Applications of the First-order Integer Arithmetic to the Verification of Programs with Lists

Radu Iosif, joint work with Marius Bozga
Verimag/CNRS (Grenoble, France)

First Order Arithmetic of Natural Numbers

- $\langle \mathbb{N}, +, \cdot, 0, 1 \rangle$ is undecidable [Church '36]

- $\langle \mathbb{N}, +, |, 0, 1 \rangle$ is a subtheory of $\langle \mathbb{N}, +, \cdot, 0, 1 \rangle$: $f|g \leftrightarrow \exists x \; f \cdot x = g$
  - shown to be undecidable [Robinson '49]

- $\langle \mathbb{N}, +, |, 0, 1 \rangle^{\exists}$ is the *existential* fragment of $\langle \mathbb{N}, +, |, 0, 1 \rangle$
  - shown to be decidable [Lipshitz '78]

- $\langle \mathbb{N}, +, 0, 1 \rangle$ is decidable [Presburger '29]

Goal: better understand the boundary between decidable and undecidable subtheories of FO arithmetic.

Applications of FO Arithmetic of $\mathbb{N}$

Automated verification of systems with infinite state spaces:

- Automata with Integer Counters
  - reachability problems of counter automata with flat control structures and affine transition relations is definable in PA [Comon, Yurski '98]
  - reachability problems of counter automata with flat control structures and parameterized transition relations can be defined using more expressive decidable theories [Bozga, I '06]

Applications of FO Arithmetic of $\mathbb{N}$

Automated verification of systems with infinite state spaces:

- Automata with Integer Counters
  - reachability problems of counter automata with flat control structures and affine transition relations is definable in PA [Comon, Yurski '98]
  - reachability problems of counter automata with flat control structures and parameterized transition relations can be defined using more expressive decidable theories [Bozga, I '06]
- Programs with Dynamic Linked Data Structures
  - verification of properties relative to the sizes of data structures
  - programs with lists [Bouajjani, Bozga, Habermehl, I, Moro, Vojnar '06]
  - programs with trees [Habermehl, I, Vojnar '06]
  - recent work on defining decidable classes of programs with lists

Outline

- Decidability within the Theory of Addition and Divisibility
  - Least Common Multiple Fragment
  - Left-Divides Fragment
- Reachability Problems for Programs with Lists
  - n-Cyclic Heaps, $n > 1$
  - Acyclic and 1-Cyclic Heaps
- Conclusions

Proving undecidability of $\langle \mathbb{N}, +, |, 0, 1 \rangle$

Hilbert's 10^{th} Problem:

10. **Determination of the solvability of a Diophantine equation.** Given a diophantine equation with any number of unknown quantities and with rational integral numerical coefficients: *To devise a process according to which it can be determined by a finite number of operations whether the equation is solvable in rational integers.*

Proving undecidability of $\langle \mathbb{N}, +, |, 0, 1 \rangle$

- the least common multiple relation is definable in $\langle \mathbb{N}, +, |, 0, 1 \rangle$:

$$[x, y] = z \iff \forall t \; x|t \wedge y|t \leftrightarrow z|t$$

Proving undecidability of $\langle \mathbb{N}, +, |, 0, 1 \rangle$

- the least common multiple relation is definable in $\langle \mathbb{N}, +, |, 0, 1 \rangle$:

$$[x, y] = z \iff \forall t \ x|t \wedge y|t \leftrightarrow z|t$$

- the perfect square relation is definable in $\langle \mathbb{N}, +, |, 0, 1 \rangle$:

$$x^2 = y \iff y + x = [x, x + 1]$$

Proving undecidability of $\langle \mathbb{N}, +, |, 0, 1 \rangle$

- the least common multiple relation is definable in $\langle \mathbb{N}, +, |, 0, 1 \rangle$:

$$[x, y] = z \iff \forall t \; x|t \wedge y|t \leftrightarrow z|t$$

- the perfect square relation is definable in $\langle \mathbb{N}, +, |, 0, 1 \rangle$:

$$x^2 = y \iff y + x = [x, x + 1]$$

- multiplication is defined using the following:

$$z = xy \iff 4z = (x + y)^2 - (x - y)^2$$

Proving undecidability of $\langle \mathbb{N}, +, |, 0, 1 \rangle$

- the least common multiple relation is definable in $\langle \mathbb{N}, +, |, 0, 1 \rangle$:

$$[x, y] = z \iff \forall t \; x|t \wedge y|t \leftrightarrow z|t$$

- the perfect square relation is definable in $\langle \mathbb{N}, +, |, 0, 1 \rangle$:

$$x^2 = y \iff y + x = [x, x + 1]$$

- multiplication is defined using the following:

$$z = xy \iff 4z = (x + y)^2 - (x - y)^2$$

- we have reduced Hilbert's 10^{th} Problem to satisfiability of $\langle \mathbb{N}, +, |, 0, 1 \rangle$
  - $\langle \mathbb{N}, +, [.], 0, 1 \rangle$ is undecidable
  - $\langle \mathbb{N}, +, |, 0, 1 \rangle$ with one universally quantified variable is undecidable

The Left-Divides Fragment of $\langle \mathbb{N}, +, |, 0, 1 \rangle$

Syntactic fragment of formulas with following *prenex form*:

$$\mathcal{L}_{|}^{(n)}\colon \mathbf{Q_1 z_1 \ldots Q_n z_n R_1 x_1 \ldots R_m x_m} \varphi(\vec{x}, \vec{z})$$

where each divisibility predicate is of the form:

$$f(\vec{z}) | g(\vec{x}, \vec{z})$$

and there are no implicit dependecies between $\vec{z}$ and $\vec{x}$.

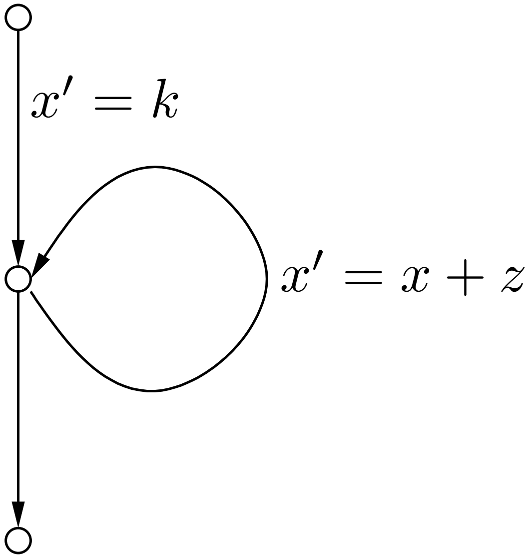
Good subject for research:

- is undecidable but has interesting decidable sub-fragments
- $\mathcal{L}_{|}^{(1)}$ is decidable

The Left-Divides Fragment of $\langle \mathbb{N}, +, |, 0, 1 \rangle$

Consider the following counter machine:

- x is the working variable
- z is system parameter
- $k \in \mathbb{Z}$ is a constant



$$\exists n \geq 0 \; [x = nz + k] \iff z | x - k \wedge (z \geq 0 \wedge x \geq k \;\vee\; z \leq 0 \wedge x \leq k)$$

Tools for Proving Decidability

The Chinese Remainder Theorem

We use a generalized version of the CRT:

$$\exists x \bigwedge_{i=1}^{K} m_i | (a_i x - r_i) \iff \bigwedge_{1 \leq i,j \leq K} (a_i m_j, a_j m_i) | (a_i r_j - a_j r_i) \ \wedge \ \bigwedge_{i=1}^{K} (a_i, m_i) | r_i$$

where (A, B) denotes the greatest common divisor of A and B

CRT eliminates existential quantifiers introducing g.c.d. subterms.

Taming the Greatest Common Divisor

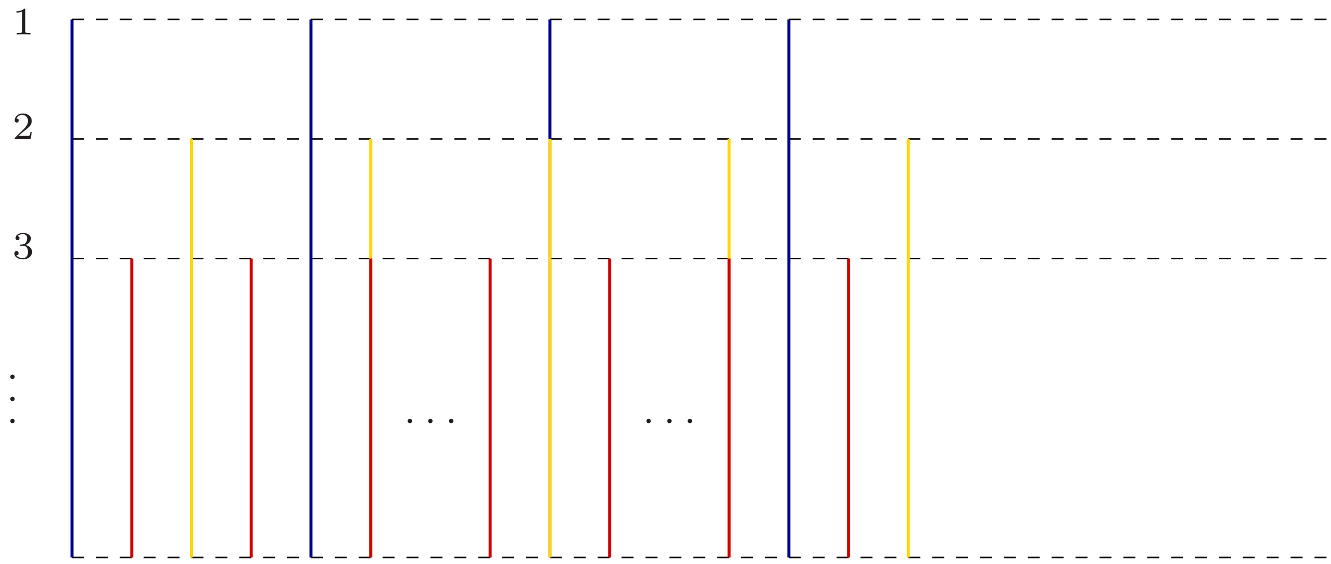
$$A(x) = ax + b, B(x) = cx + d$$

$$\exists p, q, r \in \mathbb{Z}\ \forall x\ .\ pA(x) = qB(x) + r$$

$$\forall x \Big[\big(A(x), B(x) \big) = \big(A(x), r \big) \Big]$$

$$\forall x \Big[\big(A(x), r \big) \mid F(x) \iff \bigvee_{i=0}^{r-1} A(x) \equiv i \mod r\ \wedge\ (i, r) | F(x) \Big]$$

A 25$ Conjecture of Erdös



N arithmetic progressions $\{a_i + b_i\mathbb{N}\}_{i=1}^{N}$ cover $\mathbb{N}$ $\iff$ they cover the set $1, 2, 3, \ldots, 2^N$. [R. Crittenden and C. Vanden Eynden '69]

Elimination of universal quantifiers:

$$\forall x \bigvee_{i=1}^{N} b_i | x - a_i \iff \bigwedge_{t=1}^{2^N} \bigvee_{i=1}^{N} b_i | t - a_i$$

A Generalized Version of Erdös' Lemma

N progressions $\{a_i + b_i\mathbb{N}\}_{i=1}^{N}$ cover a progression $a + b\mathbb{N}$ $\iff$ they cover the set $\{a, a+b, \ldots, a+b\cdot(2^N-1)\}$.

Elimination of universal quantifiers in presence of negation:

$$\forall x \bigvee_{i=1}^{N} b_i | x - a_i \quad \vee \quad \bigvee_{j=1}^{M} c_j \not| \; x - d_j \iff$$

$$\forall x \bigwedge_{j=1}^{M} c_j | x - d_j \quad \rightarrow \quad \bigvee_{i=1}^{N} b_i | x - a_i \iff$$

$$\underbrace{\neg\exists y \bigwedge_{j=1}^{M} c_j | y - d_j}_{\text{vacuity}} \vee \underbrace{\exists y \bigwedge_{j=1}^{M} c_j | y - d_j}_{\text{non-vacuity}} \wedge \underbrace{\bigwedge_{t=1}^{2^N} \bigvee_{i=1}^{N} b_i | \big([c_j]_{j=1}^{M} \cdot t + y\big) - a_i}_{\text{coverage}}$$

Reduction of $\mathcal{L}_{|}^{(1)}$ to Presburger Arithmetic

$$Q_1 x_1 \ldots Q_n x_n \bigvee_{i=1}^{N} \Big(\bigwedge_{j=1}^{M_i} f_{ij}(z) \mid f'_{ij}(\vec{x}, z) \wedge \bigwedge_{j=1}^{P_i} g_{ij}(z) \not| g'_{ij}(\vec{x}, z) \wedge \varphi_i(\vec{x}) \wedge \psi_i(z) \Big)$$

where $f_{ij}, f'_{ij}, g_{ij}, g'_{ij}$ are linear terms and φ_i, ψ_i are Presburger formulae

- eliminate quantified variables $x_n, x_{n-1}, \ldots, x_1$
- translate the solved form into Presburger Arithmetic

Outline

- Decidability within the Theory of Addition and Divisibility
  - Least Common Multiple Fragment
  - Left-Divides Fragment
- Reachability Problems for Programs with Lists
  - n-Cyclic Heaps, $n > 1$
  - Acyclic and 1-Cyclic Heaps
- Conclusions

Programs with Lists

Sequential non-recursive programs with singly-linked list data structures:

```
struct node {
        int data;
        struct node * next;
} * list;
```

A heap is a graph in which each node has at most one successor.

Safety: *whenever the control reaches a certain location, the heap of the program will satisfy a given property*

Termination: *the final location is reachable on every given input*

Syntax of Programs with Lists

- pointer variables $PVar = \{u, v, w, \ldots\}$ ranging over memory locations
- assignments:

  $u :=$ null $\mid u :=$ new $\mid u := w \mid u := w.$next $\mid u.$next $:=$ null $\mid u.$next $:= w$

- conditionals:

  $$\text{while} \mid \text{if } (u = v)\ \ldots$$

- assertions:

  $$\text{assert}(u = v)$$

- may use boolean combinations, e.g.

  $$\text{if } (u = v\ \wedge\ w = t) \ldots$$

Syntax of Programs with Lists

- pointer variables $PVar = \{u, v, w, \ldots\}$ ranging over memory locations
- assignments:

  $u :=$ null $\mid u :=$ new $\mid u := w \mid u := w$.next $\mid u$.next $:=$ null $\mid u$.next $:= w$

Destructive updating assignments

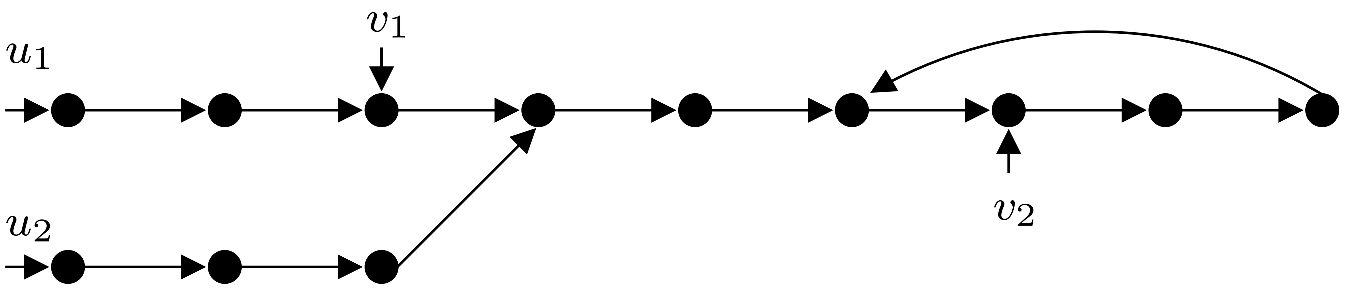
Syntax of Programs with Lists

- pointer variables $PVar = \{u, v, w, \ldots\}$ ranging over memory locations
- assignments:

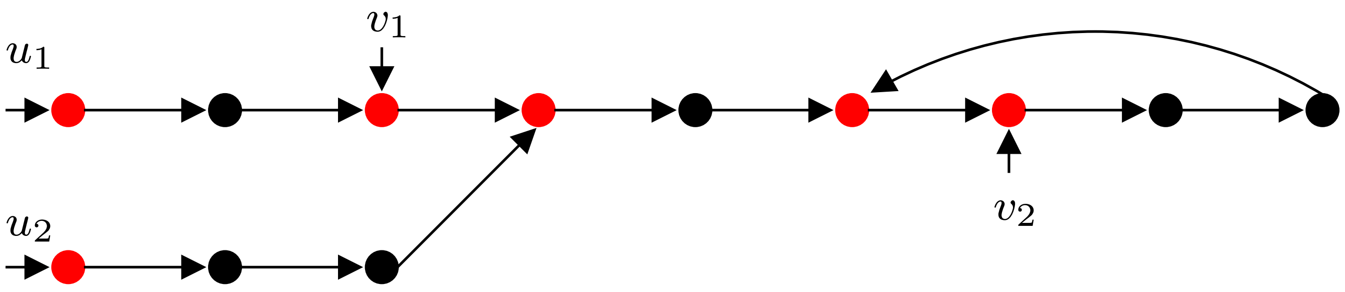
  $u :=$ null $\mid$ $u :=$ new $\mid$ $u := w$ $\mid$ $u := w$.next $\mid$ u.next $:=$ null $\mid$ u.next $:= w$

Flatness: no **while** or **if** occurs inside the body of a **while**

1-Selector Heaps

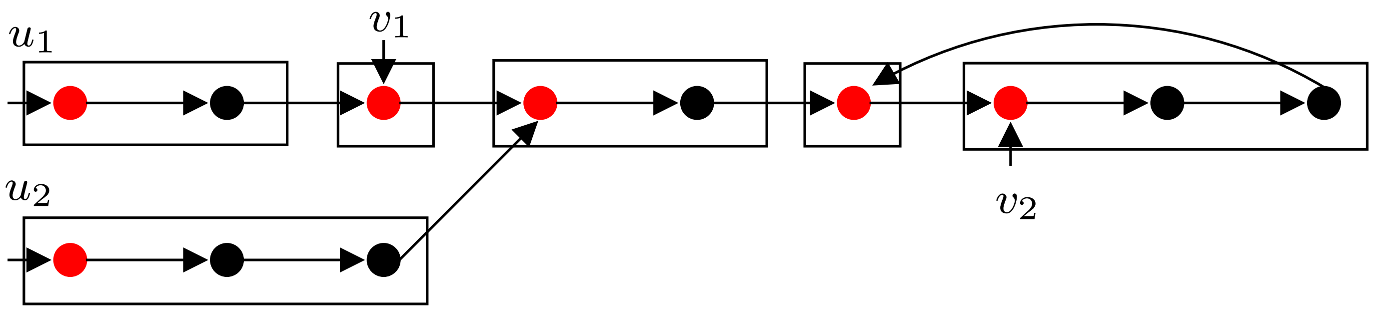


1-Selector Heaps



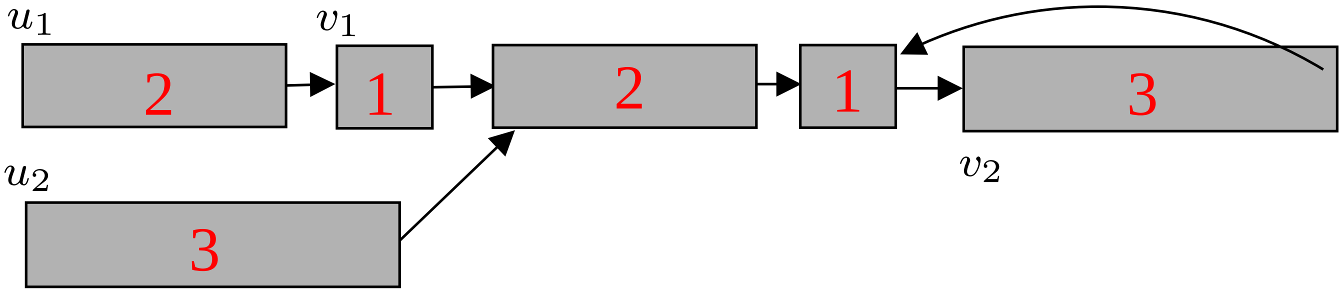
Cut points are either pointed to by a variable or have at least two predecessors

1-Selector Heaps

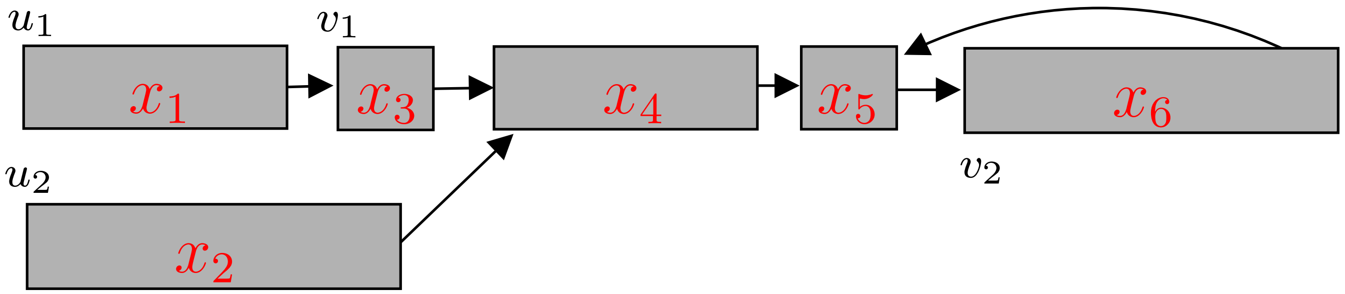


All nodes between two cut points belong to the same equivalence class

Parametric Heaps



Parametric Heaps



Symbolic representation of an infinite set of concrete heaps

Verification Problems for Programs with Lists

[**Safety**] Let P be a list program, $S = \{l_i \ : \ \mathsf{assert}(\varphi_i)\}_{i=1}^{k}$ a set of assertions occurring in P, and $H(\vec{x})$ a parametric heap. P is said to be safe w.r.t $H(\vec{x})$ and S iff for all $H(\vec{n}) \in H(\vec{x})$, and $1 \leq i \leq k$, φ_i is true whenever P, started on input $H(\vec{n})$, reaches l_i.

Verification Problems for Programs with Lists

[**Safety**] Let P be a list program, $S = \{l_i \ : \ \text{assert}(\varphi_i)\}_{i=1}^{k}$ a set of assertions occurring in P, and $H(\vec{x})$ a parametric heap. P is said to be safe w.r.t $H(\vec{x})$ and S iff for all $H(\vec{n}) \in H(\vec{x})$, and $1 \leq i \leq k$, φ_i is true whenever P, started on input $H(\vec{n})$, reaches l_i.

[**Termination**] Let P be a list program, and $H(\vec{x})$ a parametric heap. P is said to terminate w.r.t $H(\vec{x})$ iff for all $H(\vec{n}) \in H(\vec{x})$, P started on input $H(\vec{n})$, has a finite execution.

Undecidability Results

n-Cyclic Heaps $n > 1$

The problems of verifying safety and termination properties, for flat list programs without destructive updates, running on n-cyclic inputs, with $n > 1$, are undecidable.

n-Cyclic Heaps $n > 1$

The problems of verifying safety and termination properties, for flat list programs without destructive updates, running on n-cyclic inputs, with $n > 1$, are undecidable.

- Programs run over unbounded heaps of a given shape
- Runs encode arithmetic relations between lengths of lists in the input

n-Cyclic Heaps $n > 1$

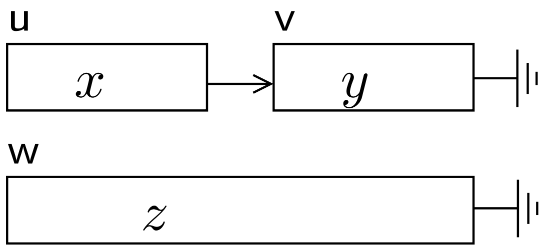
The problems of verifying safety and termination properties, for flat list programs without destructive updates, running on n-cyclic inputs, with $n > 1$, are undecidable.

- Programs run over unbounded heaps of a given shape
- Runs encode arithmetic relations between lengths of lists in the input

Flat Program for Addition $x + y = z$

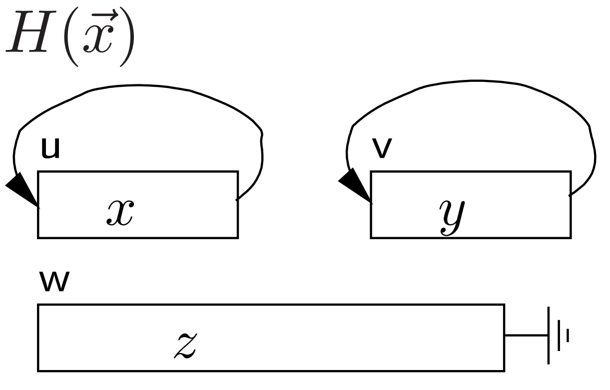
```
1: i := u;
2: j := w;
3: while (i ≠ null ∧ j ≠ null) do
4:          i := i.next;
5:          j := j.next;
6: od;
7: if (i = null ∧ j = null)
OK: ...
          else
ERR: ...
```

$H(\vec{x})$



Flat Program for Least Common Multiple $[x, y] = z$

```
1: i := u.next;
2: j := v.next;
3: k := w.next;
4: while (¬ (i = u ∧ j = v)) do
5:         i := i.next;
6:         j := j.next;
7:         k := k.next;
8: od;
9: if (k = null)
OK: ...
        else
ERR: ...
```



Reduction of complemented HTP to Safety/Termination

Given a Diophantine system S, build a program P and a parametric heap H, such that S doesn't have a solution iff P, started on any input from H, is safe (terminates)

$$\neg\left(\exists \vec{x} \;.\; \bigwedge_{i=1}^{n} P_i(\vec{x}) = 0\right) \iff \forall \vec{x} \;.\; \bigvee_{i=1}^{n} P_i(\vec{x}) \neq 0$$

$Prog(P_1)$;

if $Cond(P_1 = 0)$ then $Prog(P_2)$;

  if $Cond(P_2 = 0)$ then $Prog(P_3)$;

    ...

    assert(false); /* while (true) do od */

    ...

  fi

fi

Extensions of the Undecidability Results

Programs working on arbitrary heaps:

- use a tester to check if the input has a given shape

Extensions of the Undecidability Results

Programs working on arbitrary heaps:

- use a tester to check if the input has a given shape

Non-deterministic programs started with the empty heap:

- use a non-deterministic constructor to build heaps of a given shape

Decidability Results

Acyclic and 1-Cyclic Heaps

The problems of verifying safety and termination properties, for flat list programs without destructive updates, running on n-cyclic inputs, with $n \leq 1$, are decidable.

Acyclic and 1-Cyclic Heaps

The problems of verifying safety and termination properties, for flat list programs without destructive updates, running on n-cyclic inputs, with $n \leq 1$, are decidable.

Encode the program as a counter automaton $M = \langle \vec{x}, Q, q_0, \rightarrow \rangle$:

- $\vec{x} = \{x_1, x_2, \ldots, x_k\}$ is a set of counters
- Q is a set of control states, and $q_0 \in Q$ is the initial state
- $q \xrightarrow{R(\vec{x}, \vec{x}')} q'$, where $R(\vec{x}, \vec{x}')$ is an arithmetic formula

Acyclic and 1-Cyclic Heaps

The problems of verifying safety and termination properties, for flat list programs without destructive updates, running on n-cyclic inputs, with $n \leq 1$, are decidable.

Encode the program as a counter automaton $M = \langle \vec{x}, Q, q_0, \rightarrow \rangle$:

- $\vec{x} = \{x_1, x_2, \ldots, x_k\}$ is a set of counters
- Q is a set of control states, and $q_0 \in Q$ is the initial state
- $q \xrightarrow{R(\vec{x},\vec{x}')} q'$, where $R(\vec{x}, \vec{x}')$ is an arithmetic formula

Reachability relation definability:

- $R^n(\vec{x}, \vec{x}') =$ in/out relation after n iterations, and $R^* = \bigcup_{n \in \mathbb{N}} R^n$
- Is R^* definable in a logic decidable for satisfiability?
- A positive answer entails decidability of reachability problems

A Decidable Class of Counter Automata

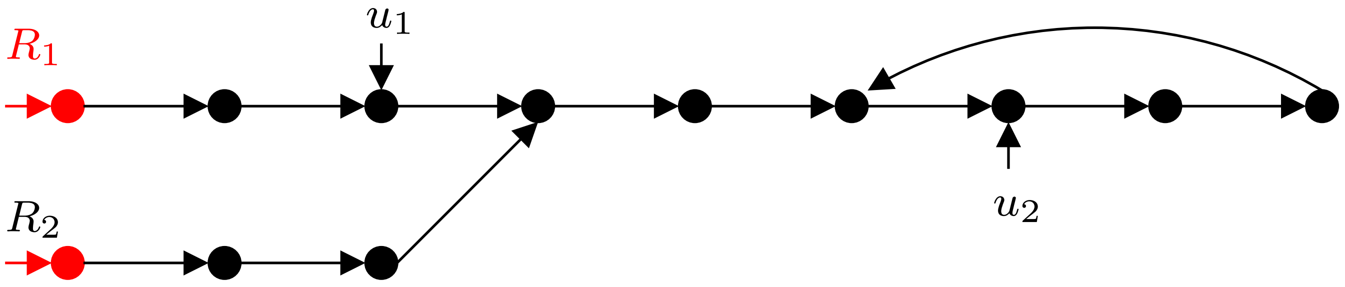
- Flat control structure: no nested loops
- Transitions of the form $q \xrightarrow{g(\vec{x},\vec{y}) \wedge f(\vec{y},\vec{y}')} q'$:
  - $\vec{x}$ are parameters: no assignments to them
  - $\vec{y}$ are counters
  - guards $g(\vec{x}, \vec{y})$ are boolean combinations of linear/divisibility constraints
  - actions $f(\vec{y}, \vec{y}')$ are reset $(y = 0)$, copy $(y_i' = y_j)$ or increment $(y' = y + 1)$

A Decidable Class of Counter Automata

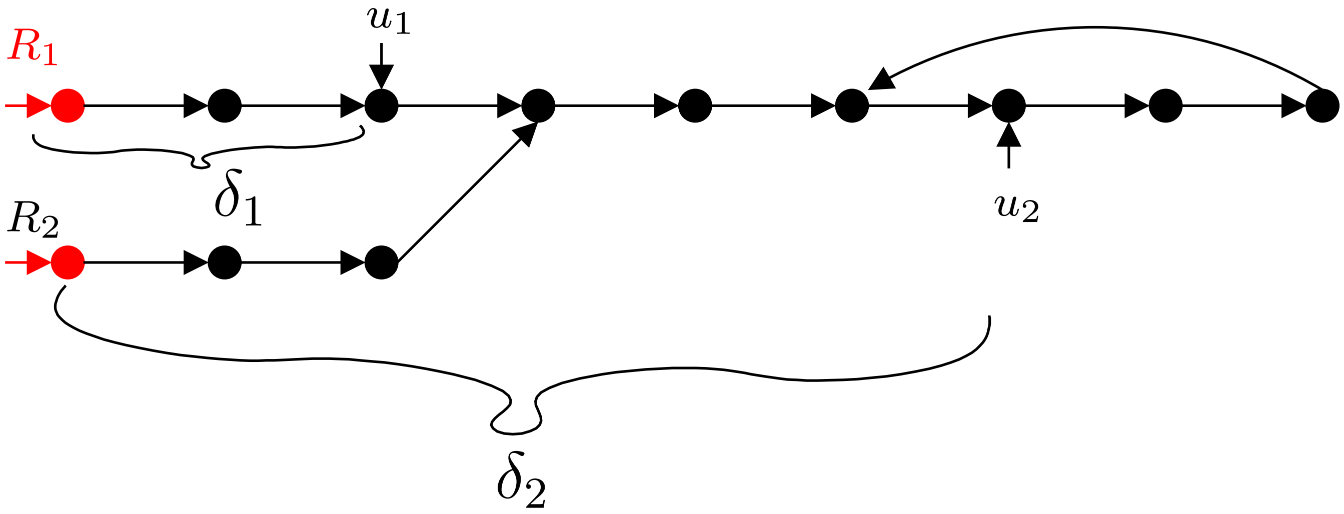
- Flat control structure: no nested loops
- Transitions of the form $q \xrightarrow{g(\vec{x},\vec{y}) \wedge f(\vec{y},\vec{y}')} q'$:
  - $\vec{x}$ are parameters: no assignments to them
  - $\vec{y}$ are counters
  - guards $g(\vec{x},\vec{y})$ are boolean combinations of linear/divisibility constraints
  - actions $f(\vec{y},\vec{y}')$ are reset ($y = 0$), copy ($y_i' = y_j$) or increment ($y' = y + 1$)

For this class R^* is definable in the fragment $\mathcal{L}_|^{(1)}$ of $\langle \mathbb{N}, +, |, 0, 1 \rangle$

Encoding Program States

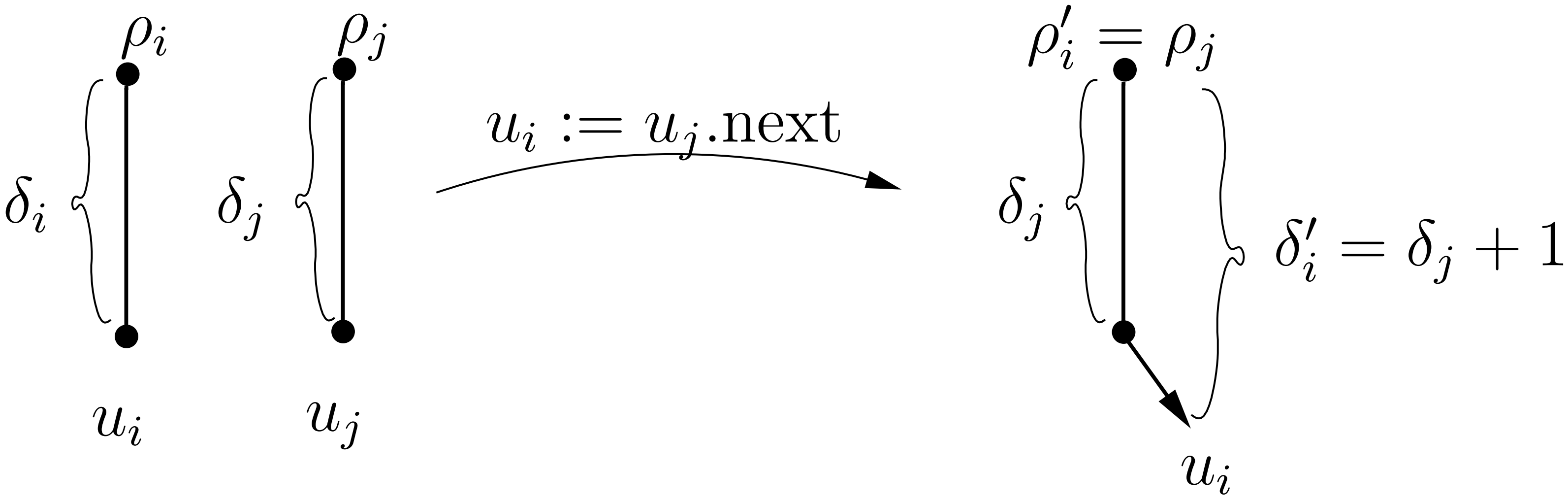


Encoding Program States



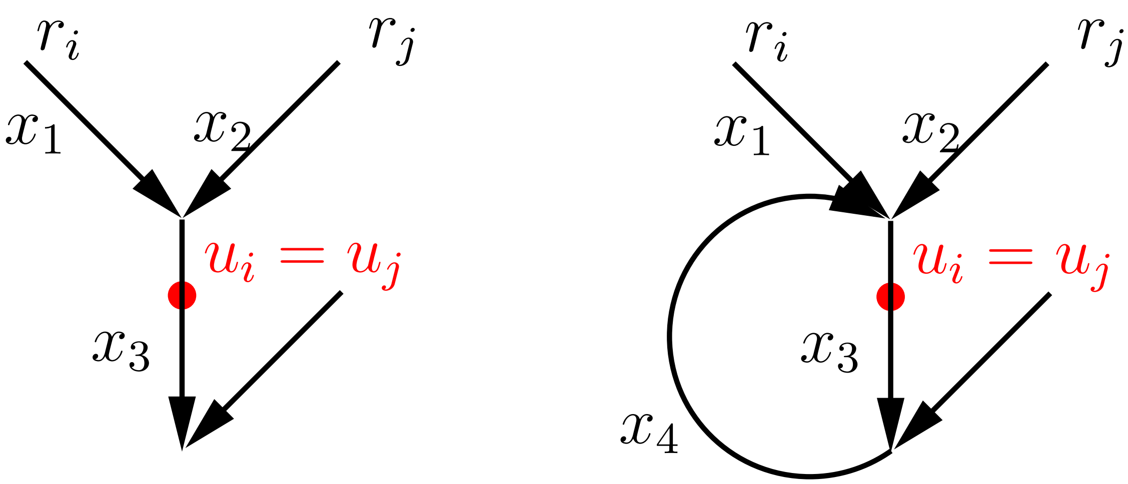
$$\{\langle R_1, \delta_1 \rangle, \langle R_2, \delta_2 \rangle\}$$

Encoding Assignments



The encoding preserves the control structure of the program

Encoding Guards



(a) (b)

If $u_i = \langle r_i, y_i \rangle$ and $u_j = \langle r_j, y_j \rangle$:

a) $0 \le y_i - x_1 \le x_3 \wedge 0 \le y_j - x_2 \le x_3 \wedge y_i - x_1 = y_j - x_2$

b) $0 \le y_i - x_1 \wedge 0 \le y_j - x_2 \wedge x_3 + x_4 \mid (y_i - x_1) - (y_j - x_2)$

Outline

- Decidability within the Theory of Addition and Divisibility
  - Least Common Multiple Fragment
  - Left-Divides Fragment
- Reachability Problems for Programs with Lists
  - n-Cyclic Heaps, $n > 1$
  - Acyclic and 1-Cyclic Heaps
- Conclusions

Conclusions

- Study of satisfiability problem within $\langle \mathbb{N}, +, |, 0, 1 \rangle$:
  - in general quantifier alternation leads to undecidability
  - fragments based on the separation of variables on the left/right of $|$ may be decidable
- Application of arithmetic to programs with singly-linked lists:
  - undecidability of safety/termination due to the complexity of the input
  - flat programs without destructive updates are undecidable if the input heap has more than one cycle
  - decidability established for acyclic and 1-cyclic heaps