Counting the Solutions of Presburger Equations without Enumerating Them

Bernard Boigelot
Louis Latour

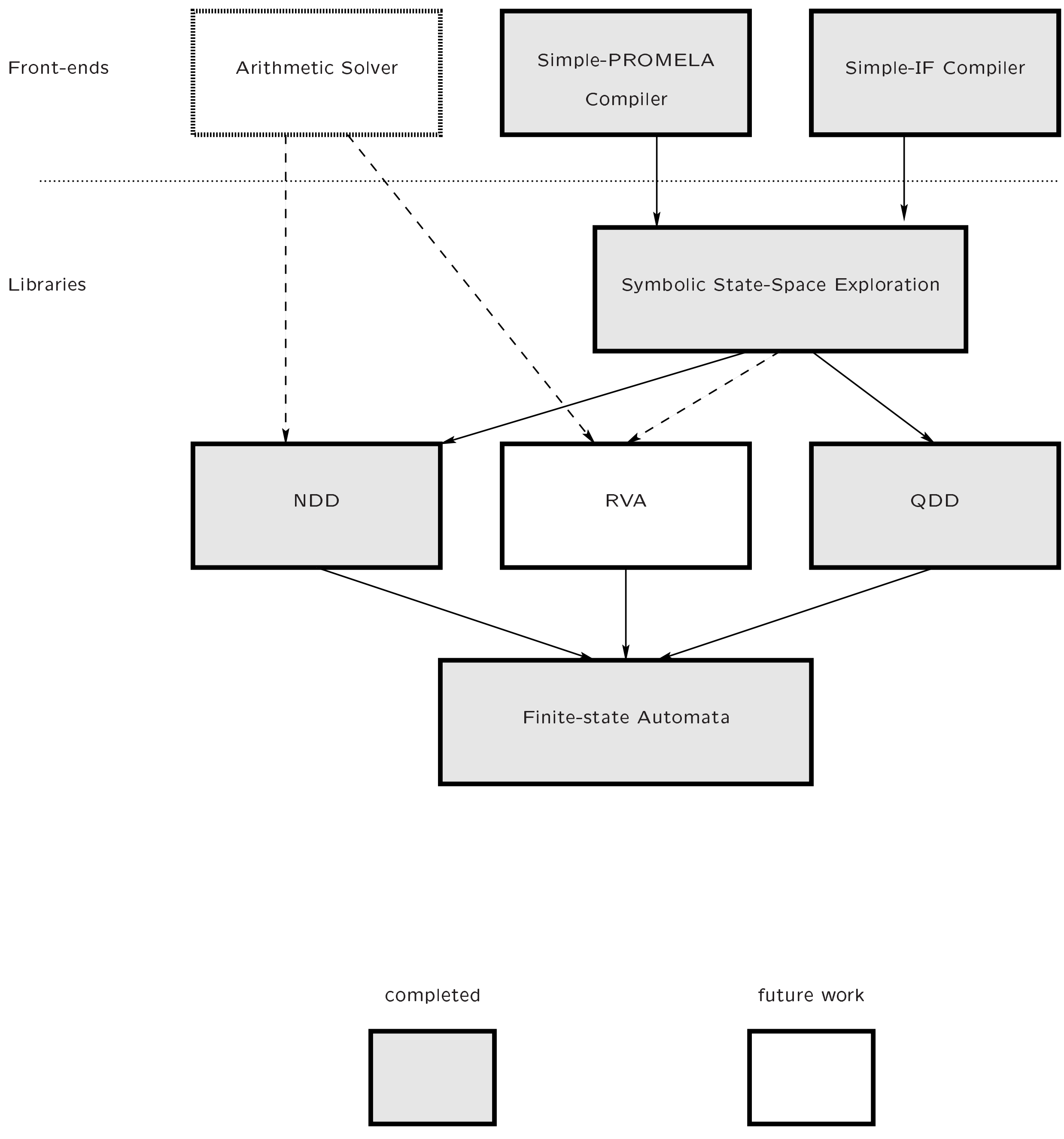
Université de Liège

Framework: The LASH Toolset

The LASH (*Liège Automata-based Symbolic Handler*) package is a collection of tools for

- building finite-state automata and performing basic automata operations,
- constructing finite-state representations of finite and infinite sets of data values (integer and real vectors, unbounded FIFO channels, . . .)
- performing various operations over represented sets;
- carrying out the symbolic state-space exploration of infinite-state systems modeled as communicating state machines;
- translating programs expressed in high-level languages (Promela, IF) into CSMs.

Current Status of LASH



Number Decision Diagrams

A *Number Decision Diagram (NDD)* is a finite-state representation of a subset of $\mathbf{Z}^n$.

Principles:

- Numbers are encoded in a base $r > 1$, most significant digit first;
- Negative values are encoded by their r's-complement;
- The length of the encodings is not fixed;
- A vector $\vec{v} = (v_1, v_2, \ldots, v_n)$ is encoded by words of the form

  $$d_{p-1,1}d_{p-1,2}\cdots d_{p-1,n} \quad d_{p-2,1}\cdots d_{p-2,n} \quad \cdots \quad d_{0,1}d_{0,2}\cdots d_{0,n},$$

  where for each i, $d_{p-1,i}d_{p-2,i}\cdots d_{0,i}$ is an encoding of v_i.

  The first n symbols of a vector encoding form its *sign header*.

Examples $(r = 2)$**:**

$$\begin{aligned} \textit{Enc}[4] &= \{0100, 00100, 000100, \ldots\}; \\ \textit{Enc}[-6] &= \{1010, 11010, 111010, \ldots\}; \\ \textit{Enc}[(4, -6)] &= \{01100100, 0101100100, \ldots\}. \end{aligned}$$

- An NDD representing a set $S \subseteq \mathbf{Z}^n$ is a finite-state automaton accepting the language formed by the encodings of the vectors belonging to S.

Theorems:

- A subset of $\mathbf{Z}^n$ is representable by an NDD in a base r if and only if it can be defined in the first-order theory $\langle \mathbf{Z}, +, \leq, V_r \rangle$, where for each $z \in \mathbf{Z}_0$, $V_r(z)$ is the highest natural power of r dividing z [Büchi60, Bruyère83].

- A subset of $\mathbf{Z}^n$ is representable by an NDD in every base $r > 1$ if and only if it can be defined in Presburger arithmetic, i.e., the first-order theory $\langle \mathbf{Z}, +, \leq \rangle$ [Cobham69, Semenov77].

Counting Solutions of Presburger Equations: What and Why?

The problem: Counting exactly how many vectors belong to a set represented by an NDD.

Motivations:

- Estimating the resources used by a program, or quantifying the performances of some systems;
- Studying the evolution of the number of reachable states of a system with respect to the value of some parameters;
- Detecting quickly inconsistencies between different releases of a program, without checking explicitly a large number of properties;
- ...

Counting Solutions of Presburger Equations: How?

Main Idea: The number of accepting paths of a deterministic automaton can be computed in linear time:

- If the automaton has reachable cycles, then the number of paths is infinite.
- Otherwise, the number of paths originating in a state s is given by

  $$N(s) = \sum_{d \in \Sigma, s' \in \mathsf{succ}(s,d)} N(s') + \mathsf{acc}(s),$$

  where acc(s) is equal to 1 if s is accepting, and to 0 otherwise.

  This value can easily be computed at all states by a depth-first strategy.

Problem: In NDDs, vectors and encodings are not linked by a one-to-one relationship.

Solution:

- Ensure that NDDs are always constructed so as to accept all the encodings of the vectors that they recognize, and
- Develop
  - a procedure for deciding whether the set represented by an NDD is finite or infinite, and
  - a transformation method for turning an NDD representing a finite set into an automaton accepting exactly one encoding of each recognized vector.

Projecting NDDs

The first requirement is problematic as far as the projection operation is concerned.

Definition:

$\exists_i S = \{(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n) \mid (v_1, \ldots, v_n) \in S\}$.

Problem:

Deleting the transition labels corresponding to the i-th vector component is not sufficient, because the resulting automaton does not always accept all encodings of vectors.

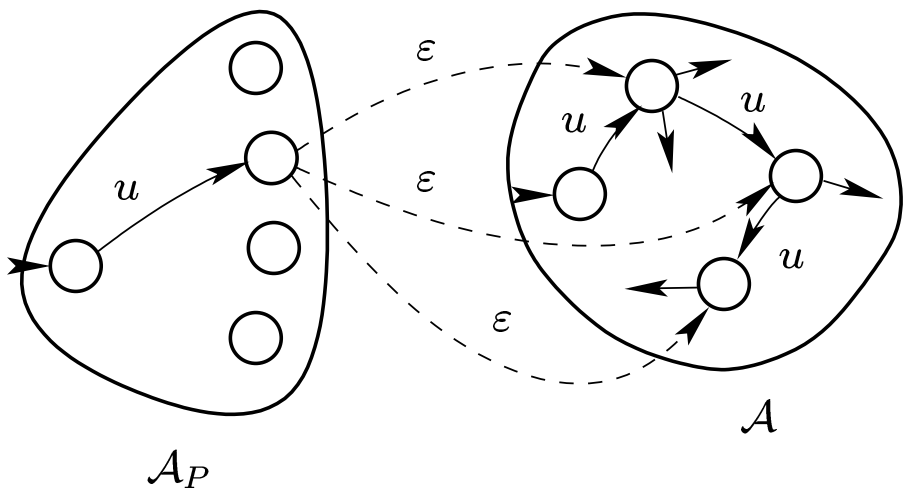
Example:

$$\begin{array}{rcl} S & = & \{(1, 5, 2)\} \\ \mathit{Enc}[S] & = & (000)^+\,010\,001\,110 \\ \mathit{Enc}[S]|_{1,3} & = & (00)^+\,00\,01\,10 \\ & \neq & \mathit{Enc}[\exists_2 S] = (00)^+\,01\,10 \end{array}$$

Projecting NDDs: Solution

Idea: For every sign header $u \in \{0, r-1\}^n$, if the automaton $\mathcal{A}$ accepts a word of the form $u^k w$, then it should also accept $u^i w$ for every $0 < i < k$.

This property can be enforced by building a *prefix automaton* $\mathcal{A}_P$ recognizing the sign headers, and then linking $\mathcal{A}_P$ to $\mathcal{A}$ by means of ε transitions.



Problem: This method is exponentially costly in n.

Improved solution: In the prefix automaton, avoid separating the sign headers that are not distinguished from each other by the main automaton.

This can be done by the following procedure:

1. The initial prefix automaton is obtained by intersecting $\mathcal{A}$ with an automaton accepting $\{0, r-1\}^n$;

2. During the exploration of $\mathcal{A}$, whenever one detects two sign headers that are undistinguished in $\mathcal{A}_P$ but that lead to different states in $\mathcal{A}$, one refines $\mathcal{A}_P$ so as to separate these headers.

Transforming NDDs

The problems:

- Determining whether the set S represented by an NDD $\mathcal{A}$ is finite or infinite;
- In the former case, transforming $\mathcal{A}$ into an automaton accepting exactly one encoding of each vector in S.

Solution: Assume that $\mathcal{A}$ is deterministic. The reachable cycles present in $\mathcal{A}$ can be of three types:

- *Sign loops*, that are only followed while reading sign headers;
- *Inflating loops*, that are never followed while reading sign headers;
- *Mixed loops*, that do not belong to the other two categories.

The set S is infinite if and only if $\mathcal{A}$ has inflating or mixed loops. Transforming $\mathcal{A}$ is equivalent to removing its sign loops.

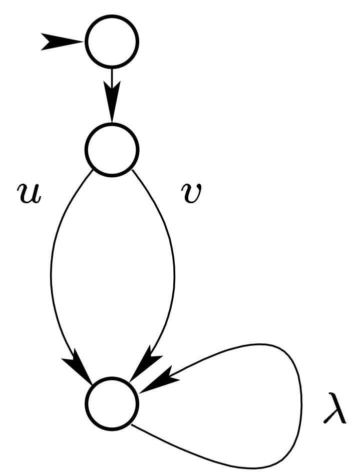
Classifying loops

Our approach relies on the following result.

Theorem: In a deterministic and minimal NDD $\mathcal{A}$, a loop λ is a sign loop if and only if it can only be reached by a single path (not containing any occurrence of that loop).

Proof:

- *If λ is reachable by at least two paths.* Then, one can visit λ while not reading a sign header.



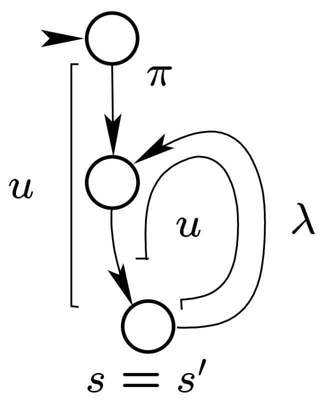
Indeed, λ cannot be labeled by a sign header consistent with both u and v.

- *If λ is reachable by only one path π.*

  Let $u \in \{0, r-1\}^n$ be the sign header read by $\pi\lambda$, and let s, s' be the states of $\mathcal{A}$ reached after reading u and uu.

  The languages accepted by $\mathcal{A}$ from the states s and s' must be identical, since for every $w \in \{0, \ldots, r-1\}$, $uw \in L(A)$ if and only if $uuw \in L(A)$.

  Since $\mathcal{A}$ is minimal, this implies $s = s'$.

  Therefore, λ is a sign loop.

Thanks to this theorem, loops can easily be classified as sign or non-sign loops by a simple depth-first search. Sign loops can be removed as soon as they are detected.

Summary

In order to count the number of solutions of a Presburger formula φ:

1. One builds an NDD $\mathcal{A}$ representing $S = \{\vec{v} \mid \varphi(v)\}$;
2. One determinizes and minimizes $\mathcal{A}$;
3. One checks that $\mathcal{A}$ has only sign loops, and removes all of them;
4. One counts the number of accepting paths of the resulting automaton.

Examples of use

LASH run on the lift example ($N = 1000000$):

```
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 : 148 NDD state(s).
Starting state-space exploration...
   intermediate result : 428 NDD state(s), 3 states.
   intermediate result : 694 NDD state(s), 1000003 states.
   intermediate result : 971 NDD state(s), 3999999 states.
   intermediate result : 1783 NDD state(s), 500005499997 states.
   intermediate result : 3014 NDD state(s), 1500006499995 states.
   intermediate result : 4186 NDD state(s), 3500004499994 states.
   intermediate result : 4603 NDD state(s), 6499997499999 states.
   intermediate result : 5101 NDD state(s), 7999995000000 states.
   intermediate result : 5658 NDD state(s), 8999994000000 states.
   intermediate result : 5661 NDD state(s), 9499993500000 states.
   intermediate result : 5666 NDD state(s), 9999993000000 states.
Fixpoint reached in 11 step(s).
*** Program validated.
Runtime statistics:
  residual memory : 0 byte(s).
  max memory      : 2862752 byte(s).
```

$N = 1000000000$:

```
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...
   intermediate result : 638 NDD state(s), 3 states.
   intermediate result : 1044 NDD state(s), 1000000003 states.
   intermediate result : 1461 NDD state(s), 3999999999 states.
   intermediate result : 2709 NDD state(s), 500000005499999997 states.
   intermediate result : 4596 NDD state(s), 1500000006499999995 states.
   intermediate result : 6409 NDD state(s), 3500000004499999994 states.
   intermediate result : 7020 NDD state(s), 6499999997499999999 states.
   intermediate result : 7808 NDD state(s), 7999999995000000000 states.
   intermediate result : 8655 NDD state(s), 8999999994000000000 states.
   intermediate result : 8658 NDD state(s), 9499999993500000000 states.
   intermediate 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).
```