Transitive Closures of Affine Integer Tuple Relations and their Overapproximations

Sven Verdoolaege **Albert Cohen** Anna Beletska

PARKAS group, INRIA and École Normale Supérieure de Paris, France

September 15, 2011

Outline

1. **Motivating Applications**
   - Iteration Space Slicing
   - Equivalence Checking
2. Problem Formulation
3. Reachability Analysis
4. Technical Contribution
5. Experimental Results
6. Conclusion and Perspectives

Iteration Space Slicing: Beletska et al.

Can we parallelize this code?

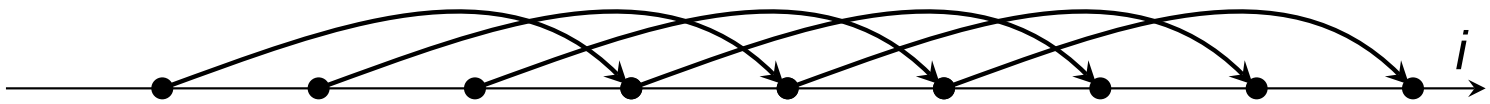
```
for (i = 1; i <= n; ++i)
    a[i] = f(a[i-3]);
```

Iteration Space Slicing: Beletska et al.

Can we parallelize this code?

```
for (i = 1; i <= n; ++i)
    a[i] = f(a[i-3]);
```

Dependences:

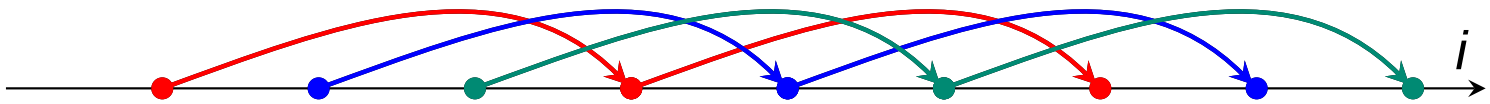


Iteration Space Slicing: Beletska et al.

Can we parallelize this code?

```
for (i = 1; i <= n; ++i)
    a[i] = f(a[i-3]);
```

Dependences:

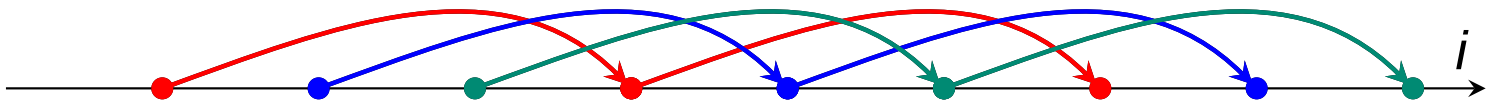


Iteration Space Slicing: Beletska et al.

Can we parallelize this code?

```
for (i = 1; i <= n; ++i)
    a[i] = f(a[i-3]);
```

Dependences:




```
for (i = 1; i <= min(n-3,3); ++i) { // parallel loop
    a[i] = f(a[i-3]);
    for (j = i + 3; j <= n; j += 3)
        a[j] = f(a[j-3]);
}
```

$\Rightarrow$ partition iterations according to dependences

$\Rightarrow$ compute transitive closure of dependences

Equivalence Checking: Barthou et al.

Are the following two programs equivalent?

```
for (i = 0; i < 10; ++i)
  b[i] = f(a[i]);
for (i = 10; i < 20; ++i)
  b[i] = f(a[i]);
c[0] = 0;
for (i = 1; i < 20; ++i)
  c[i] = b[i-1];
```

```
c[0] = 0;
b[0] = f(a[0]);
for (i = 0; i < 19; ++i) {
  i1 = i+1;
  b[i1] = f(a[i1]);
  c[i1] = b[i];
}
```

Equivalence Checking: Barthou et al.

Are the following two programs equivalent?

```
for (i = 0; i < 10; ++i)
  b[i] = f(a[i]);
for (i = 10; i < 20; ++i)
  b[i] = f(a[i]);
c[0] = 0;
for (i = 1; i < 20; ++i)
  c[i] = b[i-1];
```

```
c[0] = 0;
b[0] = f(a[0]);
for (i = 0; i < 19; ++i) {
  i1 = i+1;
  b[i1] = f(a[i1]);
  c[i1] = b[i];
}
```

1. Construct Minsky machine ($\rightsquigarrow$ cross product of dependence graphs)
   - initial state: outputs are equal
   - transition: exiting state equality requires entering state equality
   - final failure states (e.g., different function called)
   - final success states: inputs are equal

Equivalence Checking: Barthou et al.

Are the following two programs equivalent?

```
for (i = 0; i < 10; ++i)
  b[i] = f(a[i]);
for (i = 10; i < 20; ++i)
  b[i] = f(a[i]);
c[0] = 0;
for (i = 1; i < 20; ++i)
  c[i] = b[i-1];
```

```
c[0] = 0;
b[0] = f(a[0]);
for (i = 0; i < 19; ++i) {
  i1 = i+1;
  b[i1] = f(a[i1]);
  c[i1] = b[i];
}
```

1. Construct Minsky machine ($\rightsquigarrow$ cross product of dependence graphs)
   - initial state: outputs are equal
   - transition: exiting state equality requires entering state equality
   - final failure states (e.g., different function called)
   - final success states: inputs are equal
2. Derive accessibility relation from regular expression

   concatenation $\rightarrow$ composition

   branches $\rightarrow$ union

   cycles $\rightarrow$ transitive closure

Equivalence Checking: Barthou et al.

Are the following two programs equivalent?

```
for (i = 0; i < 10; ++i)
  b[i] = f(a[i]);
for (i = 10; i < 20; ++i)
  b[i] = f(a[i]);
c[0] = 0;
for (i = 1; i < 20; ++i)
  c[i] = b[i-1];
```

```
c[0] = 0;
b[0] = f(a[0]);
for (i = 0; i < 19; ++i) {
  i1 = i+1;
  b[i1] = f(a[i1]);
  c[i1] = b[i];
}
```

1. Construct Minsky machine ($\rightsquigarrow$ cross product of dependence graphs)
   - initial state: outputs are equal
   - transition: exiting state equality requires entering state equality
   - final failure states (e.g., different function called)
   - final success states: inputs are equal
2. Derive accessibility relation from regular expression

   concatenation $\rightarrow$ composition

   branches $\rightarrow$ union

   cycles $\rightarrow$ transitive closure
3. Equivalent iff $\begin{cases} \text{failure states:} & \text{empty relation} \\ \text{success states:} & \text{relation covered by equality of array indices} \end{cases}$

Outline

1. Motivating Applications
2. **Problem Formulation**
   - Quasi-affine Integer Tuple Sets and Relations
   - Powers and Transitive Closures
   - Approximation
3. Reachability Analysis
4. Technical Contribution
5. Experimental Results
6. Conclusion and Perspectives

Quasi-affine Integer Tuple Sets and Relations

Quasi-affine integer sets and relations

$$S(\mathbf{s}) = \{\, \mathbf{x} \in \mathbb{Z}^d \mid \exists \mathbf{z} \in \mathbb{Z}^e : A\mathbf{x} + B\mathbf{s} + D\mathbf{z} \geq \mathbf{c} \,\}$$

$$R(\mathbf{s}) = \{\, \mathbf{x}_1 \to \mathbf{x}_2 \in \mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2} \mid \exists \mathbf{z} \in \mathbb{Z}^e : A_1\mathbf{x}_1 + A_2\mathbf{x}_2 + B\mathbf{s} + D\mathbf{z} \geq \mathbf{c} \,\}$$

- "basic" types: "convex" sets and maps (relations)
  - affine equality + inequality constraints
  - parameters **s**
  - (optional) explicit representation of existentially quantified variables as integer divisions
- union types: sets and maps

  $\Rightarrow$ (disjoint) unions of basic sets/maps
- Note: any relation in Presburger arithmetic can be put into this form

Quasi-affine Integer Tuple Sets and Relations

Quasi-affine integer sets and relations

$$S(\mathbf{s}) = \{\, \mathbf{x} \in \mathbb{Z}^d \mid \exists \mathbf{z} \in \mathbb{Z}^e : A\mathbf{x} + B\mathbf{s} + D\mathbf{z} \geq \mathbf{c} \,\}$$

$$R(\mathbf{s}) = \{\, \mathbf{x}_1 \to \mathbf{x}_2 \in \mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2} \mid \exists \mathbf{z} \in \mathbb{Z}^e : A_1\mathbf{x}_1 + A_2\mathbf{x}_2 + B\mathbf{s} + D\mathbf{z} \geq \mathbf{c} \,\}$$

- "basic" types: "convex" sets and maps (relations)
  - affine equality + inequality constraints
  - parameters **s**
  - (optional) explicit representation of existentially quantified variables as integer divisions
- union types: sets and maps

  $\Rightarrow$ (disjoint) unions of basic sets/maps
- Note: any relation in Presburger arithmetic can be put into this form

Quasi-affine Integer Tuple Sets and Relations

Quasi-affine integer sets and relations

$$S(\mathbf{s}) = \{ \mathbf{x} \in \mathbb{Z}^d \mid \exists \mathbf{z} \in \mathbb{Z}^e : A\mathbf{x} + B\mathbf{s} + D\mathbf{z} \geq \mathbf{c} \}$$

$$R(\mathbf{s}) = \{ \mathbf{x}_1 \to \mathbf{x}_2 \in \mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2} \mid \exists \mathbf{z} \in \mathbb{Z}^e : A_1\mathbf{x}_1 + A_2\mathbf{x}_2 + B\mathbf{s} + D\mathbf{z} \geq \mathbf{c} \}$$

- "basic" types: "convex" sets and maps (relations)
  - affine equality + inequality constraints
  - parameters **s**
  - (optional) explicit representation of existentially quantified variables as integer divisions
- union types: sets and maps

  $\Rightarrow$ (disjoint) unions of basic sets/maps
- Note: any relation in Presburger arithmetic can be put into this form

Quasi-affine Integer Tuple Sets and Relations

Quasi-affine integer sets and relations

$$S(\mathbf{s}) = \{\, \mathbf{x} \in \mathbb{Z}^d \mid \exists \mathbf{z} \in \mathbb{Z}^e : A\mathbf{x} + B\mathbf{s} + D\mathbf{z} \geq \mathbf{c} \,\}$$

$$R(\mathbf{s}) = \{\, \mathbf{x}_1 \to \mathbf{x}_2 \in \mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2} \mid \exists \mathbf{z} \in \mathbb{Z}^e : A_1\mathbf{x}_1 + A_2\mathbf{x}_2 + B\mathbf{s} + D\mathbf{z} \geq \mathbf{c} \,\}$$

- "basic" types: "convex" sets and maps (relations)
  - affine equality + inequality constraints
  - parameters **s**
  - (optional) explicit representation of existentially quantified variables as integer divisions
- union types: sets and maps

  $\Rightarrow$ (disjoint) unions of basic sets/maps
- Note: any relation in Presburger arithmetic can be put into this form

Remarks

- need not be a function

Example

$$\{x \to y \mid y \geq 3 + x \ \wedge \ y \leq 4 + x\}$$

Remarks

- need not be a function

Example

$$\{x \to y \mid y \geq 3 + x \ \wedge \ y \leq 4 + x\}$$

- may involve parameters

Example

$$n \to \{(x,y) \to (1 + x, 1 - n + y) \mid n \geq 2\}$$

Remarks

- need not be a function

Example

$$\{x \to y \mid y \geq 3 + x \ \wedge \ y \leq 4 + x\}$$

- may involve parameters

Example

$$n \to \{(x, y) \to (1 + x, 1 - n + y) \mid n \geq 2\}$$

- may involve existentially quantified variables

Example

$$n \to \{x \to y \mid \exists \alpha, \beta : 7\alpha = -2 + n \ \wedge \ 5\beta = -1 - x + y \ \wedge \ y \geq 6 + x\}$$

Remarks

- need not be a function

Example

$$\{x \to y \mid y \geq 3+x \ \wedge \ y \leq 4+x\}$$

- may involve parameters

Example

$$n \to \{(x,y) \to (1+x, 1-n+y) \mid n \geq 2\}$$

- may involve existentially quantified variables

Example

$$n \to \{x \to y \mid \exists \alpha, \beta : 7\alpha = -2+n \ \wedge \ 5\beta = -1-x+y \ \wedge \ y \geq 6+x\}$$

- may involve unions

Powers

Definition (Power of a Relation)

Let $R \in \mathbb{Z}^n \to 2^{\mathbb{Z}^d \to \mathbb{Z}^d}$ be a relation and $k \in \mathbb{Z}_{\geq 1}$ a positive number, then power k of relation R is defined as

$$R^k \triangleq \begin{cases} R & \text{if } k = 1 \\ R \circ R^{k-1} & \text{if } k \geq 2 \end{cases}$$

Powers

Definition (Power of a Relation)

Let $R \in \mathbb{Z}^n \to 2^{\mathbb{Z}^d \to \mathbb{Z}^d}$ be a relation and $k \in \mathbb{Z}_{\geq 1}$ a positive number, then power k of relation R is defined as

$$R^k \triangleq \begin{cases} R & \text{if } k = 1 \\ R \circ R^{k-1} & \text{if } k \geq 2 \end{cases}$$

Example

$$R = \{x \to x + 1\}$$
$$R^k = \{x \to x + k \mid k \geq 1\}$$

Transitive Closures

Definition (Transitive Closure of a Relation)

Let $R \in \mathbb{Z}^n \to 2^{\mathbb{Z}^d \to \mathbb{Z}^d}$ be a relation, then the transitive closure R^+ of R is the union of all positive powers of R,

$$R^+ \triangleq \bigcup_{k \geq 1} R^k$$

State of the art: algorithm by Kelly et al. implemented in Omega calculator

Transitive Closures

Definition (Transitive Closure of a Relation)

Let $R \in \mathbb{Z}^n \to 2^{\mathbb{Z}^d \to \mathbb{Z}^d}$ be a relation, then the transitive closure R^+ of R is the union of all positive powers of R,

$$R^+ \triangleq \bigcup_{k \geq 1} R^k$$

State of the art: algorithm by Kelly et al. implemented in Omega calculator

Example

$$R = \{x \to x + 1\}$$
$$R^k = \{x \to x + k \mid k \geq 1\}$$
$$R^+ = \{x \to y \mid \exists k \geq 1 : y = x + k\} = \{x \to y \mid y \geq x + 1\}$$

Approximation

Fact

Given an affine relation R, the power R^k (with k a parameter) and the transitive closure R^+ may not be affine relations, or even computable

Example

$$R = \{ x \to 2x \}$$

$$R^k = \{ x \to 2^k x \}$$

Approximation

Fact

Given an affine relation R, the power R^k (with k a parameter) and the transitive closure R^+ may not be affine relations, or even computable

Example

$$R = \{ x \to 2x \}$$
$$R^k = \{ x \to 2^k x \}$$

$\Rightarrow$ we need to accept approximate results

Our target applications require overapproximations
$\Rightarrow$ compute relation $\mathcal{T}(R)$ such that $R^+ \subseteq \mathcal{T}(R)$

- $\mathcal{T}(R)$ should be as close to R^+ as possible
- we want to know when the result is exact, i.e., when $\mathcal{T}(R) = R^+$

Outline

1. Motivating Applications
2. Problem Formulation
3. **Reachability Analysis**
4. Technical Contribution
5. Experimental Results
6. Conclusion and Perspectives

Reachability Analysis using Transitive Closures

Given

- transition relation T
- initial states S_0

Set of reachable states is described by

$$T^+(S_0)$$

We compute overapproximation of transitive closure
$\Rightarrow$ overapproximation of reachable states

Transitive Closures using Reachability Analysis

- Transitive closure: relation between input and output variables
- Reachability analysis: invariants among variables at a control point

  $\Rightarrow$ introduce extra set of constant variables equal to input

Transitive Closures using Reachability Analysis

- Transitive closure: relation between input and output variables
- Reachability analysis: invariants among variables at a control point

  $\Rightarrow$ introduce extra set of constant variables equal to input

Example

$$\{x \to x + 1\}$$

Using ASPIC syntax:

```
var x,x0;
states a;
transition t0 := { from   := a; to: a;
                   action := x' = x + 1; }

Region init   := { x0 = x };
```

Transitive Closures using Reachability Analysis

- Relations instead of functions

  $\Rightarrow$ introduce extra set of variables

  $\Rightarrow$ assign arbitrary value

  (? in ASPIC; extra loop in FAST)

- Existentially quantified variables

  $\Rightarrow$ similar to handling of relations

  (number of extra variables equal to maximal number of existentially quantified variables)

- Parameters

  $\Rightarrow$ handle as constant variables

Outline

1. Motivating Applications
2. Problem Formulation
3. Reachability Analysis
4. **Technical Contribution**
   - Algorithm
   - Exactness
   - Decompositions
5. Experimental Results
6. Conclusion and Perspectives

Base Case: Convex, Non-Parametric, Affine Relation

Main ideas

1. Consider a relation as a (possibly infinite) union of translations
2. Take any sequence of steps from an element in the domain of the relation to an element in the range of the relation

(same Ancourt et al., generalization of Kelly et al. and Beletska et al.)

$$R = \{ x \to y \mid y \geq 3 + x \ \wedge \ y \leq 4 + x \}$$

- Compute offsets

$$\Delta = \{ \mathbf{y} - \mathbf{x} \mid \mathbf{x} \to \mathbf{y} \in R \} = \{ d \mid 3 \leq d \leq 4 \}$$

- Unbounded approximation of $k\,\Delta$ for $k \geq 1$
  $\Rightarrow$ multiply constant term by k and project out k

$$\Delta' = \{ d \mid \exists k \geq 1 : 3k \leq d \leq 4k \}$$

- Construct paths and intersect with domain and range

$$\{ \mathbf{x} \to \mathbf{y} \in (\mathrm{dom}\, R \times \mathrm{ran}\, R) \mid \exists \mathbf{d} \in \Delta' : \mathbf{y} = \mathbf{x} + \mathbf{d} \}$$

Parameters

Three options:

- Handle as constant variables
  $\Rightarrow$ project out parameters from Δ

Parameters

Three options:

- Handle as constant variables
  $\Rightarrow$ project out parameters from Δ
- Classify constraints
  - involving only variables $A_1\mathbf{x} + \mathbf{c}_1 \geq \mathbf{0}$
    $\Rightarrow$ base case:
    $$A_1\mathbf{x} + k\mathbf{c}_1 \geq \mathbf{0}$$
  - involving only parameters $B_2\mathbf{s} + \mathbf{c}_2 \geq \mathbf{0}$
    $\Rightarrow$ copy:
    $$B_2\mathbf{s} + \mathbf{c}_2 \geq \mathbf{0}$$
  - involving both variables and parameters $A_3\mathbf{x} + B_3\mathbf{s} + \mathbf{c}_3 \geq \mathbf{0}$
    $\Rightarrow$ copy those rows j that satisfy
    $$\Delta \cap \{\, \mathbf{y} - \mathbf{x} \mid B_{3,j}\mathbf{s} + c_{3,j} > 0 \,\} = \emptyset$$

Parameters

Three options:

- Handle as constant variables
  $\Rightarrow$ project out parameters from Δ
- Classify constraints
  - involving only variables $A_1\mathbf{x} + \mathbf{c}_1 \geq \mathbf{0}$
    $\Rightarrow$ base case:
    $$A_1\mathbf{x} + k\mathbf{c}_1 \geq \mathbf{0}$$
  - involving only parameters $B_2\mathbf{s} + \mathbf{c}_2 \geq \mathbf{0}$
    $\Rightarrow$ copy:
    $$B_2\mathbf{s} + \mathbf{c}_2 \geq \mathbf{0}$$
  - involving both variables and parameters $A_3\mathbf{x} + B_3\mathbf{s} + \mathbf{c}_3 \geq \mathbf{0}$
    $\Rightarrow$ copy those rows j that satisfy
    $$\Delta \cap \{\, \mathbf{y} - \mathbf{x} \mid B_{3,j}\mathbf{s} + c_{3,j} > 0 \,\} = \emptyset$$
- Both $\Rightarrow$ intersection

Parameters Example

Input relation

$$R = n \to \{ (x,y) \to (1+x, 1-n+y) \mid n \geq 2 \}$$

- Handle as constant variables

$$n \to \{ (x,y) \to (x',y') \mid n \geq 2 \ \wedge \ x' \geq 1+x \ \wedge \ y' \leq 1-n+y \}$$

- Classify constraints

$$n \to \{ (x,y) \to (x',y') \mid n \geq 2 \ \wedge \ x' \geq 1+x \ \wedge \ y' \leq x+y-x' \}$$

- Both $\Rightarrow$ intersection

$$n \to \{ (x,y) \to (x',y') \mid n \geq 2 \ \wedge \ x' \geq 1+x \ \wedge \ y' \leq 1-n+y \ \wedge \ y' \leq x+y-x' \}$$

Existentially Quantified Variables

1. Compute unique representation in terms of variables and parameters
   $\Rightarrow$ parametric integer linear programming (PIP)
2. Classify
   - definition only involves variables
     $\Rightarrow$ treat as variable
   - definition only involves parameters
     $\Rightarrow$ treat as parameter
   - definition involves both
     $\Rightarrow$ ignore any constraint involving the existentially quantified variable

Unions

- Compute $R^+ = \left(\bigcup_i R_i \right)^+$

$$\Delta R \stackrel{\triangle}{=} \bigcup_i \Delta_i$$

- Handle each Δ_i separately (addition is commutative)

$$P = ((P_m \cup \mathrm{Id}) \circ \cdots \circ (P_1 \cup \mathrm{Id})) \cap \{ \mathbf{x} \to \mathbf{y} \mid k = \sum_i k_i > 0 \}$$

with

$$P_i = \{ \mathbf{x} \to \mathbf{y} \mid \exists \mathbf{d} \in \Delta_i' : \mathbf{y} = \mathbf{x} + \mathbf{d} \}$$

- Compute

$$\mathcal{T}(R) = P \cap (\mathrm{dom}\, R \times \mathrm{ran}\, R)$$

Exactness

Is approximation $R^+ \subseteq \mathcal{T}(R)$ exact, i.e., is $\mathcal{T}(R) = R^+$?

Exactness

Is approximation $R^+ \subseteq \mathcal{T}(R)$ exact, i.e., is $\mathcal{T}(R) = R^+$?

- Acyclic relations: observe that

$$R^+ \stackrel{\triangle}{=} R \cup \left(R \circ R^+\right)$$

Unique fixed point for acyclic R

$\Rightarrow$ check $\mathcal{T}(R) \subseteq R \cup (R \circ \mathcal{T}(R))$

(check acyclicity of $\mathcal{T}(R)$ for a sufficient condition on acyclicity of R^+)

Exactness

Is approximation $R^+ \subseteq \mathcal{T}(R)$ exact, i.e., is $\mathcal{T}(R) = R^+$?

- Acyclic relations: observe that

$$R^+ \stackrel{\triangle}{=} R \cup \left(R \circ R^+\right)$$

  Unique fixed point for acyclic R

  $\Rightarrow$ check $\mathcal{T}(R) \subseteq R \cup (R \circ \mathcal{T}(R))$

  (check acyclicity of $\mathcal{T}(R)$ for a sufficient condition on acyclicity of R^+)

- Possibly cyclic relations

  $\Rightarrow$ check exactness of approximation $\mathcal{P}_k(R)$ of power R^k

$$\begin{aligned} \mathcal{P}_k(R) &\subseteq R && \text{for } k = 1 \\ \mathcal{P}_k(R) &\subseteq R \circ \mathcal{P}_{k-1}(R) && \text{for } k \geq 2 \end{aligned}$$

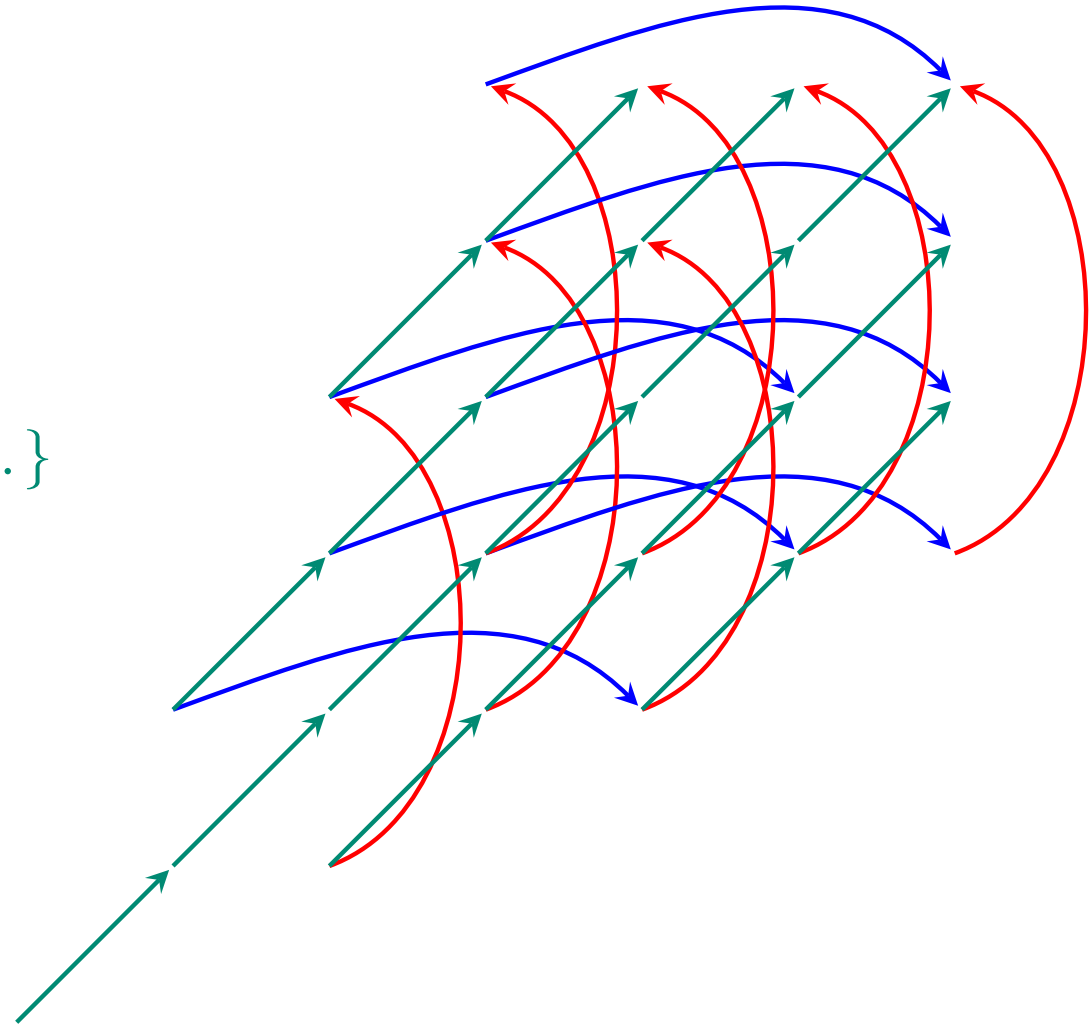
  Note: power exact $\Rightarrow$ transitive closure exact

  (but not the other way around)

Exactness Example

$$R = R_1 \cup R_2 \cup R_3$$

$$R_1 = \{ (i,j) \to (i+3,j) \mid \ldots \}$$
$$R_2 = \{ (i,j) \to (i,j+3) \mid \ldots \}$$
$$R_3 = \{ (i,j) \to (i+1,j+1) \mid \ldots \}$$



Exactness Example

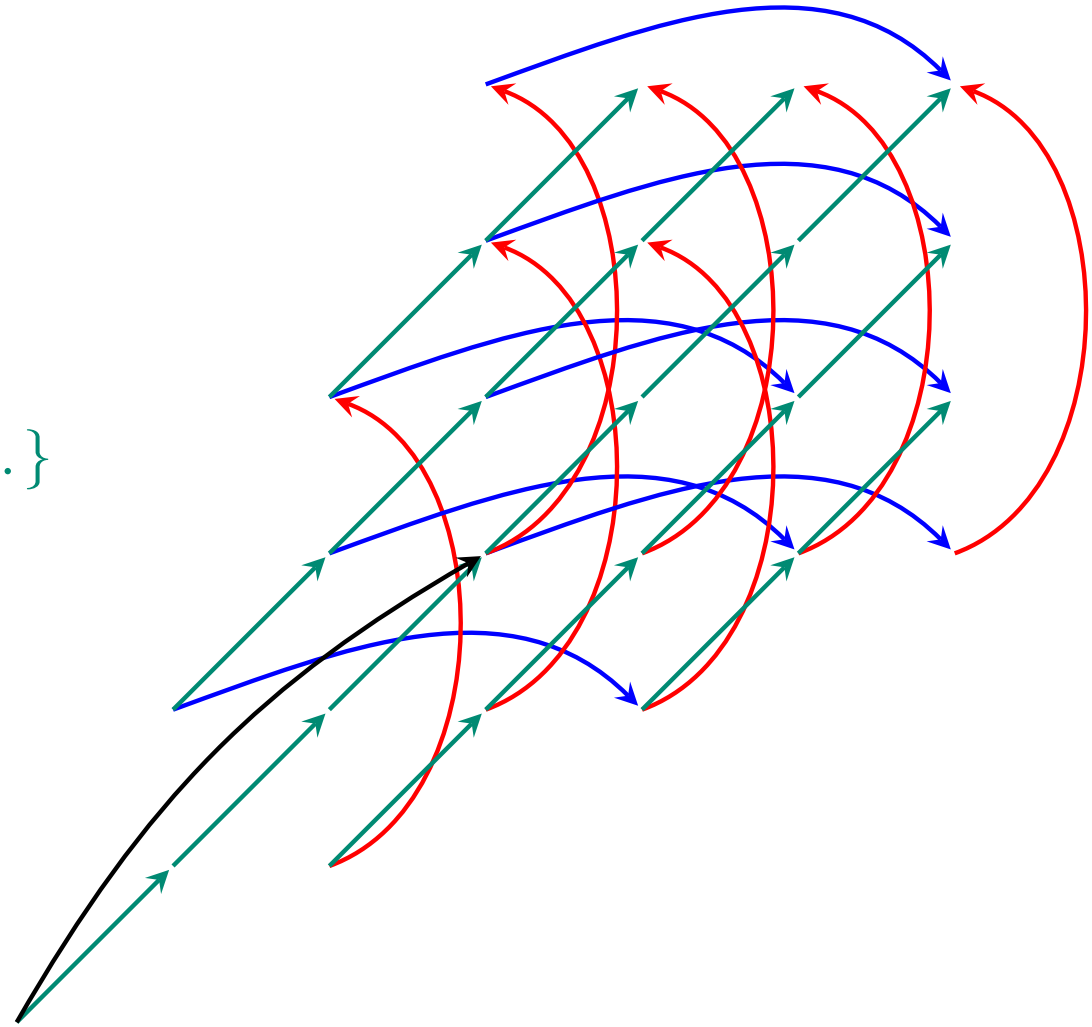
$R = R_1 \cup R_2 \cup R_3$

$R_1 = \{ (i,j) \to (i+3,j) \mid \ldots \}$

$R_2 = \{ (i,j) \to (i,j+3) \mid \ldots \}$

$R_3 = \{ (i,j) \to (i+1,j+1) \mid \ldots \}$

$(1,1) \to (4,4)$ in $\mathcal{P}_2(R)$, but not in R^2



Exactness Example

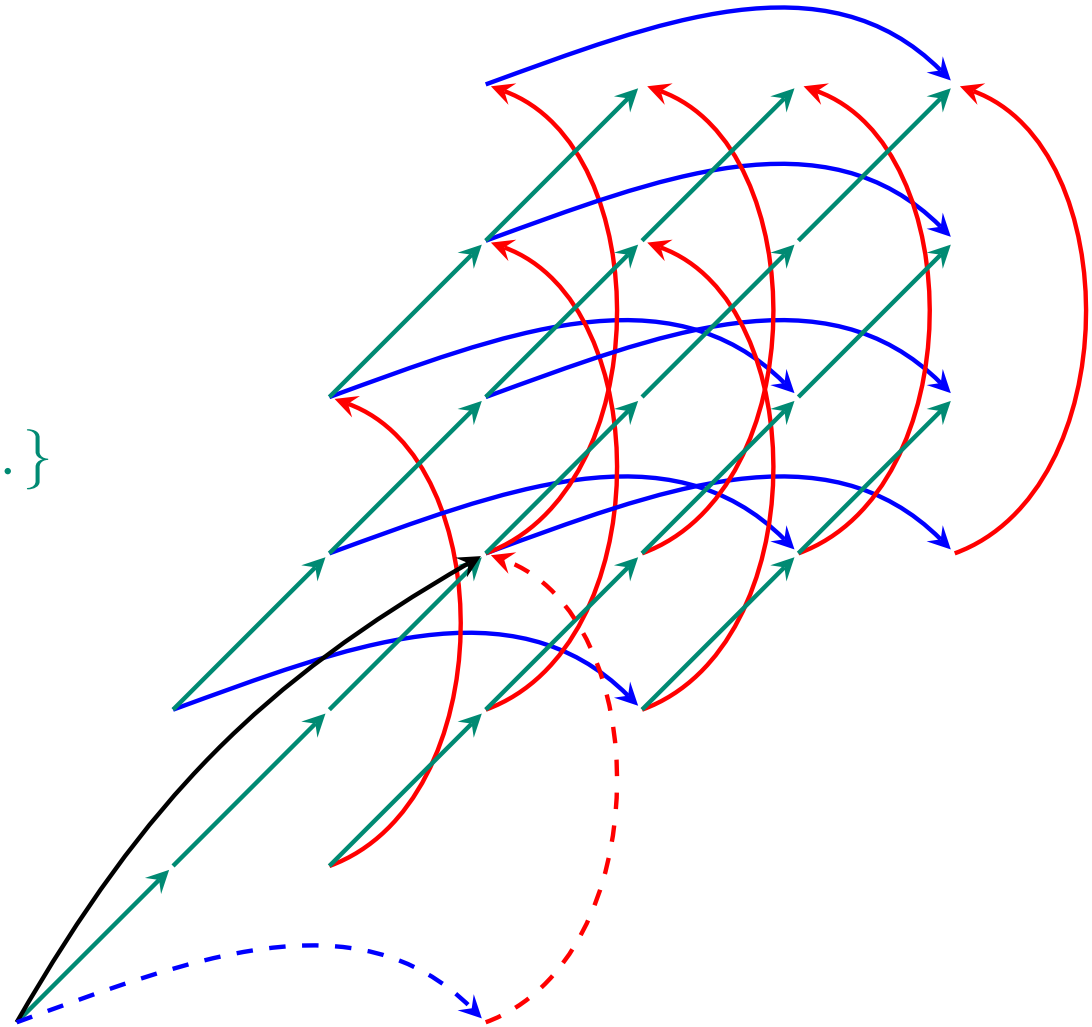
$R = R_1 \cup R_2 \cup R_3$

$R_1 = \{ (i,j) \to (i+3,j) \mid \ldots \}$

$R_2 = \{ (i,j) \to (i,j+3) \mid \ldots \}$

$R_3 = \{ (i,j) \to (i+1,j+1) \mid \ldots \}$

$(1,1) \to (4,4)$ in $\mathcal{P}_2(R)$, but not in R^2



Exactness Example

$R = R_1 \cup R_2 \cup R_3$

$R_1 = \{(i,j) \to (i+3,j) \mid \ldots\}$

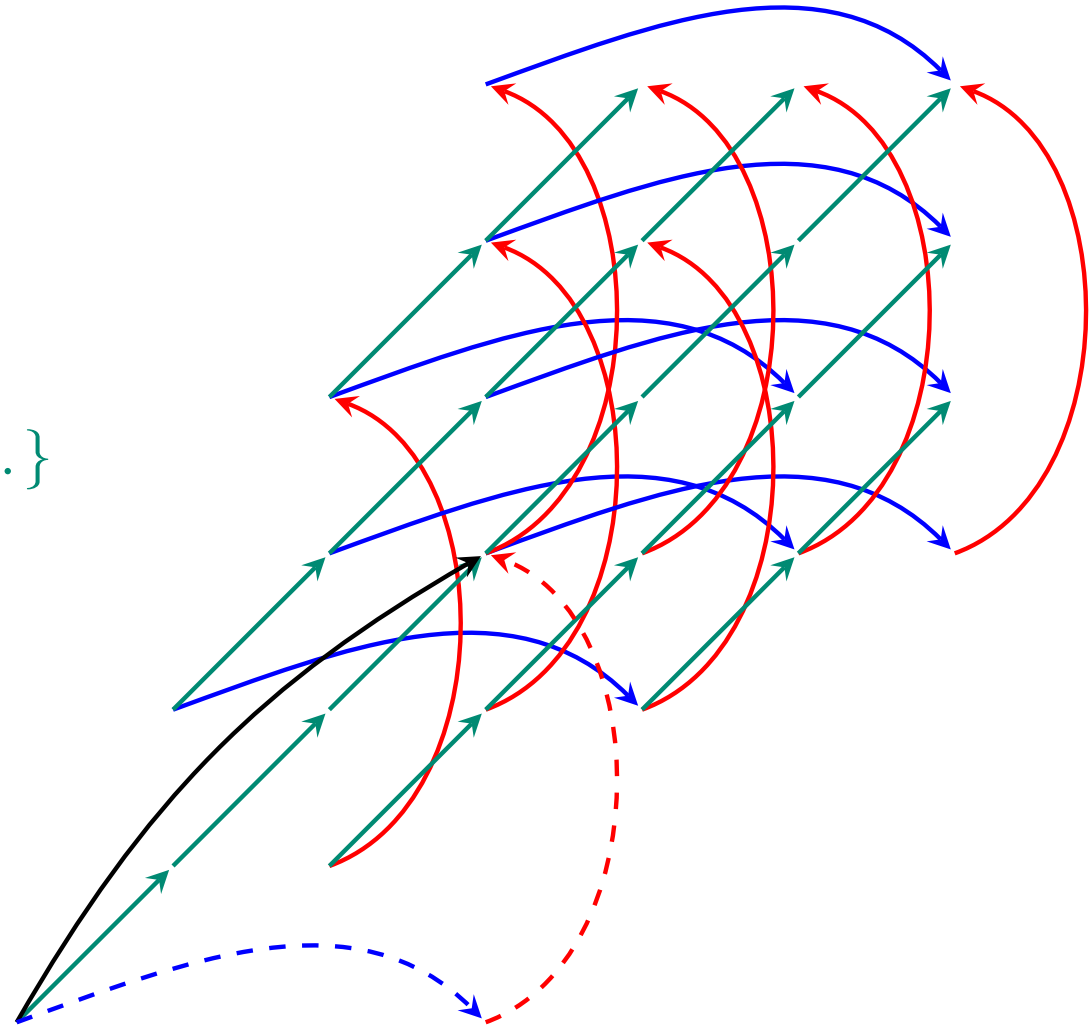
$R_2 = \{(i,j) \to (i,j+3) \mid \ldots\}$

$R_3 = \{(i,j) \to (i+1,j+1) \mid \ldots\}$

$(1,1) \to (4,4)$ in $\mathcal{P}_2(R)$, but not in R^2

However $(1,1) \to (4,4)$ in R^3

In fact, $\mathcal{T}(R) = R^+$



Decompositions

Decompositions can improve accuracy

- Incremental computation

$$R = \bigcup_j R_j$$

compute R^+ as

$$R^+ = R_i^+ \cup \left(\bigcup_{j \neq i} R_i^* \circ R_j \circ R_i^* \right)^+$$

Similar to Kelly et al., but different profitability heuristic

- Strongly connected components
- Domain partitioning
  $\Rightarrow$ apply Kelly et al.'s modified Floyd-Warshall algorithm

Outline

1. Motivating Applications
2. Problem Formulation
3. Reachability Analysis
4. Technical Contribution
5. **Experimental Results**
6. Conclusion and Perspectives

Implementation

Integer Set Library

Our method is implemented in ISL, distributed under the terms of the LGPL
`http://freshmeat.net/projects/isl`

See the project's page for a more detailed description of the functionalities and internals of ISL

We performed systematic comparisons with state-of-the-art tools:

OMEGA+: Kelly et al., 1996; new version by Chen, 2009

STING: Sankaranarayanan et al., 2004

ASPIC: Gonnord et al., 2006

FAST: Leroux et al., 2008

Experimental Results

Equivalence checking

	Omega+				
	ISL	TC	AC	FAST	ASPIC	STING
exact	472	366	267	139	201	215
inexact	67	157	266	0	268	240
failure	34	50	40	434	104	118

Experimental Results

Iteration Space Slicing

			Omega+				
		ISL	TC	AC	FAST	ASPIC	STING
mem	exact	70	58	43	25	15	39
	inexact	7	11	50	0	87	22
	failure	57	65	41	109	32	73
val	exact	72	57	43	28	37	39
	inexact	2	26	56	0	41	22
	failure	60	51	35	106	56	73

Experimental Results

Reachability Analysis

From ASPIC web site: 21 relations

ISL fails	ASPIC fails	ISL wins	ASPIC wins	identical	uncomparable
0 (2)	1	7	3	2	6

From LEVER test cases (safety analysis problems):

ISL only succeeds in 7 out of 32 relations

Outline

1. Motivating Applications
2. Problem Formulation
3. Reachability Analysis
4. Technical Contribution
5. Experimental Results
6. **Conclusion and Perspectives**

Conclusion and Perspectives

Conclusion

- Novel algorithm to compute the overapproximation of the transitive closure of a Presburger relation
- Significantly more accurate and faster than previous methods

Perspectives

- Characterize the differences between closure and reachability
- Design a more robust heuristic for both closure and reachability
- Better handling of cyclic graphs