

# Automata-based decision procedure for Presburger Logic

Deepak D'Souza

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

02 September 2021

# Outline

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

# Presburger Logic

- First-Order logic of  $(\mathbb{N}, <, +)$ .
- Interpreted over  $\mathbb{N} = \{0, 1, 2, 3, \dots\}$ .
- 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  $\text{FO}(<)$ ).
  - 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 the following problems:

- Is a given Presburger logic sentence true or not (validity problem)?
- Given a Presburger logic formula  $\varphi(x, y)$ , do there exist natural numbers  $x$  and  $y$  satisfying  $\varphi$  (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:

$$t = t \mid t < t$$

- General formulas ( $\varphi$ ):

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

We denote by  $L(\varphi)$  the set of all interpretations for variables  $\mathbb{V}$  that satisfy  $\varphi$ .

# Overall idea

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

$x$  001111

$y$  100011

$z$  011100

- 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.

Exercise: Give an automaton for  $y = 2x + 1$



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

Accepting run on:

$x (= 0) :$  000

$y (= 2) :$  010

$z (= 1) :$  001

$x (= 15) :$  001111

$y (= 35) :$  100011

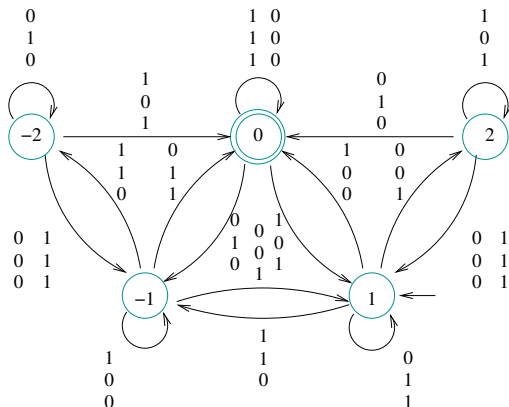
$z (= 28) :$  011100

but none on:

$x (= 1) :$  001

$y (= 2) :$  010

$z (= 1) :$  001

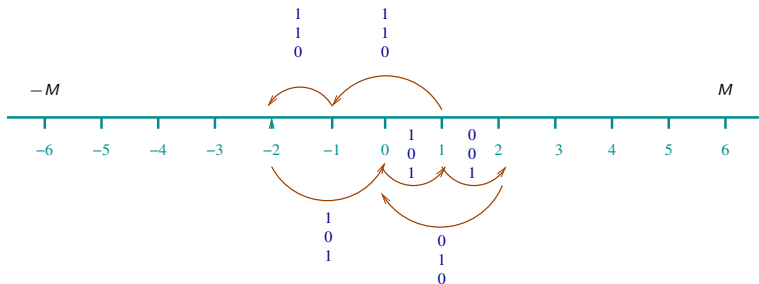


# Construction for atomic formulas: Idea

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

$x$	001111
$y$	100011
$z$	011100

Keep track of the **weighted** sum **needed** in the future to reach the original weighted sum of  $b$ .



# Construction for atomic formulas (=)

Consider formula  $\varphi : a_1x_1 + a_2x_2 + \cdots + a_nx_n = b$ , with  $a_i \in \mathbb{Z}$ :

Construct automaton  $\mathcal{A}_\varphi$  as follows:

- Begin with initial state labelled  $b$ .
- In general, if state is  $c$ , on reading bit vector  $(\theta_1, \dots, \theta_n)$ 
  - Check if  $(a_1\theta_1 + \cdots + a_n\theta_n) \equiv c \pmod{2}$ .
  - Move to state labelled  $\frac{c - (a_1\theta_1 + \cdots + a_n\theta_n)}{2}$ .
  - Else, move to “Error” state.
- Make state with label 0 the (only) final state.

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

$x$	001111
$y$	100011
$z$	011100

# Exercise: Construct the automaton for $y = 2x + 1$

Using the algorithm.

# Bounded state claim

## Claim

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

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

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).

# Weighted Sum

- Fix an atomic formula  $\varphi$ :  $a_1x_1 + \cdots + a_nx_n = b$
- Define **weighted sum** of a string  $w = d_k \cdots d_0 \in (\{0, 1\}^n)^*$ :

$$wsum(w) = a_1(k_1) + \cdots + a_n(k_n),$$

where  $k_1, \dots, k_n$  are the numbers represented by  $w$ .

- Thus, if  $w \neq \epsilon$  with  $|w| = k + 1$ , then

$$\begin{aligned} wsum(w) &= a_1(2^k d_k(1) + \cdots + 2^0 d_0(1)) + \\ &\quad \dots \\ &\quad a_n(2^k d_k(n) + \cdots + 2^0 d_0(n)) \end{aligned}$$

If  $w = \epsilon$ , then  $wsum(w)$  is defined to be 0.

## Claim

If  $w = v \cdot u$  then  $wsum(w) = 2^{|u|} \cdot wsum(v) + wsum(u)$ .

# Correctness of construction for atomic formulas with =

## Claim

After reading  $u \in (\{0, 1\}^k)^*$  the automaton  $\mathcal{A}_\varphi$  will be in state

$$\begin{cases} c \text{ such that } c \cdot 2^{|u|} + wsum(u) = b & \text{if } wsum(u) \equiv b \pmod{2^{|u|}} \\ \text{Error} & \text{otherwise} \end{cases}$$

Proof: By induction on  $|u|$ .

- Base case:  $u = \epsilon$
- Induction step:  $u = d \cdot w$

# Construction for $\leq$

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

- One approach:
  - Begin with initial state label  $b$
  - From state  $c$  on input  $(\theta_1, \dots, \theta_n)$  go to state

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

- and make all states with labels  $c \geq 0$ , final.
  - State labels are still in the range  $[-M, M]$ .
  - Note that remaining weighted sum **is an integer**.
- Another approach: Replace by  $\exists z(a_1x_1 + \cdots + a_nx_n + z = b)$ .



# Construction for general formulas

- We use models in  $(\{a\} \times \{0, 1\}^n)^+$  ( $0 \leq n$ ). Thus models are **non-empty** words of tuples of the form  $(a, 0, 1, \dots, 0)$ . All operations (including complementation) is wrt this universe of models.
- For a given formula  $\varphi$ , we define a relation  $R_\varphi$  that relates valuations for variables (say  $\mathbb{V}$ ) with models  $w$  of the form above.
- Let  $A_\varphi$  denote the alphabet  $\{a\} \times \{0, 1\}^{|FV(\varphi)|}$ .
- Then  $(\mathbb{V}, w) \in R_\varphi$  iff  $w \in A_\varphi^+$  and for each  $x \in FV(\varphi)$ ,  $\mathbb{V}(x) = (w(x))_2$ .
- We use “ $(w(x))_2$ ” to denote the value of the binary string corresponding to the row for  $x$  in  $w$ .
- Note that  $R_\varphi$  is a many-to-many relation.

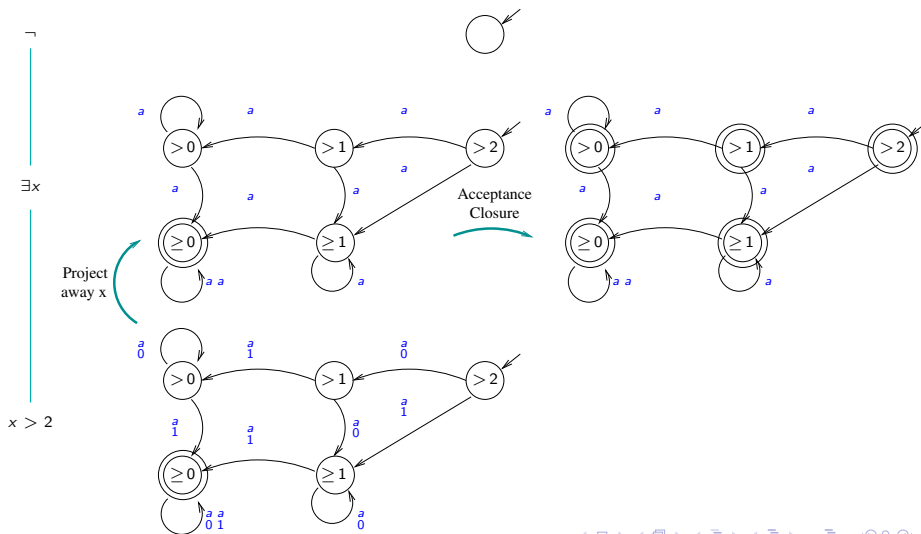
# Construction for general formulas

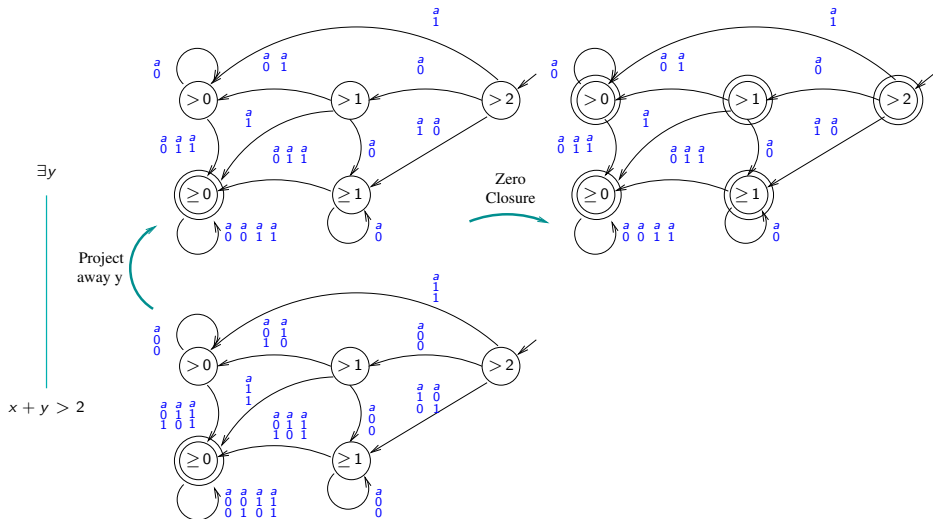
## Claim

For any Presburger logic formula  $\varphi$  we can construct an automaton  $\mathcal{A}_\varphi$  that accepts precisely the set  $R_\varphi(L(\varphi))$ .

We construct  $\mathcal{A}_\varphi$  inductively:

- For atomic formulas, construct as described earlier.
- For  $\psi_1 \vee \psi_2$ , we add rows for new variables (for example  $x$  in  $FV(\psi_2) - FV(\psi_1)$ ) in the automata  $\mathcal{A}_{\psi_1}$  and  $\mathcal{A}_{\psi_2}$ , and then “union” them.
- For  $\neg\psi$ , we construct an automaton for  $A_\psi^+ - L(\mathcal{A}_\psi)$ .
- For  $\exists x\psi$ , we do the following:
  - Project out the row for  $x$  in  $\mathcal{A}_\varphi$
  - If no free vars in  $\varphi$ , then take **acceptance-closure**.
  - Else (if there are free vars in  $\varphi$ ), take **zero-closure**.

Illustrating acceptance-closure:  $\neg \exists x (x > 2)$ 

Illustrating zero-closure:  $\exists y(x + y > 2)$ 

# Deciding the logical questions

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

- If  $\varphi$  is a sentence (no free variables), then  $\mathcal{A}_\varphi$  runs on the single-letter alphabet  $\{a\}$ . Then  $\varphi$  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  $\varphi$  has free variables, then  $\varphi$  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).