Presburger Arithmetic

So far, we started from automata and "bent" logics so that it could represent the same objects as automata, namely sets of words.
In this part of the lecture, we take the opposite route:
we start from a logics of independent interest, representing sets of natural numbers, and we "bend" automata so that they can be used to represent the same objects.
In other words, we use automata as DATA STRUCTURES to represent, compute and manipulate sets of models of formulas over $(\mathbb{N},+)$.

Def Presburger Arithmetic (PA) is the FO logics over $\mathbb{N}$ with addition.
Fix the signature $(\underbrace{\{0, 1, +\}}_{\text{Functional}}, \underbrace{\{<, =, \{\equiv_k\}_{k>1}\}}_{\text{Predicate}})$ ($\equiv_k$: congruence modulo k)

The syntax of PA formulas follows the grammar

$$t ::= 0 \mid 1 \mid t+t \qquad \text{(terms)}$$
$$\varphi ::= t_1 = t_2 \mid t_1 < t_2 \mid t_1 \equiv_k t_2 \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \exists x:\varphi \qquad \text{(formulas)}$$

Formulas are interpreted over the fixed structure $(\mathbb{N}, 0^{\mathbb{N}}, 1^{\mathbb{N}}, +^{\mathbb{N}}, <^{\mathbb{N}}, =^{\mathbb{N}}, \equiv_k^{\mathbb{N}})$

Interpretations $I : V \to \mathbb{N}$ map free variables to naturals and are extended to terms as expected: $I(0) := 0^{\mathbb{N}}$ $I(1) := 1^{\mathbb{N}}$ $I(t+t') := I(t) +^{\mathbb{N}} I(t')$
The satisfaction relation $I \models \varphi$ is defined as follows

$I \models t_1 \sim t_2$ if $I(t_1) \sim^{\mathbb{N}} I(t_2)$ (for $\sim \in \{=, <, \equiv_k\}$)
$I \models \neg\varphi$ if $I \not\models \varphi$
$I \models \varphi_1 \wedge \varphi_2$ if $I \models \varphi_1$ and $I \models \varphi_2$
$I \models \exists x:\varphi$ if there is $n \in \mathbb{N}$ such that $I[n/x] \models \varphi$

As usual we assume variables are ordered, so we can view a vector $\vec{v} = \begin{pmatrix} v_1 \\ \vdots \\ v_m \end{pmatrix} \in \mathbb{N}^m$ as an interpretation $[{}^{v_i}/_{x_i}]$ and write $\vec{v} \models \varphi$ accordingly.

When $\vec{v} \models \varphi$ we say $\vec{v}$ is a <u>model</u>, or a <u>solution</u> of φ.

We say φ is satisfiable if it has at least one solution (a closed formula can have the empty vector as a solution).

The <u>Solution Space</u> of a formula φ is the set

$$Sol(\varphi) := \{ \vec{v} \in \mathbb{N}^m \mid \vec{v} \models \varphi \}$$

A set $S \subseteq \mathbb{N}^m$ is <u>Presburger-definable</u> if there is a formula $\varphi \in PA$ such that $S = Sol(\varphi)$.

<u>Abbreviations</u> For each $k \in \mathbb{N}$ we use the abbreviations

$$k := \underbrace{1 + \ldots + 1}_{k \text{ times}} \qquad kt := \underbrace{t + \ldots + t}_{k \text{ times}} \qquad t_1 \geq t_2 := \neg (t_1 < t_2)$$

→ <u>A note on expressivity</u>: not all the elements of the signature are strictly needed

$$y = 0 \equiv \forall x : \neg (x < y)$$

$$y = 1 \equiv \forall x : (x < y \rightarrow x = 0) \wedge y > 0$$

so to use the constant 0 (or 1) in φ use y instead of the const. and wrap $\exists y : y = 0 \wedge \varphi[{}^{0}/_{y}]$ (as above) (or $y = 1$) (or $[{}^{1}/_{y}]$)

Moreover

$$(t_1 = t_2) \equiv \neg (t_1 < t_2) \wedge \neg (t_2 < t_1)$$

$$(t_1 \equiv_k t_2) \equiv \exists q : t_1 = kq + t_2 \wedge t_2 < k$$

So $PA[<, =, \equiv_k]$ is equiexpressive to $PA[<]$ (with or without 0 and/or 1)

<u>However</u>: our goal here is to show decidability of satisfiability for PA and we know that the difficult case is quantification. Hence, It is undesirable to use quantification to represent constants: to decide satisfiability of $2 = 1 + 1$ we should not need to reason about $\exists x!$ Therefore from now on we will consider $PA[<]$ with 0 and 1 as primitives, even if not strictly necessary.

Goal: Can we use automata to represent $Sol(\varphi)$?

First we have to find a way to see $Sol(\varphi)$ as a set of words.
But we write numbers as words all the time! Take binary representation.

Def Let $\mathbb{B} = \{0,1\}$. The function $LSBF : \mathbb{N} \to \mathcal{P}(\mathbb{B}^*)$ maps a number n to its binary representation (with least significant bit as the leftmost symbol) optionally padded with arbitrarily many 0s: $LSBF(n) = bin(n) \cdot 0^*$

($LSBF$: Least Significant Bit First; $\mathcal{P}(\mathbb{B}^*)$: set of sequences of bits)

We extend $LSBF$ to vectors by using words over vectors of bits:

$$LSBF : \mathbb{N}^n \to \mathcal{P}((\mathbb{B}^n)^*)$$

$$LSBF\begin{pmatrix} m_1 \\ \vdots \\ m_n \end{pmatrix} = \left\{ \begin{pmatrix} b_1^1 \\ b_n^1 \end{pmatrix} \begin{pmatrix} b_1^2 \\ b_n^2 \end{pmatrix} \cdots \begin{pmatrix} b_1^k \\ b_n^k \end{pmatrix} \;\middle|\; \forall i \;\; b_i^1 \ldots b_i^k \in LSBF(m_i) \right\}$$

The **language** of $\varphi \in PA$ is $\mathcal{L}(\varphi) := \bigcup_{\vec{v} \in Sol(\varphi)} LSBF(\vec{v})$

Example $LSBF\left(\begin{pmatrix} 2 \\ 5 \\ 0 \end{pmatrix}\right) = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}^*$

since $LSBF(2) = 010^*$
$LSBF(5) = 1010^*$
$LSBF(0) = 0^*$

Note
To treat the case $n=0$ uniformly we set $LSBF(()) = \{\varepsilon\}$ (the empty vector)
so that $\mathcal{L}(\varphi) = \emptyset$ iff φ is unsat
also when φ is closed

Theorem Let $\varphi \in PA[<]$. We can effectively construct a DFA A_φ over $\mathbb{B}^n$ with $\mathcal{L}(A_\varphi) = \mathcal{L}(\varphi)$.

As a direct consequence we can decide satisfiability of a PA formula by emptiness check on A_φ.

Proof We construct a suitable A_φ by induction over the structure of φ.
Assume we can construct A_ψ for some formulas ψ, ψ'. It can be readily checked that $\mathcal{L}(\neg\psi) = \overline{\mathcal{L}(A_\psi)}$, $\mathcal{L}(\psi \wedge \psi') = \mathcal{L}(A_\psi \cap A_{\psi'})$, $\mathcal{L}(\exists x \psi) = \pi_x(A_\psi)$ (projection)
This takes care of the induction step, it remains to show a suitable DFA exists for the base cases $t_1 < t_2$. To handle these cases uniformly we introduce the notation $\vec{a}\,\vec{x} \leqslant b$ with $\vec{a} = (a_1 \dots a_n)$ and $a_1, \dots, a_n, b \in \mathbb{Z}$ to mean $a_1x_1 + a_2x_2 + \dots + a_nx_n \leqslant b$. Each formula $t_1 < t_2$ can be rewritten as $\vec{a}\,\vec{x} \leqslant b$ for some $\vec{a}$ and b. Example:

$$\underbrace{2y + z}_{t_1} < \underbrace{5 + x}_{t_2} \equiv \underbrace{(-1\ 2\ 1)}_{\vec{a}} \underbrace{\begin{pmatrix} x \\ y \\ z \end{pmatrix}}_{\vec{x}} \leqslant \underbrace{4}_{b}$$

Key idea for constructing DFA $A_{\vec{a}b}$ over $\mathbb{B}^n$ such that $\mathcal{L}(A_{\vec{a}b}) = \mathcal{L}(\vec{a}\vec{x} \leqslant b)$
The states of $A_{\vec{a}b}$ are integers $q \in \mathbb{Z}$ and we design the transitions so that

$\circledast$ $q \in \mathbb{Z}$ accepts $w \in (\mathbb{B}^n)^*$ in $A_{\vec{a}b}$ iff w encodes $\vec{c} \in \mathbb{N}^n$ with $\vec{a}\,\vec{c} \leqslant q$

Now, if $\circledast$ holds, by setting b as the initial state we accept the words in $\mathcal{L}(\vec{a}\vec{x} \leqslant b)$ exactly, as desired.
How do we have to define transitions so that $\circledast$ holds?
Observe that if $q \xrightarrow{\vec{\beta}} q'$ with $\vec{\beta} \in \mathbb{B}^n$, q is recognising # numbers the first digit of which are the components of β:

$\vec{\beta}\,w'$ is accepted from q iff w' is accepted from q'

but if w' encodes $\vec{c}' \in \mathbb{N}^n$, the word $\vec{\beta}\,w'$ encodes $\vec{c} = 2\vec{c}' + \vec{\beta}$ (remember: least significant is the left most)

Hence $\vec{c}'$ is accepted from q' iff $2\vec{c}' + \vec{\beta}$ is accepted from q
so to satisfy $\circledast$ we need

$$\vec{a}\,\vec{c}' \leqslant q' \text{ iff } \vec{a}(2\vec{c}' + \vec{\beta}) \leqslant q$$

which we can use to calculate q':

$$\vec{a}(2\vec{c}' + \vec{\beta}) \leq q$$
$$2\vec{a}\vec{c}' + \vec{a}\vec{\beta} \leq q$$
$$\vec{a}\vec{c}' \leq \tfrac{1}{2}(q - \vec{a}\vec{\beta})$$
$$\vec{a}\vec{c}' \leq \underbrace{\left\lfloor \tfrac{1}{2}(q - \vec{a}\vec{\beta}) \right\rfloor}_{\text{the } q' \text{ satisfying } \circledast} \quad \text{(because } \vec{a}\vec{c}' \text{ are integers!)}$$

So we add to $A_{\vec{a}\vec{x}}$ all the transitions with

$$q \xrightarrow{\vec{\beta}} \left\lfloor \tfrac{1}{2}(q - \vec{a}\vec{\beta}) \right\rfloor \quad \text{(for each } \vec{\beta} \in \mathbb{B}^n\text{)}$$

Note that these transitions make $A_{\vec{a}b}$ deterministic!

To keep $A_{\vec{a}\vec{x}}$ finite state, we only add the transitions actually reachable from b, our initial state. (we will prove this later).

The final states of $A_{\vec{a}\vec{b}}$ are by definition exactly the ones accepting the empty word, which encodes the vector with 0 in all components ($\vec{0}$).

So q is final in $A_{\vec{a}b}$ iff q accepts ε iff q accepts encodings of $\vec{0}$ iff (by $\circledast$) $\vec{a}\vec{0} \leq q$. So we set to final all states q in $A_{\vec{a}b}$ with $q \geq 0$.

Overall, the algorithm to construct $A_{\vec{a}b}$ works by constructing a sequence of automata

$$A^0_{\vec{a}b} := (\{b\}, b, \emptyset, \emptyset)$$
$$A^{k+1}_{\vec{a}b} := (Q^{k+1}_{\vec{a}b}, b, \delta^{k+1}_{\vec{a}b}, F^{k+1}_{\vec{a}b})$$

where $\delta^{k+1}_{\vec{a}b} := \delta^k_{\vec{a}b} \cup \{ q \xrightarrow{\vec{\beta}} \lfloor \tfrac{1}{2}(q - \vec{a}\beta) \rfloor \mid q \in Q^k_{\vec{a}b}, \vec{\beta} \in \mathbb{B}^n \}$

$Q^{k+1}_{\vec{a}b} :=$ all states mentioned in $\delta^{k+1}_{\vec{a}b}$

$F^{k+1}_{\vec{a}b} := \{ q \in Q^{k+1}_{\vec{a}b} \mid q \geq 0 \}$

Until for some $h \in \mathbb{N}$ we have $A^h_{\vec{a}b} = A^{h+1}_{\vec{a}b} =: A_{\vec{a}b}$

Correctness It suffices to show that for any $w \in (\mathbb{B}^n)^*$ (by induction on length of w)

q accepts w in some $A^k_{\vec{a}b}$ iff w encodes $\vec{c}$ with $\vec{a}\vec{c} \leq q$

Termination We need to show that there is a $h \in \mathbb{N}$ so that $A^h_{\vec{a}b} = A^{h+1}_{\vec{a}b}$, i.e. that the algorithm terminates and indeed $A_{\vec{a}b}$ has finitely many states.

We prove this by proving the invariant:

$$\forall k \in \mathbb{N}. \forall q \in Q^k_{\vec{a}b} : \quad -|b| - s \leq q \leq |b| + s$$

(absolute value!)

where $s = \sum_{i=1}^{n} |a_i|$. By showing this is true we show that any q that is added in the algorithm is an integer within this finite range.

Proof by induction on k

(k=0) $Q^0_{\vec{a}b} = \{b\}$ ✓

(k+1) Assume $Q^k_{\vec{a}b}$ satisfies the hypothesis.

Take $q' \in Q^{k+1}_{\vec{a}b} \setminus Q^k_{\vec{a}b}$ (the others sat. condition trivially)

we have $\exists q \in Q^k_{\vec{a}b}$ s.t. $q' = \lfloor \frac{1}{2}(q - \vec{a}\vec{\beta}) \rfloor$ for some $\vec{\beta} \in \mathbb{B}^n$.

By induction hypothesis

$$-|b| - s \leq q \leq |b| + s$$

and hence

$$-|b| - s \leq \left\lfloor \frac{-|b| - 2s}{2} \right\rfloor \leq \left\lfloor \frac{1}{2}(-|b| - s - \vec{a}\vec{\beta}) \right\rfloor \leq \underbrace{\left\lfloor \frac{1}{2}(q - \vec{a}\vec{\beta}) \right\rfloor}_{q'} \leq \left\lfloor \frac{1}{2}(|b| + s - \vec{a}\vec{\beta}) \right\rfloor \leq \left\lfloor \frac{|b| + 2s}{2} \right\rfloor \leq |b| + s$$

($-s \leq \vec{a}\vec{\beta} \leq s$ by def of s)

□

Example Consider $\varphi = 2x - y \leq 2$. We have $\varphi \equiv \underbrace{(2\ -1)}_{\vec{a}} \underbrace{\begin{pmatrix} x \\ y \end{pmatrix}}_{\vec{x}} \leq \underbrace{2}_{b}$

$$A_{\vec{a}b} = (Q_{\vec{a}b}, b, \delta_{\vec{a}b}, \{q \in Q_{\vec{a}b} \mid q \geq 0\})$$

as depicted in the next page

Compute a few transitions:

$$2 \xrightarrow{\binom{0}{0}} \left\lfloor \tfrac{1}{2}\left(2 - (2\ -1)\binom{0}{0}\right)\right\rfloor = 1$$

$$2 \xrightarrow{\binom{0}{1}} \left\lfloor \tfrac{1}{2}\left(2 - (2\ -1)\binom{0}{1}\right)\right\rfloor = 1$$

$$2 \xrightarrow{\binom{1}{0}} \left\lfloor \tfrac{1}{2}\left(2 - (2\ -1)\binom{1}{0}\right)\right\rfloor = 0$$

$$2 \xrightarrow{\binom{1}{1}} \left\lfloor \tfrac{1}{2}\left(2 - (2\ -1)\binom{1}{1}\right)\right\rfloor = 0$$

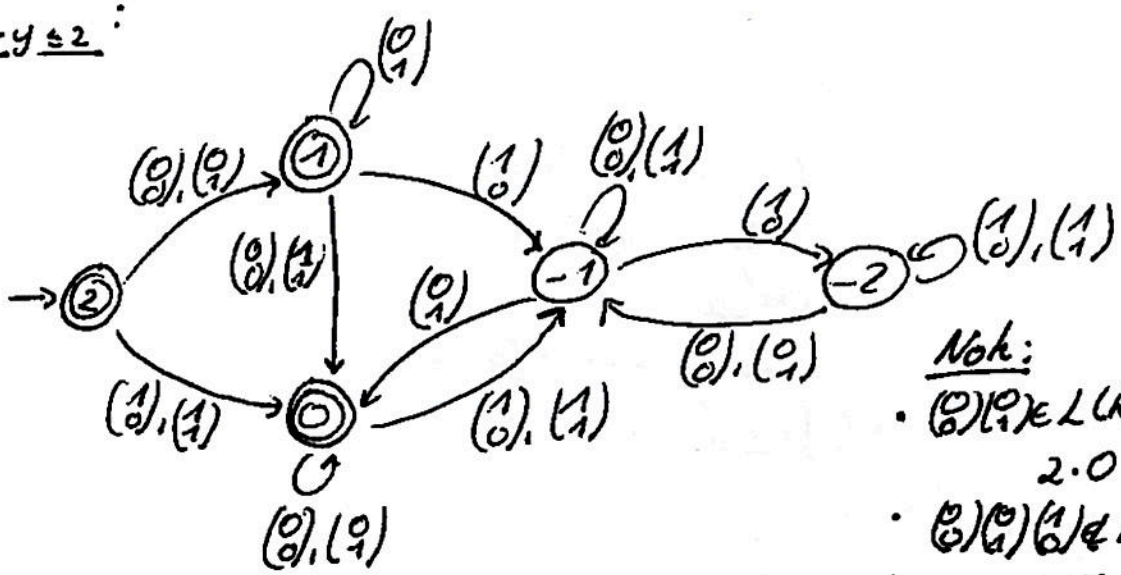
$$1 \xrightarrow{\binom{0}{0}} \left\lfloor \tfrac{1}{2}\left(1 - (2\ -1)\binom{0}{0}\right)\right\rfloor = 0$$

$$1 \xrightarrow{\binom{0}{1}} \left\lfloor \tfrac{1}{2}\left(1 - (2\ -1)\binom{0}{1}\right)\right\rfloor = 1$$

$$1 \xrightarrow{\binom{1}{0}} \left\lfloor \tfrac{1}{2}\left(1 - (2\ -1)\binom{1}{0}\right)\right\rfloor = -1$$

$$\vdots$$

$A_{2x-y\leq 2}$:



Noh:

- $\binom{0}{0}\binom{0}{1} \in L(A_{\leq})$ and $2 \cdot 0 - 2 \leq 2$
- $\binom{0}{0}\binom{0}{0}\binom{1}{0} \notin L(A_{\leq})$ since $2 \cdot 4 - 2 \not\leq 2$.

For additional material see

Esparza, Automata Theory, an Algorithmic Approach, 2010

(see links on webpage)