Timed pushdown automata and branching vector addition systems

Sławomir Lasota
University of Warsaw

joint work with Lorenzo Clemente, Filip Mazowiecki and Ranko Lazic

AVERTS 2016, Chennai
Definable sets

offer a right setting for timed models of computation, like timed automata, or timed pushdown automata.
Definable sets

offer a right setting for timed models of computation, like
timed automata, or timed pushdown automata.

Definable PDA

have decidable non-emptiness problem, by reduction to
an extension of BVASS in dimension 1.
Plan

• Motivation

• Definable NFA

• Definable PDA

• The core problem: equations over sets of integers

• Branching vector addition systems in dimension 1
Time domain

- reals
- rationals
- integers

\{ \text{dense time} \}

\text{discrete time}

any choice of time domain is fine
Time domain

- reals
- rationals
- integers

\{ \text{dense time} \}
\{ \text{discrete time} \}

any choice of time domain is fine
Time domain

- reals
- rationals
- integers

\{ \text{dense time} \}

\text{discrete time}

No restriction to non-negative!

any choice of time domain is fine
Let input alphabet be **reals**

No restriction to non-negative!

Any choice of time domain is fine.

- **reals**
- rationals
- integers

**dense time**

**discrete time**
Let input alphabet be \textbf{reals}

Timed automata assume monotonic input words:
Timed automata with uninitialized clocks [Alur, Dill 1990]
Timed automata with uninitialized clocks [Alur, Dill 1990]
Timed automata \[\text{with uninitialized clocks}\] [Alur, Dill 1990]

input alphabet = reals
Timed automata
with uninitialized clocks

input alphabet = reals

c_1 := 0

[Alur, Dill 1990]
Timed automata with uninitialized clocks

input alphabet = reals

$c_1 := 0$

$0 < c_1 < 2$

$c_2 := 0$

[Alur, Dill 1990]
Timed automata with uninitialized clocks

input alphabet = reals

$c_1 := 0$

$0 < c_1 < 2$

$c_2 := 0$

$(2 < c_1 < 3) \land (c_2 = 1 \lor c_2 = 2)$

[Alur, Dill 1990]
Timed automata with uninitialized clocks

\[ c_1 := 0 \]
\[ 0 < c_1 < 2 \]
\[ c_2 := 0 \]
\[ (2 < c_1 < 3) \land (c_2 = 1 \lor c_2 = 2) \]

input alphabet = reals

the automaton accepts words \( t_1 \ t_2 \ t_3 \in \mathbb{R}^3 \) such that
Timed automata with uninitialized clocks

Input alphabet = reals

The automaton accepts words $t_1 \ t_2 \ t_3 \in \mathbb{R}^3$ such that

$0 < c_1 < 2$
$c_2 := 0$
$(2 < c_1 < 3) \land (c_2 = 1 \lor c_2 = 2)$
Timed automata

with uninitialized clocks

input alphabet = reals

c₁ := 0
0 < c₁ < 2
c₂ := 0
(2 < c₁ < 3) ∧
(c₂ = 1 ∨ c₂ = 2)

the automaton accepts words t₁ t₂ t₃ ∈ R³ such that

2..3

0..2

{t₁ t₂ t₃}
Timed automata with uninitialized clocks

input alphabet = reals

\[ c_1 := 0 \]
\[ 0 < c_1 < 2 \]
\[ c_2 := 0 \]
\[ (2 < c_1 < 3) \land (c_2 = 1 \lor c_2 = 2) \]

The automaton accepts words \( t_1 \ t_2 \ t_3 \in \mathbb{R}^3 \) such that

\[ t_1 \ t_2 \ t_3 \in \{1 \text{ or } 2\} \]

\[ \{0..2\} \]

\[ \{2..3\} \]
Deterministic timed automata don’t minimize

\[ c_1 := 0 \]
\[ 0 < c_1 < 2 \]
\[ c_2 := 0 \]
\[ (2 < c_1 < 3) \land (c_2 = 1 \lor c_2 = 2) \]

\[ c_1 : 0 \to 2..3 \]
\[ c_2 : 0..2 \]
\[ 1 \lor 2 \]
Deterministic timed automata don’t minimize

\[ c_1 := 0 \]
\[ 0 < c_1 < 2 \]
\[ c_2 := 0 \]
\[ (2 < c_1 < 3) \land (c_2 = 1 \lor c_2 = 2) \]
Deterministic timed automata don’t minimize

\[
\begin{align*}
\text{c}_1 &:= 0 \\
0 < c_1 < 2 \\
\text{c}_2 &:= 0 \\
(2 < c_1 < 3) \land (c_2 = 1 \lor c_2 = 2) \\
(c_1=0, c_2=\frac{1}{3}) &\equiv (c_1=0, c_2=1\frac{1}{3})
\end{align*}
\]
Towards timed pushdown automata
Towards timed pushdown automata

- timed automata [Alur, Dill 1990]
Towards timed pushdown automata

- timed automata [Alur, Dill 1990]
- pushdown timed automata [Bouajjani, Echahed, Robbana 1994]
Towards timed pushdown automata

• timed automata [Alur, Dill 1990] finite stack alphabet
• pushdown timed automata [Bouajjani, Echahed, Robbana 1994]
• dense-timed pushdown automata [Abdulla, Atig, Stenman 2012]
  • clocks can be pushed onto stack
  • the emptiness problem \text{EXPTIME}-c
Towards timed pushdown automata

- timed automata [Alur, Dill 1990] finite stack alphabet
- pushdown timed automata [Bouajjani, Echahed, Robbana 1994]
- dense-timed pushdown automata [Abdulla, Atig, Stenman 2012]
- recursive timed automata [Trivedi, Wojtczak 2010], [Benerecetti, Minopoli, Peron 2010]
Towards timed pushdown automata

- timed automata [Alur, Dill 1990] finite stack alphabet
- pushdown timed automata [Bouajjani, Echahed, Robbana 1994]
- dense-timed pushdown automata [Abdulla, Atig, Stenman 2012]
  - clocks can be pushed onto stack
  - the emptiness problem EXPTIME-c

Theorem 1: [Clemente, L. 2015]
Dense-timed pushdown automata are expressively equivalent to pushdown timed automata.
Towards timed pushdown automata

- timed automata [Alur, Dill 1990]
  finite stack alphabet
- pushdown timed automata [Bouajjani, Echahed, Robbana 1994]
- dense-timed pushdown automata [Abdulla, Atig, Stenman 2012]
  - clocks can be pushed onto stack
  - the emptiness problem EXPTIME-c

Theorem 1: [Clemente, L. 2015]
Dense-timed pushdown automata are expressively equivalent to pushdown timed automata.

An accidental combination of
- stack discipline
- monotonicity of time
- syntactic restrictions
• do not invent a new definition
• do not invent a new definition

• re-interpret a classical definition in \textbf{definable} sets, with finiteness relaxed to \textbf{orbit-finiteness}
• do not invent a new definition
• re-interpret a classical definition in **definable** sets, with finiteness relaxed to **orbit-finiteness**

• alphabet $A$
• states $Q$
• transitions $\delta \subseteq Q \times A \times Q$
• $I, F \subseteq Q$
• do not invent a new definition

• re-interpret a classical definition in **definable** sets, with finiteness relaxed to **orbit-finiteness**

• alphabet $A$
• states $Q$
• transitions $\delta \subseteq Q \times A \times Q$
• $I, F \subseteq Q$
• do not invent a new definition

• re-interpret a classical definition in definable sets, with finiteness relaxed to orbit-finiteness

• alphabet \( A \)
• states \( Q \)
• transitions \( \delta \subseteq Q \times A \times Q \)
• \( I, F \subseteq Q \)
In search of lost definition

- Motivation
- Definable NFA
- Definable PDA
- The core problem: equations over sets of integers
- Branching vector addition systems in dimension 1
Motivation

Definable NFA

Definable PDA

The core problem: equations over sets of integers

Branching vector addition systems in dimension 1

NFA re-interpreted in definable sets
Timed automata are register automata with uninitialized clocks

[Bojańczyk, L. 2012]
Timed automata are register automata with uninitialized clocks

[Bojańczyk, L. 2012]
Timed automata are register automata

\[ c_1 := 0 \]
\[ 0 < c_1 < 2 \]
\[ c_2 := 0 \]
\[ (2 < c_1 < 3) \land (c_2 = 1 \lor c_2 = 2) \]

\[ t \]
\[ c_1 := t \]
\[ 0 < t - c_1 < 2 \]
\[ c_2 := t \]
Timed automata are register automata
with uninitialized clocks

[Bojańczyk, L. 2012]

$c_1 := 0$
$0 < c_1 < 2$
$c_2 := 0$
$(2 < c_1 < 3) \land$
$(c_2 = 1 \lor c_2 = 2)$

$t$
$c_1 := t$
$0 < t-c_1 < 2$
$c_2 := t$
$(2 < t-c_1 < 3) \land$
Timed automata are register automata

[Bojańczyk, L. 2012]

\[ c_1 := 0 \quad 0 < c_1 < 2 \quad (2 < c_1 < 3) \land (c_2 = 1 \lor c_2 = 2) \]

\[ c_2 := 0 \]

\[ t \quad c_1 := t \quad 0 < t-c_1 < 2 \quad (2 < t-c_1 < 3) \land (t-c_2 = 1 \lor t-c_2 = 2) \]

\[ c_2 := t \]
Timed automata are register automata with uninitialized clocks

\[ c_1 := 0 \]
\[ 0 < c_1 < 2 \]
\[ c_2 := 0 \]
\[ (2 < c_1 < 3) \land (c_2 = 1 \lor c_2 = 2) \]

\[ c_1 := t \]
\[ 0 < t - c_1 < 2 \]
\[ c_2 := t \]
\[ (2 < t - c_1 < 3) \land (t - c_2 = 1 \lor t - c_2 = 2) \]

the guards use the structure \((\mathbb{R}, <, +1)\)

\[ 0 < t-c_1 < 2 \iff c_1 < t < c_1+2 \]

[Bojańczyk, L. 2012]
Timed automata are register automata

\[ c_1 := 0 \]
\[ 0 < c_1 < 2 \]
\[ c_2 := 0 \]
\[ (2 < c_1 < 3) \land (c_2 = 1 \lor c_2 = 2) \]

With uninitialized clocks, the guards use the structure \((\mathbb{R}, <, +1)\)

\[ 0 < t - c_1 < 2 \]
\[ c_2 := t \]
\[ (2 < t - c_2 < 3) \land (t - c_2 = 1 \lor t - c_2 = 2) \]

\[ c_1 := t \]
\[ 0 < t - c_1 < 2 \]
\[ c_2 := t \]

\[ 0 < c_2 - c_1 < 2 \]
Timed automata are register automata with uninitialized clocks

[Bojańczyk, L. 2012]

c₁ := 0

0 < c₁ < 2

(2 < c₁ < 3) ∧
(c₂ = 1 ∨ c₂ = 2)

c₂ := 0

0 < c₂ − c₁ < 2

(2 < c₂ − c₁ < 3) ∧
(c₂ = 1 ∨ t - c₂ = 2)

t

c₁ := t

0 < t - c₁ < 2

(t - c₂ = 1 ∨ t - c₂ = 2)

c₂ := t

0 < c₂ - c₁ < 2

the guards use the structure \((R, <, +1)\)

e.g. \(0 < t - c₁ < 2\) iff \(c₁ < t < c₁ + 2\)
(<, +1)-definable sets

FO(<, +1) formula \( \phi(x_1, \ldots, x_n) \) defines a subset of \( n \)-tuples of reals, for instance

\[
\phi(x_1, x_2) \equiv \exists x_3 \ (x_1 < x_3 \land x_2 = x_3 + 3)
\]
definable sets

An FO($<, +1$) formula $\phi(x_1, \ldots, x_n)$ defines a subset of $n$-tuples of reals, for instance

$$\phi(x_1, x_2) \equiv \exists x_3 \ (x_1 < x_3 \land x_2 = x_3 + 3)$$
definable sets

$\text{FO}(\prec, +1)$ formula $\phi(x_1, \ldots, x_n)$ defines a subset of $n$-tuples of reals, for instance

$$\phi(x_1, x_2) \equiv \exists x_3 \ (x_1 < x_3 \land x_2 = x_3 + 3)$$

$\text{FO}(\prec, +1) = \text{QF}(\prec, +1) =$
definable sets

FO($<$, +1) formula $\phi(x_1, \ldots, x_n)$ defines a subset of $n$-tuples of reals, for instance

$$\phi(x_1, x_2) \equiv \exists x_3 \ (x_1 < x_3 \land x_2 = x_3 + 3)$$

$$\text{FO}(<, +1) = \text{QF}(<, +1) = \bigvee_{\text{finite}} \bigwedge_{\text{finite}} x_i - x_j \in I_{ij}$$
definable sets

**FO(<, +1) formula** $\phi(x_1, \ldots, x_n)$ defines a subset of $n$-tuples of reals, for instance

$$\phi(x_1, x_2) \equiv \exists x_3 \ (x_1 < x_3 \land x_2 = x_3 + 3)$$

**FO(<, +1) = QF(<, +1) = \bigvee_{\text{finite}} \bigwedge_{\text{finite}} (x_i - x_j \in I_{ij})**

for instance:

$$\phi(x_1, x_2) \equiv x_1 + 3 < x_2 \equiv x_2 - x_1 \in (3, \infty)$$
Automorphisms $\pi$ of $(\mathbb{R}, <, +1)$ act on a definable set thus splitting it into orbits.
Orbit-finiteness

Automorphisms $\pi$ of $(\mathbb{R}, <, +1)$ act on a definable set thus splitting it into orbits.

For instance, $(-1, \frac{1}{3})$ and $(3, 4\frac{1}{3})$ and $(1\frac{1}{3}, 3)$ are in the same orbit.
Automorphisms $\pi$ of $(\mathbb{R}, <, +1)$ act on a definable set thus splitting it into orbits.

For instance, $(-1, \frac{1}{3})$ and $(3, 4\frac{1}{3})$ and $(1\frac{1}{3}, 3)$ are in the same orbit.

Example:

$x_1 + 3 < x_2 \equiv x_2 - x_1 \in (3, \infty)$

orbit-infinite
Automorphisms $\pi$ of $(R, <, +1)$ act on a definable set thus splitting it into orbits.

For instance, $(-1, \frac{1}{3})$ and $(3, 4\frac{1}{3})$ and $(1\frac{1}{3}, 3)$ are in the same orbit.

Example:

$$x_1 + 3 < x_2 \equiv x_2 - x_1 \in (3, \infty)$$ orbit-infinite

$$x_1 + 3 < x_2 \leq x_1 + 7 \equiv x_2 - x_1 \in (3, 7]$$ orbit-finite
Automorphisms $\pi$ of $(\mathbb{R}, <, +1)$ act on a definable set thus splitting it into orbits.

For instance, $(-1, \frac{1}{3})$ and $(3, 4\frac{1}{3})$ and $(1\frac{1}{3}, 3)$ are in the same orbit.

Example:

$x_1 + 3 < x_2 \equiv x_2 - x_1 \in (3, \infty)$ \hspace{1cm} \text{orbit-infinite}

$x_1 + 3 < x_2 \leq x_1 + 7 \equiv x_2 - x_1 \in (3, 7]$ \hspace{1cm} \text{orbit-finite}

A definable set is orbit-finite

iff

it is defined using bounded intervals only
Definable NFA

- alphabet $A$
- states $Q$
- transitions $\delta \subseteq Q \times A \times Q$
- $I, F \subseteq Q$
Definable NFA

- alphabet $A$
- states $Q$
- transitions $\delta \subseteq Q \times A \times Q$
- $I, F \subseteq Q$
Definable NFA

- alphabet $A$ 
  $\phi_A(x_1, \ldots, x_n)$
- states $Q$ 
  $\phi_Q(x_1, \ldots, x_m)$
- transitions $\delta \subseteq Q \times A \times Q$ 
  $\phi_\delta(x_1, \ldots, x_{m+n+m})$
- $I, F \subseteq Q$ 
  $\phi_I(x_1, \ldots, x_m), \phi_F(x_1, \ldots, x_m)$
Definable NFA

- alphabet $A$
- states $Q$
- transitions $\delta \subseteq Q \times A \times Q$
- $I, F \subseteq Q$

\[
\begin{align*}
\{ & \phi_A(x_1, \ldots, x_n) \\
& \phi_Q(x_1, \ldots, x_m) \\
& \phi_\delta(x_1, \ldots, x_{m+n+m}) \\
& \phi_I(x_1, \ldots, x_m), \ \phi_F(x_1, \ldots, x_m) \\
\} \quad (<, +1)\text{-definable}
\end{align*}
\]
Definable NFA

- alphabet $A$
- states $Q$
- transitions $\delta \subseteq Q \times A \times Q$
- $I, F \subseteq Q$

Runs, acceptance, language recognized, etc. are defined exactly as for classical NFA!

$(<, +1)$-definable

$\phi_A(x_1, \ldots, x_n)$
$\phi_Q(x_1, \ldots, x_m)$
$\phi_\delta(x_1, \ldots, x_{m+n+m})$
$\phi_I(x_1, \ldots, x_m), \phi_F(x_1, \ldots, x_m)$
Definable NFA

- alphabet $A$
- states $Q$
- transitions $\delta \subseteq Q \times A \times Q$
- $I, F \subseteq Q$

$\{\text{orbit-finite}\} \implies \{\text{(<, +1)-definable}\}$

$\phi_A(x_1, \ldots, x_n)$
$\phi_Q(x_1, \ldots, x_m)$
$\phi_\delta(x_1, \ldots, x_{m+n+m})$
$\phi_I(x_1, \ldots, x_m), \phi_F(x_1, \ldots, x_m)$

Runs, acceptance, language recognized, etc. are defined exactly as for classical NFA!
Register automata = definable NFA

c_1 := t
0 < t - c_1 < 2
c_2 := t
0 < c_2 - c_1 < 2

(2 < t - c_1 < 3) ∧
(t - c_2 = 1 ∨ t - c_2 = 2)
Register automata = definable NFA

\[
Q = \{ \perp \} \cup R \cup \{ (c_1, c_2) \in \mathbb{R} \times \mathbb{R} : 0 < c_2 - c_1 < 2 \} \cup \{ T \}
\]
Register automata = definable NFA

\[ Q = \{\bot\} \cup R \cup \{ (c_1, c_2) \in \mathbb{R} \times \mathbb{R} : 0 < c_2 - c_1 < 2 \} \cup \{\top\} \]

\[ \phi_Q(c_0, c_1, c_2) \equiv c_0 = c_1 = c_2 \lor c_0 + 1 = c_1 = c_2 \lor c_0 + 2 = c_1 < c_2 < c_1 + 2 \lor c_0 + 3 = c_1 = c_2 \]

states:

- \( c_1 := t \)
- \( 0 < t - c_1 < 2 \)
- \( c_2 := t \)
- \( (2 < t - c_1 < 3) \land (t - c_2 = 1 \lor t - c_2 = 2) \)
Register automata = definable NFA

states: \[ Q = \{ \bot \} \cup R \cup \{ (c_1, c_2) \in \mathbb{R} \times \mathbb{R} : 0 < c_2 - c_1 < 2 \} \cup \{ T \} \]
\[ \phi_Q(c_0, c_1, c_2) \equiv c_0 = c_1 = c_2 \lor c_0 + 1 = c_1 = c_2 \lor c_0 + 2 = c_1 < c_2 < c_1 + 2 \lor c_0 + 3 = c_1 = c_2 \]

transitions: \[ \delta = \{ (\bot, t, c_1') : c_1' = t \} \cup \]
Register automata = definable NFA

states: \( Q = \{ \bot \} \cup R \cup \{ (c_1, c_2) \in \mathbb{R} \times \mathbb{R} : 0 < c_2 - c_1 < 2 \} \cup \{ \top \} \)

\( \phi_Q(c_0, c_1, c_2) \equiv c_0 = c_1 = c_2 \lor c_0 + 1 = c_1 = c_2 \lor c_0 + 2 = c_1 < c_2 < c_1 + 2 \lor c_0 + 3 = c_1 = c_2 \)

transitions: \( \delta = \{ (\bot, t, c_1') : c_1' = t \} \cup \{ (c_1, t, (c_1', c_2')) : 0 < t - c_1 < 2 \land c_1 = c_1' \land c_2' = t \} \)
Register automata = definable NFA

states: $Q = \{ \bot \} \cup R \cup \{ (c_1, c_2) \in \mathbb{R} \times \mathbb{R} : 0 < c_2 - c_1 < 2 \} \cup \{ \top \}$

$\phi_Q(c_0, c_1, c_2) \equiv c_0 = c_1 = c_2 \lor c_0 + 1 = c_1 = c_2 \lor c_0 + 2 = c_1 < c_2 < c_1 + 2 \lor c_0 + 3 = c_1 = c_2$

transitions: $\delta = \{ (\bot, t, c_1') : c_1' = t \} \cup$

$\{ (c_1, t, (c_1', c_2')) : 0 < t - c_1 < 2 \land c_1 = c_1' \land c_2' = t \} \cup$

$\{ ((c_1, c_2), t, \top) : (2 < t - c_1 < 3) \land (t - c_2 = 1 \lor t - c_2 = 2) \}$
Register automata = definable NFA

states: \( Q = \{ \bot \} \cup R \cup \{ (c_1, c_2) \in \mathbb{R} \times \mathbb{R} : 0 < c_2 - c_1 < 2 \} \cup \{ \top \} \)

\( \phi_Q(c_0, c_1, c_2) \equiv c_0 = c_1 = c_2 \lor c_0 + 1 = c_1 = c_2 \lor c_0 + 2 = c_1 < c_2 < c_1 + 2 \lor c_0 + 3 = c_1 = c_2 \)

transitions: \( \delta = \{ (\bot, t, c_1') : c_1' = t \} \cup \)

\( \{ (c_1, t, (c_1', c_2')) : 0 < t - c_1 < 2 \land c_1 = c_1' \land c_2' = t \} \cup \)

\( \{ ((c_1, c_2), t, \top) : (2 < t - c_1 < 3) \land (t - c_2 = 1 \lor t - c_2 = 2) \} \)

\( \phi_\delta(c_0, c_1, c_2, t, c_0', c_1', c_2') \equiv \ldots \)
Timed automata vs. definable NFA

Definable NFA are like **updatable** timed automata [Bouyer, Duford, Fleury 2000], but:
Timed automata vs. definable NFA

Definable NFA are like updatable timed automata [Bouyer, Duford, Fleury 2000], but:

- in every location, clock valuations are restricted by an orbit-finite constraint (invariant)
Definable NFA are like **updatable** timed automata [Bouyer, Duford, Fleury 2000], but:

- in every location, clock valuations are restricted by an **orbit-finite** constraint (invariant)
- number of clocks may vary from one location to another
Definable NFA are like updatable timed automata [Bouyer, Duford, Fleury 2000], but:

• in every location, clock valuations are restricted by an orbit-finite constraint (invariant)

• number of clocks may vary from one location to another

• the input needs not be monotonic (but can be enforced to be) nor non-negative
Timed automata vs. definable NFA

Definable NFA are like updatable timed automata [Bouyer, Duford, Fleury 2000], but:

- in every location, clock valuations are restricted by an orbit-finite constraint (invariant)
- number of clocks may vary from one location to another
- the input needs not be monotonic (but can be enforced to be) nor non-negative
- alphabet letters may be tuples of timestamps
Timed automata vs. definable NFA

definable NFA

timed automata
  with uninitialized clocks
Timed automata vs. definable NFA

deterministic **definable** NFA

deterministic timed automata
  with uninitialized clocks
Timed automata vs. definable NFA

deterministic **definable** NFA

deterministic timed automata
  with uninitialized clocks

integer
Timed automata vs. definable NFA

deterministic **definable** NFA

deterministic **timed automata**  
with uninitialized clocks

integer

< 2
Timed automata vs. definable NFA

deterministic **definable** NFA

deterministic timed automata
  with uninitialized clocks

- integer
- $< 2$
Timed automata vs. definable NFA

deterministic **definable** NFA

deterministic timed automata
  with uninitialized clocks

integer

< 2 < 2 ...

16
Timed automata vs. definable NFA

deterministic definable NFA

deterministic timed automata
  with uninitialized clocks

minimal automata for languages of deterministic timed automata
  with uninitialized clocks

integer

<2 <2 ...

16
Timed automata vs. definable NFA

deterministic **definable** NFA

deterministic timed automata
  with uninitialized clocks

minimal automata for languages of deterministic timed automata
  with uninitialized clocks

integer

< 2 < 2 ...

closed under minimization
Timed automata vs. definable NFA

Theorem: [Bojańczyk, L. 2012]
Deterministic definable NFA do minimize.
Timed automata vs. definable NFA

Theorem: [Bojańczyk, L. 2012] Deterministic definable NFA do minimize. Likewise, if $\text{FO}(<, +1)$ is replaced by $\text{FO}(<, +)$.
In search of lost definition

- Motivation
- Definable NFA
- Definable PDA
- The core problem: equations over sets of integers
- Branching vector addition systems in dimension 1
• Motivation

• Definable NFA

• Definable PDA

• The core problem: equations over sets of integers

• Branching vector addition systems in dimension 1
Definable PDA

- alphabet $A$
- states $Q$
- stack alphabet $S$
- push $\subseteq Q \times A \times Q \times S$
- pop $\subseteq Q \times S \times A \times Q$
- $I, F \subseteq Q$
Definable PDA

- alphabet $A$
- states $Q$
- stack alphabet $S$
- push $\subseteq Q \times A \times Q \times S$
- pop $\subseteq Q \times S \times A \times Q$
- $I, F \subseteq Q$

\[
\begin{align*}
\phi_A(x_1, \ldots, x_n) \\
\phi_Q(x_1, \ldots, x_m) \\
\phi_S(x_1, \ldots, x_k) \\
\phi_{\text{push}}(x_1, \ldots, x_{m+n+m+k}) \\
\phi_{\text{pop}}(x_1, \ldots, x_{m+k+n+m}) \\
\phi_I(x_1, \ldots, x_m), \quad \phi_F(x_1, \ldots, x_m)
\end{align*}
\]
Definable PDA

- alphabet \( A \)
- states \( Q \)
- stack alphabet \( S \)
- \( \text{push} \subseteq Q \times A \times Q \times S \)
- \( \text{pop} \subseteq Q \times S \times A \times Q \)
- \( I, F \subseteq Q \)

\( \{ \text{orbit-finite} \} \)

\( \{ (\langle, +1 \rangle \)-definable \} \)

\[
\phi_A(x_1, \ldots, x_n) \\
\phi_Q(x_1, \ldots, x_m) \\
\phi_S(x_1, \ldots, x_k) \\
\phi_{\text{push}}(x_1, \ldots, x_{m+n+m+k}) \\
\phi_{\text{pop}}(x_1, \ldots, x_{m+k+n+m}) \\
\phi_I(x_1, \ldots, x_m), \phi_F(x_1, \ldots, x_m)
\]

Acceptance defined as for classical PDA.
Example

input alphabet: \( A = R \cup \{\varepsilon\} \)

language: "ordered palindromes of even length over reals"

states:

stack alphabet:

transitions:

initial state:

accepting state:
Example

input alphabet: $A = \mathbb{R} \cup \{ \varepsilon \}$
language: "ordered palindromes of even length over reals"
states: $Q = \mathbb{R} \cup \{ \text{init, finish, acc} \}$
stack alphabet:
transitions:

initial state: init
accepting state: acc
Example

input alphabet: \( A = \mathbb{R} \cup \{\varepsilon\} \)

language: "ordered palindromes of even length over reals"

states: \( Q = \mathbb{R} \cup \{\text{init, finish, acc}\} \)

stack alphabet: \( S = \mathbb{R} \cup \{\bot\} \)

transitions:

initial state: init

accepting state: acc
Example

input alphabet: \( A = \mathbb{R} \cup \{\varepsilon\} \)

language: "ordered palindromes of even length over reals"

states: \( Q = \mathbb{R} \cup \{\text{init, finish, acc}\} \)

stack alphabet: \( S = \mathbb{R} \cup \{\bot\} \)

transitions: \( \text{push} \subseteq Q \times A \times Q \times S \)

<table>
<thead>
<tr>
<th>State</th>
<th>Action</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (\text{init}, \varepsilon, t, \bot) )</td>
<td>push ( \bot ) on stack</td>
<td></td>
</tr>
<tr>
<td>( (t, u, u, u) )</td>
<td></td>
<td>( t &lt; u )</td>
</tr>
<tr>
<td>( (t, u, \text{finish}, u) )</td>
<td></td>
<td>( t &lt; u )</td>
</tr>
</tbody>
</table>

in state \text{init}, without reading input, change state to an arbitrary real \( t \), and push \( \bot \) on stack

initial state: \text{init}

accepting state: \text{acc}
Example

input alphabet: \( A = R \cup \{\epsilon\} \)

language: "ordered palindromes of even length over reals"

states: \( Q = R \cup \{\text{init, finish, acc}\} \)

stack alphabet: \( S = R \cup \{\bot\} \)

transitions:

\[ \text{push} \subseteq Q \times A \times Q \times S \]

in state \text{finish}, pop a real \( t \) from stack, read the same \( t \) from input, and stay in the same state

\[ \begin{align*}
(\text{init}, \epsilon, t, \bot) \\
(t, u, u, u) & \quad \text{if } t < u \\
(t, u, \text{finish}, u) & \quad \text{if } t < u
\end{align*} \]

\[ \text{pop} \subseteq Q \times S \times A \times Q \]

\[ \begin{align*}
(\text{finish}, t, t, \text{finish}) \\
(\text{finish}, \bot, \epsilon, \text{acc})
\end{align*} \]

initial state: \text{init}

accepting state: \text{acc}
Definable prefix rewriting

- alphabet $A$
- states $Q$
- stack alphabet $S$
- $\rho \subseteq Q \times S^* \times A \times Q \times S^*$
- $I, F \subseteq Q$

orbit-finite

$(<, +1)$-definable
Definable prefix rewriting

- alphabet $A$
- states $Q$
- stack alphabet $S$
- $\rho \subseteq Q \times S^{\leq n} \times A \times Q \times S^{\leq m}$
- $I, F \subseteq Q$

orbit-finite

$(<, +1)$-definable
Definable prefix rewriting

- alphabet $A$
- states $Q$
- stack alphabet $S$
- $\rho \subseteq Q \times S^{\leq n} \times A \times Q \times S^{\leq m}$
- $I, F \subseteq Q$

Acceptance defined as for classical prefix rewriting.
Definable context-free grammars

- nonterminal symbols S
- terminal symbols A
- an initial nonterminal symbol
- \( \rho \subseteq S \times (S \cup A)^* \)
Definable context-free grammars

\[
\begin{align*}
\text{orbit-finite} & \quad \text{definable in FO(\langle, +1) } \\
\text{definable in FO(\langle, +1) } & \\
\end{align*}
\]

- nonterminal symbols \( S \)
- terminal symbols \( A \)
- an initial nonterminal symbol \( \rho \subseteq S \times (S \cup A)^{\leq n} \)

Generated language defined as for classical PDA.
Expressiveness of definable models

[Abdulla, Atig, Stenman 2012]

prefix rewriting

PDA

PDA with timeless stack
(finite stack alphabet)

dense-timed PDA
with uninitialized clocks

[Abdulla, Atig, Stenman 2012]
Expressiveness of definable models

[Abdulla, Atig, Stenman 2012]

[Clemente, L. 2015]
Expressiveness of definable models

[Clemente, L. 2015]

prefix rewriting

CFG

dense-timed PDA
with uninitialized clocks
[Abdulla, Atig, Stenman 2012]

PDA

constrained PDA

PDA with
timeless stack
(finite stack alphabet)

palindromes
Expressiveness of definable models

[Abdulla, Atig, Stenman 2012]

[Abdulla, Atig, Stenman 2012]

palindromes over \{a,b\} × reals with the same number of a’s and b’s

prefix rewriting

PDA

PDA with timeless stack
(finite stack alphabet)

dense-timed PDA
with uninitialized clocks

[Clemente, L. 2015]

PDA with

constrained PDA

CFG
Constrained definable PDA

- alphabet $A$
- states $Q$
- stack alphabet $S$
- $\text{push} \subseteq Q \times A \times Q \times S$
- $\text{pop} \subseteq Q \times S \times A \times Q$
- $I, F \subseteq Q$

orbit-finite

$(<, +1)$-definable
Constrained definable PDA

- alphabet $A$
- states $Q$
- stack alphabet $S$
- $\text{push} \subseteq Q \times A \times Q \times S$
- $\text{pop} \subseteq Q \times S \times A \times Q$
- $I, F \subseteq Q$
Constrained definable PDA

- alphabet $A$
- states $Q$
- stack alphabet $S$
- $\text{push} \subseteq Q \times A \times Q \times S$
- $\text{pop} \subseteq Q \times S \times A \times Q$
- $I, F \subseteq Q$

Span of transitions is bounded. **Too strong restriction!**
Constrained definable PDA

- alphabet $A$
- states $Q$
- stack alphabet $S$
- $\text{push} \subseteq Q \times A \times Q \times S$
- $\text{pop} \subseteq Q \times S \times A \times Q$
- $I, F \subseteq Q$

Span of transitions is bounded. Too strong restriction! For instance, such PDA do not recognize palindromes over reals.
Constrained definable PDA

- alphabet $A$
- states $Q$
- stack alphabet $S$
- push $\subseteq Q \times A \times Q \times S$
- pop $\subseteq Q \times S \times A \times Q$
- $I, F \subseteq Q$

$\{\text{orbit-finite}\}$

$\{\langle, +1\rangle\text{-definable}\}$
Constrained definable PDA

- alphabet $A$
- states $Q$
- stack alphabet $S$
- $\text{push} \subseteq Q \times A \times Q \times S$
- $\text{pop} \subseteq Q \times S \times A \times Q$
- $I, F \subseteq Q$

$\{\text{orbit-finite}\} \Rightarrow (\langle, +1\rangle\text{-definable})$
Constrained definable PDA

- alphabet $A$
- states $Q$
- stack alphabet $S$
- $\text{push} \subseteq Q \times A \times Q \times S$
- $\text{pop} \subseteq Q \times S \times A \times Q$
- $I, F \subseteq Q$

Theorem 2: [Clemente, L. 2015]
The non-emptiness problem is in NEXPTIME. For finite stack alphabet, EXPTIME-complete.
Constrained definable PDA

• alphabet $A$
• states $Q$
• stack alphabet $S$
• push $\subseteq Q \times A \times Q \times S$
• pop $\subseteq Q \times S \times A \times Q$
• $I, F \subseteq Q$

Theorem 2: [Clemente, L. 2015]
The non-emptiness problem is in NEXPTIME. For finite stack alphabet, EXPTIME-complete.

Fact: The model subsumes dense-timed PDA with uninitialized clocks.
Decidability of non-emptiness

[Clemente, L. 2015]

prefix rewriting

PDA

constrained PDA

PDA with finite stack alphabet

dense-timed PDA with uninitialized clocks

[Abdulla, Atig, Stenman 2012]
Decidability of non-emptiness

[Abdulla, Atig, Stenman 2012]

PDA with finite stack alphabet

PDA

Constrained PDA

In NEXPTIME

CFG

Prefix rewriting
Decidability of non-emptiness

[Abdulla, Atig, Stenman 2012]

PDA with finite stack alphabet

PDA

Constrained PDA

EXPTIME-c.

In NEXPTIME

Dense-timed PDA

with uninitialized clocks

[Abdulla, Atig, Stenman 2012]
Decidability of non-emptiness

[Clemente, L. 2015]

[Abdulla, Atig, Stenman 2012]
Decidability of non-emptiness

[Clemente, L. 2015]
Decidability of non-emptiness

[Clemente, L. 2015]

- Undecidable
- Prefix rewriting
- PDA
  - In 2-EXPTIME
  - Constrained PDA
    - In NEXPTIME
    - PDA with finite stack alphabet
    - EXPTIME-c.
- Dense-timed PDA
  - With uninitialized clocks
  - [Abdulla, Atig, Stenman 2012]
prefix rewriting

undecidable

PDA in 2-EXPTIME

constrained PDA in NEXPTIME

PDA with finite stack alphabet

dense-timed PDA with uninitialized clocks

[Abdulla, Atig, Stenman 2012]
Theorem 3:
The non-emptiness problem of definable PDA is in 2-EXPTIME.
Theorem 3:
The non-emptiness problem of definable PDA is in 2-EXPTIME.

Complexity gap: EXPTIME … 2-EXPTIME
Towards decision procedure
Towards decision procedure

Notation:  \( q \rightarrow p \) — there is a run from state \( p \) to state \( q \) that starts and ends with the empty stack
Towards decision procedure

Notation: \( q \rightarrow p \) — there is a run from state \( p \) to state \( q \) that starts and ends with the empty stack

(base)

\[
\begin{array}{c}
\text{x} \\
\text{x} \\
\end{array}
\]
Towards decision procedure

Notation: $q \twoheadrightarrow p$ — there is a run from state $p$ to state $q$ that starts and ends with the empty stack

(base) \[ x \twoheadrightarrow x \]

(transitivity) \[ x \twoheadrightarrow y \quad y \twoheadrightarrow z \quad \Rightarrow \quad x \twoheadrightarrow z \]
Towards decision procedure

Notation: $q \rightarrow p$ — there is a run from state $p$ to state $q$ that starts and ends with the empty stack

(base)  

\[
\begin{array}{c}
\quad x \rightarrow x \\
\end{array}
\]

(transitivity)  

\[
\begin{array}{c}
x \rightarrow y \\
y \rightarrow z \\
\hline
x \rightarrow z
\end{array}
\]

(push-pop)  

\[
\begin{array}{c}
x \rightarrow y \\
x' \rightarrow y' \\
\hline
x' \rightarrow y'
\end{array}
\]

if push($x'$, $x$, $s$) and pop($y$, $s$, $y'$) for some stack symbol $s$
Towards decision procedure

Notation: \( q \rightarrow p \) — there is a run from state \( p \) to state \( q \) that starts and ends with the empty stack

(base)

\[
\begin{array}{c}
\text{x} \\
\rightarrow \\
\text{x}
\end{array}
\]

(transitivity)

\[
\begin{array}{c}
\text{x} \\
\rightarrow \\
\text{y} \\
\rightarrow \\
\text{z}
\end{array}
\]

\[
\frac{\text{x} \rightarrow \text{z}}{	ext{x} \rightarrow \text{y}}
\]

(push-pop)

\[
\begin{array}{c}
\text{x} \\
\rightarrow \\
\text{y}
\end{array}
\]

\[
\frac{\text{x} \rightarrow \text{y}}{	ext{x}' \rightarrow \text{y}'}
\]

if \( \text{push}(x', x, s) \) and \( \text{pop}(y, s, y') \) for some stack symbol \( s \)

Problem: how to make this work for orbit-finite state space?
Towards decision procedure

Notation: \( q \rightarrow p \) — there is a run from state \( p \) to state \( q \) that starts and ends with the empty stack

(base) \[
\begin{array}{c}
x \rightarrow x
\end{array}
\]

(transitivity) \[
\begin{array}{c}
x \rightarrow y \quad y \rightarrow z
\end{array}
\]

Problem: how to make this work for orbit-finite state space?

Guideline: think like state = an integer
Towards decision procedure

Notation: \( q \rightarrow p \) — there is a run from state \( p \) to state \( q \) that starts and ends with the empty stack

(base) \[ x \rightarrow x \]

(transitivity) \[ x \rightarrow y \quad y \rightarrow z \quad \rightarrow x \rightarrow z \]

(push-pop) \[ x \rightarrow y \quad \rightarrow x' \rightarrow y' \] if push(\( x' \), \( x \), \( s \)) and pop(\( y \), \( s \), \( y' \)) for some stack symbol \( s \)

Problem: how to make this work for orbit-finite state space?

Guideline: think like state = an integer
capture all differences \( y - x \), for \( x \rightarrow y \)
Towards decision procedure

• Motivation

• Definable NFA

• Definable PDA

• The core problem: equations over sets of integers

• Branching vector addition systems in dimension 1
The core problem: non-emptiness

Given a systems of equations

\[
\begin{align*}
  x_1 &= t_1 \\
  x_2 &= t_2 \\
  \vdots \\
  x_n &= t_n
\end{align*}
\]
The core problem: non-emptiness

Given a systems of equations

\[
\begin{aligned}
  x_1 &= t_1 \\
  x_2 &= t_2 \\
  \vdots \\
  x_n &= t_n
\end{aligned}
\]

where right-hand sides use:
The core problem: non-emptiness

Given a system of equations

\[
\begin{align*}
x_1 &= t_1 \\
x_2 &= t_2 \\
\cdots \\
x_n &= t_n
\end{align*}
\]

where right-hand sides use:
- constants \{-1\}, \{0\}, \{1\}
The core problem: non-emptiness

Given a system of equations

\[
\begin{align*}
x_1 &= t_1 \\
x_2 &= t_2 \\
&\quad \ldots \\
x_n &= t_n
\end{align*}
\]

where right-hand sides use:

- constants \{-1\}, \{0\}, \{1\}
- set union \(\cup\)
The core problem: non-emptiness

Given a systems of equations

\[
\begin{cases}
  x_1 &= t_1 \\
  x_2 &= t_2 \\
  \quad \vdots \\
  x_n &= t_n
\end{cases}
\]

where right-hand sides use:
- constants \{-1\}, \{0\}, \{1\}
- set union \(\cup\)
- point-wise addition +
The core problem: non-emptiness

Given a system of equations:

\[
\begin{align*}
  x_1 &= t_1 \\
  x_2 &= t_2 \\
  \vdots \\
  x_n &= t_n 
\end{align*}
\]

where right-hand sides use:

- constants \{-1\}, \{0\}, \{1\}
- set union \(\cup\)
- point-wise addition +
- limited intersection \(\cap\)
The core problem: non-emptiness

Given a system of equations

\[
\begin{align*}
  x_1 &= t_1 \\
  x_2 &= t_2 \\
  \vdots \\
  x_n &= t_n
\end{align*}
\]

where right-hand sides use:
- constants \{-1\}, \{0\}, \{1\}
- set union \(\cup\)
- point-wise addition \(+\)
- limited intersection \(\cap\)

decide, whether its least solution assigns a non-empty set to \(x_1\)?
The core problem: non-emptiness

Given a systems of equations

\[
\begin{align*}
x_1 &= t_1 \\
x_2 &= t_2 \\
\quad&\ldots\\nx_n &= t_n
\end{align*}
\]

where right-hand sides use:
- constants \{-1\}, \{0\}, \{1\}
- set union \(\cup\)
- point-wise addition +
- limited intersection \(\cap\)

decide, whether its least solution assigns a non-empty set to \(x_1\)?

for instance:

\[
\begin{align*}
x_1 &= \{0\} \cup x_2 + \{1\} \cup x_2 + \{-1\} \\
x_2 &= x_1 + \{1\} \cup x_1 + \{-1\}
\end{align*}
\]
The core problem: non-emptiness

Given a system of equations

\[
\begin{align*}
    x_1 &= t_1 \\
    x_2 &= t_2 \\
    \vdots \\
    x_n &= t_n
\end{align*}
\]

where right-hand sides use:
- constants \{-1\}, \{0\}, \{1\}
- set union \(\cup\)
- point-wise addition +
- limited intersection \(\cap\)

decide whether its least solution assigns a non-empty set to \(x_1\)?

For instance:

\[
\begin{align*}
    x_1 &= \{0\} \cup x_2 + \{1\} \cup x_2 + \{-1\} \\
    x_2 &= x_1 + \{1\} \cup x_1 + \{-1\}
\end{align*}
\]

What is the least solution with respect to inclusion?
definable PDA \quad \text{exponential blowup} \quad \text{systems of equations over sets of integers}
definable PDA \quad \text{exponential blowup} \quad \text{systems of equations over sets of integers}

\begin{align*}
\text{(base)} & \quad \frac{\_}{x \rightarrow x} \\
\text{(transitivity)} & \quad \frac{x \rightarrow y \quad y \rightarrow z}{x \rightarrow z} \\
\text{(push-pop)} & \quad \frac{x \rightarrow y}{x' \rightarrow y'}
\end{align*}
definable PDA \hspace{1cm} \text{exponential blowup} \hspace{1cm} \text{systems of equations over sets of integers}

(base) \hspace{1cm} x \rightarrow x

(transitivity) \hspace{1cm} x \rightarrow y \hspace{0.5cm} y \rightarrow z \hspace{1cm} x \rightarrow z

(push-pop) \hspace{1cm} x \rightarrow y \hspace{1cm} x' \rightarrow y'

Guideline:
think like state = an integer,
capture all differences $y - x$,
for $x \rightarrow y$
definable PDA \[ \xrightarrow{\text{exponential blowup}} \] systems of equations over sets of integers

(base) \[ x \xrightarrow{} x \]

(transitivity) \[ x \xrightarrow{} y \quad y \xrightarrow{} z \]
\[ x \xrightarrow{} z \]

(push-pop) \[ x \xrightarrow{} y \]
\[ x' \xrightarrow{} y' \]

Guideline:
think like state = an integer,
capture all differences \( y - x \),
for \( x \xrightarrow{} y \)
Guideline:
think like state = an integer,
capture all differences \( y - x \),
for \( x \rightarrow y \)
definable PDA → exponential blowup → systems of equations over sets of integers

(base) \( \overset{x \mapsto x}{\longrightarrow} \)

(transitivity) \( \overset{x \mapsto y \quad y \mapsto z}{\longrightarrow} \)
\( \overset{x \mapsto z}{\longrightarrow} \)

(push-pop) \( \overset{x \mapsto y}{\longrightarrow} \)
\( \overset{x' \mapsto y'}{\longrightarrow} \)

\( X_{pp} \supseteq \{0\} \)

\( X_{pr} \supseteq X_{pq} + X_{qr} \)

\( X_{pq} \supseteq (I + (X_{rs} \cap (J+N)) + L) \cap -(M+K) \)

Guideline:
think like state = an integer,
capture all differences \( y - x \),
for \( x \mapsto y \)
The core problem - no intersections

Given a system of equations

\[
\begin{align*}
x_1 &= t_1 \\
x_2 &= t_2 \\
&\quad \ldots \\
x_n &= t_n
\end{align*}
\]

decide, whether its least solution assigns a non-empty set to \(x_1\)?
The core problem - no intersections

Given a systems of equations

\[
\begin{align*}
  x_1 &= t_1 \\
  x_2 &= t_2 \\
  \vdots \\
  x_n &= t_n
\end{align*}
\]

decide, whether its least solution assigns a non-empty set to \( x_1 \)?

How to solve the problem in absence of intersections?

- constants \{-1\}, \{0\}, \{1\}
- set union \( \cup \)
- point-wise addition \( + \)
- limited intersection \( \cap \)
The core problem - no intersections

Given a systems of equations

\[
\begin{align*}
  x_1 &= t_1 \\
  x_2 &= t_2 \\
  \ldots \\
  x_n &= t_n 
\end{align*}
\]

decide, whether its least solution assigns a non-empty set to \( x_1 \)?

How to solve the problem in absence of intersections?

\[
\begin{align*}
  x_1 &= \{0\} \cup x_2 + \{1\} \cup x_2 + \{-1\} \\
  x_2 &= x_1 + \{1\} \cup x_1 + \{-1\}
\end{align*}
\]
The core problem - no intersections

Given a systems of equations

\[
\begin{align*}
  x_1 &= t_1 \\
  x_2 &= t_2 \\
  &\quad \ldots \\
  x_n &= t_n
\end{align*}
\]

decide, whether its least solution assigns a non-empty set to \( x_1 \)?

How to solve the problem in absence of intersections?

\[
\begin{align*}
  x_1 &= \{0\} \cup x_2 + \{1\} \cup x_2 + \{-1\} \\
  x_2 &= x_1 + \{1\} \cup x_1 + \{-1\}
\end{align*}
\]

Decidable in P
The core problem - intersections

Given a system of equations

\[
\begin{align*}
x_1 &= t_1 \\
x_2 &= t_2 \\
\vdots \\
x_n &= t_n
\end{align*}
\]

decide, whether its least solution assigns a non-empty set to $x_1$?

- constants \{-1\}, \{0\}, \{1\}
- set union $\cup$
- point-wise addition $+$
- limited intersection $\cap$
The core problem - intersections

Given a systems of equations

\[
\begin{align*}
  x_1 &= t_1 \\
  x_2 &= t_2 \\
  \ldots \\
  x_n &= t_n 
\end{align*}
\]

decide, whether its least solution assigns a non-empty set to \( x_1 \)?

The problem is undecidable for unlimited intersections.

[Jeż, Okhotin 2010]
The core problem - limited intersection

Given a system of equations

\[
\begin{align*}
    x_1 &= t_1 \\
    x_2 &= t_2 \\
    \vdots \\
    x_n &= t_n
\end{align*}
\]

decide, whether its least solution assigns a non-empty set to \( x_1 \)?

- constants \{-1\}, \{0\}, \{1\}
- set union \( \cup \)
- point-wise addition \( + \)
- limited intersection \( \cap \)
The core problem - limited intersection

Given a systems of equations

\[
\begin{align*}
x_1 &= t_1 \\
x_2 &= t_2 \\
\vdots \\
x_n &= t_n
\end{align*}
\]

decide, whether its least solution assigns a non-empty set to \( x_1 \)?

What about limited intersections: \( \_ \cap I \), for \( I \) a finite interval?
The core problem - limited intersection

Given a system of equations

\[
\begin{align*}
x_1 &= t_1 \\
x_2 &= t_2 \\
\vdots \\
x_n &= t_n
\end{align*}
\]

decide, whether its least solution assigns a non-empty set to \( x_1 \)?

What about limited intersections: \( \_ \cap I \), for I a finite interval?

\[
\begin{align*}
x_1 &= \{0\} \cup x_2 + \{1\} \cup x_2 + \{-1\} \\
x_2 &= (x_1 + \{1\} \cup x_1 + \{-1\}) \cap \{1\}
\end{align*}
\]
The core problem - limited intersection

Given a systems of equations

\[
\begin{align*}
  x_1 &= t_1 \\
  x_2 &= t_2 \\
  \quad \vdots \\
  x_n &= t_n
\end{align*}
\]

decide, whether its least solution assigns a non-empty set to \( x_1 \)?

What about limited intersections: \( \_ \cap I \), for \( I \) a finite interval?

\[
\begin{align*}
  x_1 &= \{0\} \cup x_2 + \{1\} \cup x_2 + \{-1\} \\
  x_2 &= x_1 + \{1\} \cup x_1 + \{-1\}
\end{align*}
\]

- constants \{-1\}, \{0\}, \{1\}
- set union \( \cup \)
- point-wise addition +
- limited intersection \( \cap \)

membership problem
The core problem - limited intersection

Given a system of equations

\[
\begin{align*}
  x_1 &= t_1 \\
  x_2 &= t_2 \\
  &\quad\vdots \\
  x_n &= t_n
\end{align*}
\]

decide, whether its least solution assigns a non-empty set to \( x_1 \)?

What about limited intersections: \( _\cap I \), for \( I \) a finite interval?

\[
\begin{align*}
  x_1 &= \{0\} \cup x_2 + \{1\} \cup x_2 + \{-1\} \\
  x_2 &= \{1\}
\end{align*}
\]
Given a system of equations

\[
\begin{align*}
  x_1 &= t_1 \\
  x_2 &= t_2 \\
  \vdots \\
  x_n &= t_n
\end{align*}
\]

decide, whether its least solution assigns a non-empty set to \( x_1 \)?

What about limited intersections: \( _\cap I \), for \( I \) a finite interval?
The core problem - limited intersection

Given a system of equations

\[
\begin{align*}
x_1 &= t_1 \\
x_2 &= t_2 \\
&\quad\vdots \\
x_n &= t_n
\end{align*}
\]

decide, whether its least solution assigns a non-empty set to \( x_1 \)?

What about limited intersections: \( _\cap I \), for \( I \) a finite interval?

- \textit{NP-complete}
The core problem - limited intersection

Given a systems of equations

\[
\begin{align*}
    x_1 &= t_1 \\
    x_2 &= t_2 \\
    \quad \vdots \\
    x_n &= t_n
\end{align*}
\]

decide, whether its least solution assigns a non-empty set to \( x_1 \) ?

What about limited intersections: \( _\cap I \), for \( I \) a finite interval?

- **NP-complete**
- non-emptiness of constrained definable PDA reduces to the core problem (with exponential blow-up)
The core problem - limited intersection

Given a system of equations

\[
\begin{align*}
  x_1 &= t_1 \\
  x_2 &= t_2 \\
  \vdots \\
  x_n &= t_n
\end{align*}
\]

decide, whether its least solution assigns a non-empty set to \( x_1 \)?

- constants \([-1], \{0\}, \{1\}\)
- set union \(\cup\)
- point-wise addition \(\oplus\)
- limited intersection \(\cap\)
Given a system of equations

\[
\begin{align*}
x_1 &= t_1 \\
x_2 &= t_2 \\
\vdots \\
x_n &= t_n
\end{align*}
\]

decide, whether its least solution assigns a non-empty set to \( x_1 \)?

What about \( x_1 \cap I \), for \( I \) an arbitrary interval?
The core problem - limited intersection

Given a system of equations

\[
\begin{align*}
x_1 &= t_1 \\
x_2 &= t_2 \\
& \quad \vdots \\
x_n &= t_n
\end{align*}
\]

decide, whether its least solution assigns a non-empty set to \( x_1 \)?

What about \( _\cap I \), for \( I \) an arbitrary interval?

- in \textsc{EXPTIME}, by reduction to \textsc{1-BVASS}(+ -)
The core problem - limited intersection

Given a systems of equations

\[
\begin{align*}
x_1 &= t_1 \\
x_2 &= t_2 \\
&\quad \ldots \\
x_n &= t_n
\end{align*}
\]

deide, whether its least solution assigns a non-empty set to \( x_1 \) ?

What about \( _- \cap I \), for I an arbitrary interval?

- in EXPTIME, by reduction to 1-BVASS(\(+\,\)-)
- non-emptiness of definable PDA reduces to the core problem (with exponential blow-up)
Decision procedure

definable PDA

systems of equations over sets of integers
Decision procedure

definable PDA \rightarrow \text{exponential blowup} \rightarrow \text{systems of equations over sets of integers}
Decision procedure

definable PDA \quad \text{exponential blowup} \quad \text{poly} \quad \text{systems of equations over sets of integers}
definable PDA

exponential blowup

poly

systems of equations over sets of integers

poly

1-BVASS(+ -)

Decision procedure
Decision procedure

definable PDA

exponential blowup

poly

systems of equations over sets of integers

effective

1-BVASS(+-)
Decision procedure

definable PDA

effective

exponential blowup

decision procedure

effective

systems of equations over sets of integers

poly

1-BVASS(+ -)

non-emptiness in EXPTIME

poly
Decision procedure

• Motivation

• Definable NFA

• Definable PDA

• The core problem: equations over sets of integers

• Branching vector addition systems in dimension 1
1-BVASS(+ -)
1-BVASS(\(+\ -\))

- automaton with 1 non-negative counter
1-BVASS(+ -)

- automaton with 1 non-negative counter
- run is a tree
1-BVASS(+ -)

• automaton with 1 non-negative counter
• run is a tree
• in leaves: initial state with counter=1
1-BVASS(+-)

- automaton with 1 non-negative counter
- run is a tree
- in leaves: initial state with counter=1
- transition rules:

```
          q
         /|
        / +
       l   r

          q
         /-
       l   r
```
• automaton with 1 non-negative counter
• run is a tree
• in leaves: initial state with counter=1
• transition rules:

\[
\begin{array}{c}
\text{q} \\
+ \\
\text{l} & \text{r}
\end{array}
\quad
\begin{array}{c}
\text{q} \\
- \\
\text{l} & \text{r}
\end{array}
\]

• non-emptiness problem: is there a run with a final state in the root?
Non-emptiness of $1$-BVASS$$(+ - )$
Non-emptiness of 1-BVASS(+-)

Theorem 4:

The non-emptiness problem of 1-BVASS(+-) is in EXPTIME.
Non-emptiness of 1-BVASS(+-)

Theorem 4:
The non-emptiness problem of 1-BVASS(+-) is in EXPTIME.

Proof idea:
Exponentially bounded witness.
Non-emptiness of 1-BVASS(+ -)

Theorem 4:
The non-emptiness problem of 1-BVASS(+ -) is in EXPTIME.

Proof idea:
Exponentially bounded witness.

Complexity gap: PSPACE … EXPTIME
Non-emptiness of 1-BVASS(+ -)

Theorem 4:
The non-emptiness problem of 1-BVASS(+ -) is in EXPTIME.

Proof idea:
Exponentially bounded witness.

Complexity gap: PSPACE … EXPTIME

Theorem: [Goeller, Haase, Lazic, Totzke 2016]
The non-emptiness problem of 1-BVASS(+) is in P (unary encoding).
Definable sets

offer a right setting for timed models of computation, like
timed automata, or timed pushdown automata.

Definable PDA

have decidable non-emptiness problem, by reduction to
an extension of BVASS in dimension 1.
Definable sets

offer a right setting for timed models of computation, like
timed automata, or timed pushdown automata.

Definable PDA

have decidable non-emptiness problem, by reduction to
an extension of BVASS in dimension 1.