Automata-based decision procedure for Presburger Logic

Deepak D'Souza

Department of Computer Science and Automation
Indian Institute of Science, Bangalore.

25 August 2016

Outline

1. Presburger Logic
2. Automata-based procedure
3. Decision Procedure
4. Summary

Presburger Logic

- First-Order logic of $(\mathbb{N}, <, +)$.
- Interpreted over $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.
- What you can say:

$$x + 2y < z + 1, \quad \exists x \varphi, \ \forall x \varphi, \ \neg, \wedge, \vee.$$

- Examples:
  1. $\forall x \forall y((x < y) \implies \exists z(x < z < y))$ (Also in FO(<)).
  2. Solutions to a system of linear inequalities: $\exists x \exists y(x + 2y \leq 1 \wedge x = y)$.
  3. "Every number is odd or even": $\forall x \exists y(x = 2y \vee x = 2y + 1)$.
- Studied by Mojzesz Presburger, who gave a sound and complete axiomatization, as well as a decision procedure for validity, circa 1929.

Problems to solve

Questions:

- Is there an algorithm to decide if a given Presburger logic sentence is true or not (validity problem)?
- Given a Presburger logic formula $\varphi(x, y)$, do there exist natural numbers x and y satisfying φ (satisfiability problem)?

Presburger Logic more formally

- Terms t are of the form:

$$0 \mid 1 \mid x \mid y \mid t + t$$

- Atomic formulas (f) are of the form:

$$t = t \mid t < t \mid t \leq t \mid \ldots$$

- General formulas (φ):

$$f \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \exists x \varphi \mid \forall x \varphi.$$

Overall idea

- Represent interpretation of variables as (rows of) binary strings

$$\begin{array}{ll} x & 001111 \\ y & 100011 \\ z & 011100 \end{array}$$

- Construct automata over such words, that accept all satisfying assignments of the variables, for atomic formulas.
- Use closure properties of automata to inductively construct automata for more complex formulas.

Representing numbers as binary strings

- Represent the number 3 by "011" or "0011" or "00011" etc.
- The automata will read the strings from right to left.
- Will read a tuple of bits: For example for the formula $x \leq 2y + 1$ it will read inputs from the alphabet

$$\{0, 1\}^2$$

which we represent as:

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

- Thus, automaton constructed for a given formula will accept the reverse of actual interpretations.

Automaton for $x + 2y - 3z = 1$

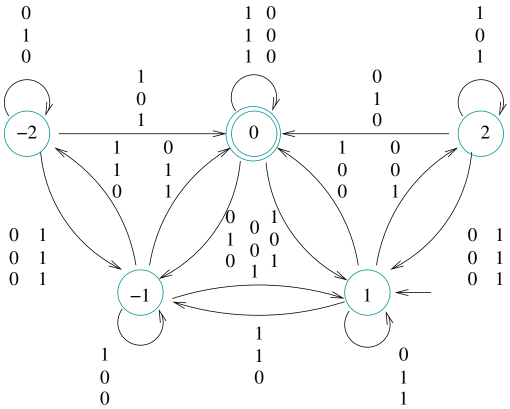
Accepting run on:

$$\begin{array}{ll} x\,(=0): & 000 \\ y\,(=2): & 010 \\ z\,(=1): & 001 \end{array}$$

$$\begin{array}{ll} x\,(=15): & 001111 \\ y\,(=35): & 100011 \\ z\,(=28): & 011100 \end{array}$$

but none on:

$$\begin{array}{ll} x\,(=1): & 001 \\ y\,(=2): & 010 \\ z\,(=1): & 001 \end{array}$$

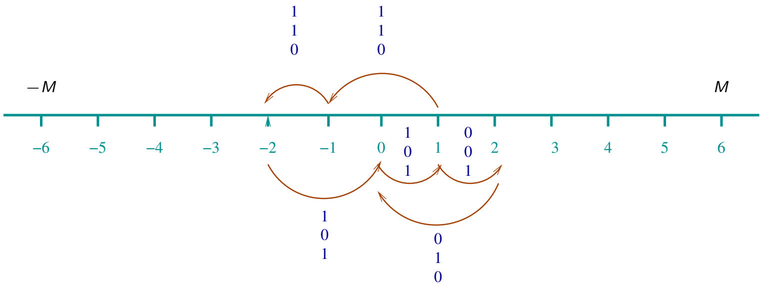


Construction for atomic formulas: Idea

Consider formula $x + 2y - 3z = 1$.

x	001111
y	100011
z	011100

Keep track of the weighted sum needed in the future to reach a weighted sum of b.



Construction for atomic formulas

Consider formula $x + 2y - 3z = 1$.

$$\begin{array}{ll} x & 00111\,1 \\ y & 10001\,1 \\ z & 01110\,0 \end{array}$$

- In general for formula $a_1x_1 + a_2x_2 + \cdots + a_nx_n = b$, with $a_i \in \mathbb{Z}$:
  - Begin with a state labelled b.
  - On reading bit vector $(\theta_1, \ldots, \theta_n)$
    - Check if $(a_1\theta_1 + \cdots + a_n\theta_n) = b \pmod 2$.
    - Move to state labelled $\frac{b-(a_1\theta_1+\cdots+a_n\theta_n)}{2}$.
  - Make state with label 0 as only final state.

Bounded state claim

Claim

The number of states is bounded by $2M + 1$ where

$$M = \max(|b|, |a_1| + \cdots + |a_n|).$$

The "remaining" weighted sum is always in the interval $[-M, M]$. Observe that the remaining weighted sum is an order less (the place value of bits goes down by a factor of 2).

- Handling inequalities:

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq b.$$

  - Replace by $\exists z(a_1x_1 + \cdots + a_nx_n + z = b)$.
  - Another approach:
    - Begin with initial state label b
    - From state c on input $(\theta_1, \ldots, \theta_n)$ go to state

$$\lfloor \frac{c - (a_1\theta_1 + \cdots + a_n\theta_n)}{2} \rfloor$$

    - and make all states with labels $c \geq 0$, final.
    - State labels are still in the range $[-M, M]$.
    - Correctness?
- Use closure under intersection (for $\wedge$), union (for $\vee$), complement (for $\neg$), and geometric projections (for $\exists$), to inductively construct automaton for φ.

Correctness of construction

- Argue the basic property that for any word $w \in (\{0,1\}^n)^+$, the automaton A_φ accepts w starting from state c iff the weighted sum of w is c. That is:

$$a_1 k_1 + \cdots + a_n k_n = c,$$

  where w represents the numbers $k_1, \ldots, k_n$.
- Proof by induction on length of w.

Deciding the logical questions

Given a Presburger logic formula φ we contruct the automaton $\mathcal{A}_\varphi$ as described, which accepts all the satisfying assignments that make φ true.

- If φ is a sentence (no free variables), then $\mathcal{A}_\varphi$ can be viewed as running on a dummy single-letter alphabet $\{a\}$. Then φ is valid iff $L(\mathcal{A}_\varphi) = a^+$. This can be checked algorithmically, by complementing $\mathcal{A}_\varphi$, intersecting with $\mathcal{A}_{a^+}$ and checking for emptiness.
- If φ has free variables, then φ is satisfiable iff $L(\mathcal{A}_\varphi)$ accepts a non-empty word. Again this can be algorithmically checked in linear time in size of $\mathcal{A}_\varphi$.

Summary

- Another application of automata-theory to solve a problem in logic.
- Automata approach gives us a convenient representation of the set of all satisfying assignments for a Presburger formula.
- Automata-based approach can be expensive (tower of exponentials), but more efficient decision procedures are known (triple exponential).