theta : 2^{\mathbf{R}^n} \rightarrow 2^{\mathbf{R}^n} : S \mapsto \left\{ \vec{x}' \in \mathbf{R}^n \mid (\exists \vec{x} \in S) \left( P \begin{bmatrix} \vec{x} \\ \vec{x}' \end{bmatrix} \leq \vec{q} \right) \right\},$$

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

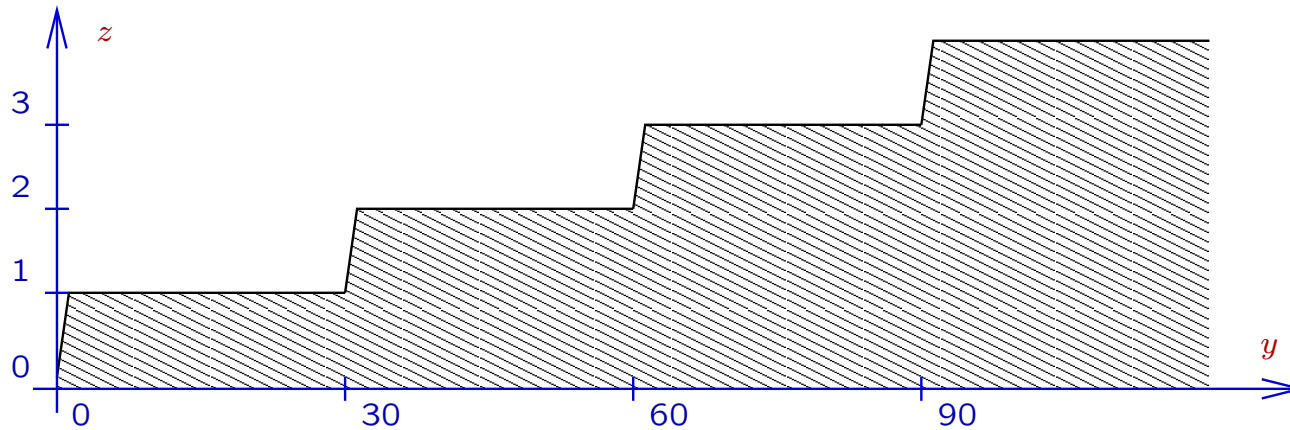
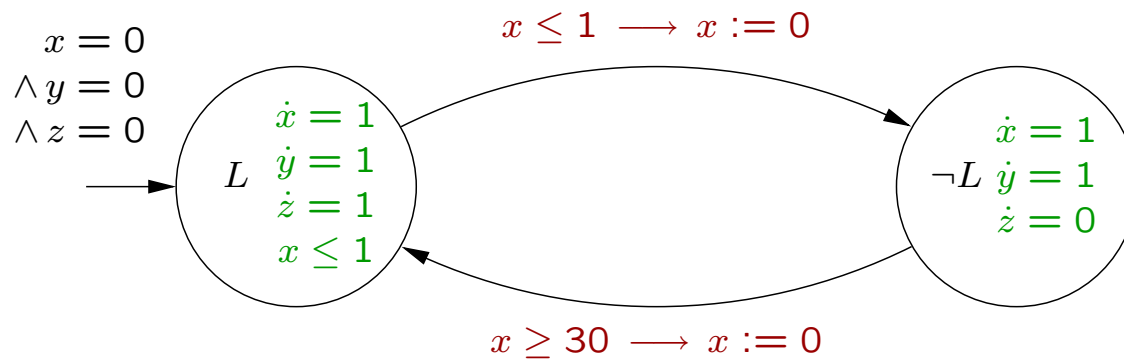
**Properties:**

- 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



# Generic Acceleration

## Principles:

- The transition relation is modeled by a **finite-state transducer**  $T$ ;
- Let  $T_0 = T \cup Id$ . The goal is to compute the limit of the sequence

$$T_0(A_0), (T_0)^2(A_0), (T_0)^3(A_0), \dots,$$

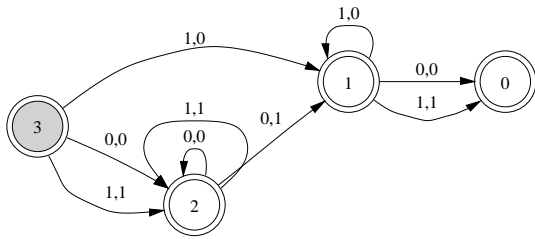
where  $A_0$  represents the set of initial configurations.

## Approach:

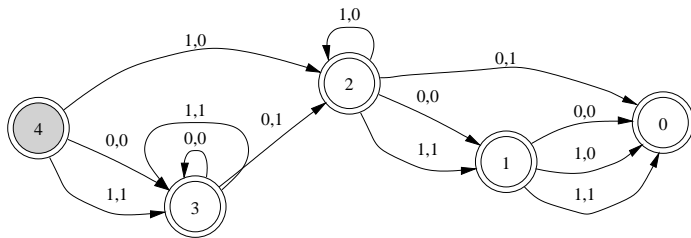
- One attempts to detect **similarities** between automata in the sequence;
- If these automata always differ by a **constant increment**, their infinite union can be captured by a finite-state structure;
- The safety and preciseness of the results can then be checked.

# Example of increments ( $x := x + 1$ ):

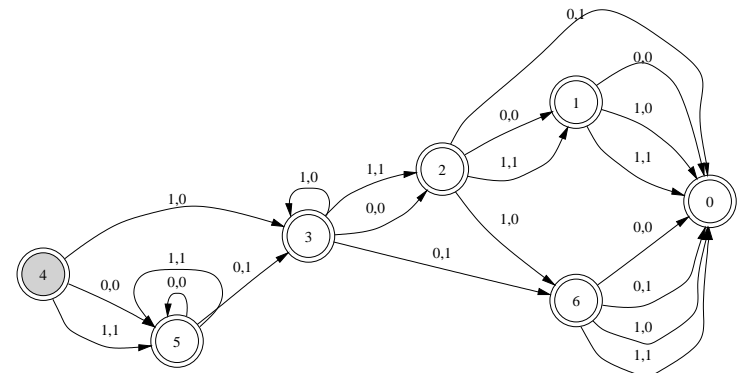
$T^2$ :



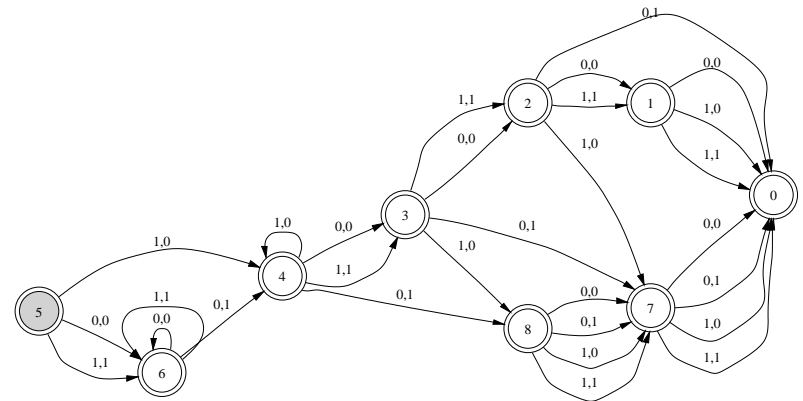
$T^4$ :



$T^8$ :



$T^{16}$ :



## 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**).