Termination and Cost Analysis: Complexity and Precision Issues

Md. Abu Naser Masud
February 6, 2013

Advisors: Samir Genaim, Germán Puebla

Doctoral Dissertation Defense
in
School of Computer Science,
Technical University of Madrid (Spain)
1. Introduction

2. Background on Cost Analysis

3. Precise Cost Analysis Techniques

4. Theoretical Complexity of Deciding Termination

5. Conclusions
The aim of **COST ANALYSIS** is to estimate the bound of resource consumption (aka cost) of executing a given program $P$ on a given input data.
What is Cost Analysis?

The aim of **COST ANALYSIS** is to estimate the bound of resource consumption (aka cost) of executing a given program $P$ on a given input data without actually executing $P$.
What is Cost Analysis?

The aim of **COST ANALYSIS** is to estimate the bound of resource consumption (aka cost) of executing a given program $P$ on a given input data without actually executing $P$. 

\text{\textcolor{red}{\textbackslash static}}
What is Cost Analysis?

The aim of COST ANALYSIS is to estimate the bound of resource consumption (aka cost) of executing a given program $P$ on any given input data without actually executing $P$. 
What is Cost Analysis?

The aim of COST ANALYSIS is to estimate the bound of resource consumption (aka cost) of executing a given program $P$ on a given input data without actually executing $P$. 
What is Cost Analysis?

The aim of **COST ANALYSIS** is to estimate the bound of resource consumption (aka cost) of executing a given program $P$ on any given input data without actually executing $P$

- Upper Bounds (**worst case**)
- Lower Bounds (**best case**)

Upper Bounds (**worst case**)

Lower Bounds (**best case**)
What is Cost Analysis?

The aim of COST ANALYSIS is to estimate the bound of resource consumption (aka cost) of executing a given program $P$ on any input data without actually executing $P$.

- Upper Bounds (worst case)
- Lower Bounds (best case)
- Non-Asymptotic: $P(x) = 2 + 3 \cdot x + 2 \cdot x^2$
- Asymptotic: $P(x) = O(x^2)$
What is Cost Analysis?

The aim of COST ANALYSIS is to estimate the bound of resource consumption (aka cost) of executing any given program $P$ on any given input data without actually executing $P$. 
What is Cost Analysis?

The aim of **cost analysis** is to estimate the bound of resource consumption (aka cost) of executing a given program $P$ on a given input data without actually executing $P$.

- Execution steps
- Visits to specific program points
- Memory (possibly with garbage collection)
Phases in Cost Analysis

First Phase

- Program P
- Static Analysis
- Cost Model

Second Phase

- Cost Relations
- Cost Relations Solver
- Worst/Best Case
Phases in Cost Analysis

First Phase:
- Program P
- Static Analysis
- Abstraction of P

Second Phase:
- Cost Model
- Cost Relations
- Cost Relations Solver
- Worst/Best Case
Phases in Cost Analysis

Program P → Static Analysis → Cost Relations → Cost Relations Solver → Worst/Best Case

First Phase

Second Phase

Quality of the solution is affected by the

- **Precision** issue in the first phase AND
- **precision-applicability** tradeoff in the second phase
CRs are abstract programs and there is no unified terminology or syntax of these abstract programs in cost analysis.
CRs are abstract programs and there is no unified terminology or syntax of these abstract programs in cost analysis.

Some abstract programs are:
1. less expressive
2. easy to solve precisely with existing tools
3. but the applicability is limited

Example - recurrence relations which are solved using symbolic computations.

Some others are more expressive but require complex analysis to solve.

1. Less precise techniques are widely applicable
2. More precise techniques are less applicable

Example - cost relations which are solved using static analysis.
CRs are abstract programs and there is no unified terminology or syntax of these abstract programs in cost analysis.

Some abstract programs are:
1. less expressive
2. easy to solve precisely with existing tools
3. but the applicability is limited
4. Example - recurrence relations which are solved using symbolic computations

\[
P(0) = 0 \\
P(x) = x + P(x - 1)
\]
CRs are abstract programs and there is no unified terminology or syntax of these abstract programs in cost analysis.

Some abstract programs are
1. less expressive
2. easy to solve precisely with existing tools
3. but the applicability is limited
4. Example - recurrence relations which are solved using symbolic computations

Some others are more expressive but require complex analysis to solve.
1. Less precise techniques are widely applicable
2. More precise techniques are less applicable
CRs are abstract programs and there is no unified terminology or syntax of these abstract programs in cost analysis.

Some abstract programs are:
1. less expressive
2. easy to solve precisely with existing tools
3. but the applicability is limited
4. Example - recurrence relations which are solved using symbolic computations

Some others are more expressive but require complex analysis to solve.
1. Less precise techniques are widely applicable
2. More precise techniques are less applicable
3. Example - cost relations which are solved using static analysis

\[
\begin{align*}
P(x, y) &= 0 & \{x = 0, y = 0\} \\
P(x, y) &= x + P(x', y') & \{x > 0, x' < x, y' = y\}
\end{align*}
\]
Theoretical interest lies in understanding the complexity of cost analysis. That means understanding the degree of solvability of inferring resource bounds for some class of programs.
Complexity Issues

- Theoretical interest lies in understanding the complexity of cost analysis. That means understanding the degree of solvability of inferring resource bounds for some class of programs.

- Computing bounds from CRs require solving termination problems of simple loops.
Complexity Issues

- Theoretical interest lies in understanding the complexity of cost analysis. That means understanding the degree of solvability of inferring resource bounds for some class of programs.

- Computing bounds from CRs require solving termination problems.

  Cost analysis requires inferring the bound on loop iterations or recursion depths.
Theoretical interest lies in understanding the complexity of cost analysis. That means understanding the degree of solvability of inferring resource bounds for some class of programs.

Computing bounds from CRs require solving termination problems. Existence or Witness of such bound proves termination.
**Complexity Issues**

- Theoretical interest lies in understanding the complexity of cost analysis. That means understanding the degree of solvability of inferring resource bounds for some class of programs.

- Computing bounds from CRs require solving termination problems of simple loops.

- Termination analysis is often a subtask of cost analysis.
Complexity Issues

- Theoretical interest lies in understanding the complexity of cost analysis. That means understanding the degree of solvability of inferring resource bounds for some class of programs.

- Computing bounds from CRs require solving termination problems of simple loops.

- Termination analysis is often a subtask of cost analysis.

- Theoretical limits of cost analysis are inherited from the limits of termination analysis.
Objectives

1. Compute upper bounds from CRs that are very precise and the approach is widely applicable and scalable.
OBJECTIVES

1. Compute upper bounds from CRs that are very precise and the approach is widely applicable and scalable.

2. Extend the approach of computing upper bounds to computing nontrivial, precise lower bounds.
**Objectives**

1. Compute upper bounds from CRs that are very precise and the approach is widely applicable and scalable.

2. Extend the approach of computing upper bounds to computing nontrivial, precise lower bounds.

3. Obtain decidability and complexity results on the termination of simple loops that arise in the context of cost analysis.
OBJECTIVES

1. Compute upper bounds from CRs that are very precise and the approach is widely applicable and scalable.

2. Extend the approach of computing upper bounds to computing nontrivial, precise lower bounds.

3. Obtain decidability and complexity results on the termination of simple loops that arise in the context of cost analysis.

4. Understand the consequences of the complexity results for termination analysis to cost analysis.
Outline

1. Introduction
2. Background on Cost Analysis
3. Precise Cost Analysis Techniques
4. Theoretical Complexity of Deciding Termination
5. Conclusions
```java
void f(int n) {
    List l = null;
    int i=0;
    while ( i<n ) {
        int j=0;
        while ( j<i ) {
            int k =0;
            for(int k=0;k<n+j;k++)
                l=new List(i*k*j,l);
            j+= (*) ? 1 : 3;
        }
        i+= (*) ? 2 : 4;
    }
}
```
void f(int n) {
    List l = null;
    int i=0;
    while ( i<n ) {
        int j=0;
        while ( j<i ) {
            for(int k=0;k<n+j;k++)
                l = new List(i*k*j,l);
            j+= (*) ? 1 : 3;
        }
        i+= (*) ? 2 : 4;
    }
}
void f(int n) {
    List l = null;
    int i = 0;
    while (i < n) {
        int j = 0;
        while (j < i) {
            for (int k = 0; k < n + j; k++)
                l = new List(i * k * j, l);
            j += (*) ? 1 : 3;
        }
        i += (*) ? 2 : 4;
    }
}

\textbf{Worst-Case (UB)}
\[ n_0 + j_0 - k_0 \]

\textbf{Best-Case (LB)}
\[ n_0 + j_0 - k_0 \]
void f(int n) {
    List l = null;
    int i=0;
    while ( i<n ) {
        int j=0;
        while ( j<i ) {
            for (int k=0; k<n+j; k++)
                l = new List(i*k*j, l);
            j+= (*) ? 1 : 3;
        }
        i+= (*) ? 2 : 4;
    }
}

Worst-Case (UB)

\[ n_0 + j_0 - k_0 \]

\[ (i_0 - j_0) \times (n_0 + i_0 - 1) \]

Best-Case (LB)

\[ n_0 + j_0 - k_0 \]

\[ \frac{i_0 - j_0}{3} \times (n_0 + i_0 - 1) \]
void \texttt{f}(int \ n) \{
    \textbf{List} \ l = \textbf{null};
    \textbf{int} \ i = 0;
    \textbf{while} ( i < n ) \{
        \textbf{int} \ j = 0;
        \textbf{while} ( j < i ) \{
            \textbf{for} ( \textbf{int} \ k = 0; k < n+j; k++ )
            \quad \textbf{l} = \textbf{new} \ \textbf{List}(i*\textbf{k}^*j,\textbf{l});
            \quad j += (\textbf{(*)}) \ ? \ 1 : 3;
        \}
        \quad i += (\textbf{(*)}) \ ? \ 2 : 4;
    \}
\}

\textbf{Worst-Case (UB)}
\begin{align*}
\frac{n_0+j_0-k_0}{(i_0-j_0)^* (n_0+i_0-1)}
\frac{n_0-i_0}{2}*(n_0-1)*(2n_0-2)
\end{align*}

\textbf{Best-Case (LB)}
\begin{align*}
\frac{n_0+j_0-k_0}{(i_0-j_0)^* (n_0+i_0-1)}
\frac{i_0-j_0}{3}*(n_0+i_0-1)
\frac{n_0-i_0}{4}\frac{n_0-1}{3}*(2n_0-2)
\end{align*}
COST ANALYSIS - WEGBREIT 1975

Program

Static Analysis

Cost Relations

CRs Solver

Best/Worst Case
while ( i<n ) {
    int j=0;
    while ( j<i ) {
        for(int k=0; k<n+j; k++)
            l=new List(i*k*j, l);
        j+= (*) ? 1:3;
    }
    i+= (*) ? 2:4;
}
while ( i<n ) {
    int j=0;
    while ( j<i ) {
        for(int k=0;k<n+j;k++)
            l=new List(i*k*j,l);
        j+= (*) ? 1:3;
    }
    i+= (*) ? 2:4;
}
while ( i<n ) {
    int j=0;
    while ( j<i ) {
        for(int k=0; k<n+j; k++)
            l=new List(i*k*j, l);
        j+= (*) ? 1:3;
    }
    i+= (*) ? 2:4;
}

\[
A(i, n) = 0 \quad \{ i \geq n \} \\
A(i, n) = B(0, i, n) + A(i', n) \quad \{ i+1 \leq n, i+2 \leq i' \leq i+4 \}
\]

\[
B(j, i, n) = 0 \quad \{ j \geq i \} \\
B(j, i, n) = C(0, j, n) + B(j', i, n) \quad \{ j+1 \leq i, j+1 \leq j' \leq j+3 \}
\]

\[
C(k, j, n) = 0 \quad \{ k \geq n+j \} \\
C(k, j, n) = 1+C(k', j, n) \quad \{ k' = k+1, k+1 \leq n+j \}
\]
Program

while ( i<n ) {
    int j=0;
    while ( j<i ) {
        for(int k=0; k<n+j; k++)
            l=new List(i*k*j, l);
        j+= (*) ? 1:3;
    }
    i+= (*) ? 2:4;
}

A(i, n) = 0 \{i\geq n\}
A(i, n) = B(0, i, n)+A(i', n) \{i+1\leq n, i+2\leq i'\leq i+4\}

B(j, i, n) = 0 \{j\geq i\}
B(j, i, n) = C(0, j, n)+B(j', i, n) \{j+1\leq i, j+1\leq j'\leq j+3\}

C(k, j, n) = 0 \{k\geq n+j\}
C(k, j, n) = 1+C(k', j, n) \{k' = k+1, k+1\leq n+j\}
Program

Static Analysis

Cost Relations

CRs Solver

Best/Worst Case

while ( i<n ) {
    int j=0;
    while ( j<i ) {
        for(int k=0; k<n+j; k++)
            l=new List(i*k*j, l);
        j+= (*) ? 1:3;
    }
    i+= (*) ? 2:4;
}

\[ A(i, n) = 0 \quad \{i \geq n\} \]

\[ A(i, n) = B(0, i, n) + A(i', n) \quad \{i+1 \leq n, i+2 \leq i' \leq i+4\} \]

\[ B(j, i, n) = 0 \quad \{j \geq i\} \]

\[ B(j, i, n) = C(0, j, n) + B(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\} \]

\[ C(k, j, n) = 0 \quad \{k \geq n+j\} \]

\[ C(k, j, n) = 1+C(k', j, n) \quad \{k' = k+1, k+1 \leq n+j\} \]
while ( i<n ) {
    int j=0;
    while ( j<i ) {
        for(int k=0; k<n+j; k++)
            l=new List(i*k*j, l);
        j+= (*) ? 1:3;
    }
    i+= (*) ? 2:4;
}

A(i, n) = 0 \quad \{i \geq n\}
A(i, n) = B(0, i, n) + A(i', n) \quad \{i+1 \leq n, i+2 \leq i' \leq i+4\}

B(j, i, n) = 0 \quad \{j \geq i\}
B(j, i, n) = C(0, j, n) + B(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\}

C(k, j, n) = 0 \quad \{k \geq n+j\}
C(k, j, n) = 1 + C(k', j, n) \quad \{k' = k+1, k+1 \leq n+j\}
Program → Static Analysis → Cost Relations → CRs Solver → Best/Worst Case

\[
\text{while ( i<n )} \{
\begin{align*}
\text{int } j &= 0; \\
\text{while ( j<i )} \{
\begin{align*}
&\text{for (int } k=0; k<n+j; k++) \\
&\quad l = \text{new List}(i*k*j, l); \\
&\quad j += (\star) ? 1:3;
\end{align*}
\end{align*}
\} \\
&\quad i += (\star) ? 2:4;
\}
\]

\[
A(i, n) = \begin{cases} 
0 & \{i \geq n\} \\
B(0, i, n) + A(i', n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\}
\end{cases}
\]

\[
B(j, i, n) = \begin{cases} 
0 & \{j \geq i\} \\
C(0, j, n) + B(j', i, n) & \{j+1 \leq i, j+1 \leq j' \leq j+3\}
\end{cases}
\]

\[
C(k, j, n) = \begin{cases} 
0 & \{k \geq n+j\} \\
1 + C(k', j, n) & \{k' = k+1, k+1 \leq n+j\}
\end{cases}
\]
Program

Static Analysis

Cost Relations

CRs Solver

Best/Worst Case

\[
\begin{align*}
\text{while} \ ( i < n ) \ {\{} \\
\quad \text{int} \ j = 0; \\
\quad \text{while} \ ( j < i ) \ {\{} \\
\quad \quad \text{for} \ ( \text{int} \ k = 0; k < n+j; k++) \\
\quad \quad \quad l = \text{new List}(i*k*j, l); \\
\quad \quad j += (*) \ ? \ 1:3; \\
\quad \}\} \\
\quad i += (*) \ ? \ 2:4; \\
\} \\
\end{align*}
\]

\[
A(i, n) = 0 \quad \{i \geq n\}
\]

\[
A(i, n) = B(0, i, n) + A(i', n) \quad \{i+1 \leq n, i+2 \leq i' \leq i+4\}
\]

\[
B(j, i, n) = 0 \quad \{j \geq i\}
\]

\[
B(j, i, n) = C(0, j, n) + B(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\}
\]

\[
C(k, j, n) = 0 \quad \{k \geq n+j\}
\]

\[
C(k, j, n) = 1 + C(k', j, n) \quad \{k' = k+1, k+1 \leq n+j\}
\]
Program

Static Analysis

Cost Relations

CRs Solver

Best/Worst Case

while ( i<n ) {
    int j=0;
    while ( j<i ) {
        for (int k=0; k<n+j; k++)
            l=new List(i*k*j, l);
        j+=(*) ? 1:3;
    }
    i+=(*) ? 2:4;
}

\[
A(i, n) = \begin{cases} 
0 & \text{if } i \geq n \\
B(0, i, n) + A(i', n) & \text{if } i+1 \leq n, i+2 \leq i' \leq i+4 
\end{cases}
\]

\[
B(j, i, n) = \begin{cases} 
0 & \text{if } j \geq i \\
C(0, j, n) + B(j', i, n) & \text{if } j+1 \leq i, j+1 \leq j' \leq j+3 
\end{cases}
\]

\[
C(k, j, n) = \begin{cases} 
0 & \text{if } k \geq n+j \\
1 + C(k', j, n) & \text{if } k' = k+1, k+1 \leq n+j 
\end{cases}
\]
while ( i<n ) {
    int j=0;
    while ( j<i ) {
        for (int k=0;k<n+j;k++)
            l=new List(i*k*j,l);
        j+= (*) ? 1:3;
    }
    i+= (*) ? 2:4;
}

\[ A(i, n) = \begin{cases} 0 & \text{if } i \geq n \\ B(0, i, n) + A(i', n) & \text{otherwise} \end{cases} \]

\[ \{i+1 \leq n, i+2 \leq i' \leq i+4\} \]

\[ B(j, i, n) = \begin{cases} 0 & \text{if } j \geq i \\ C(0, j, n) + B(j', i, n) & \text{otherwise} \end{cases} \]

\[ \{j+1 \leq i, j+1 \leq j' \leq j+3\} \]

\[ C(k, j, n) = \begin{cases} 0 & \text{if } k \geq n+j \\ 1+C(k', j, n) & \text{otherwise} \end{cases} \]

\[ \{k' = k+1, k+1 \leq n+j\} \]
Program

Static Analysis

Cost Relations

CRs Solver

Best/Worst Case

\[
\text{while ( } i < n \text{ ) }
\]
\[
\text{int } j = 0;
\]
\[
\text{while ( } j < i \text{ ) }
\]
\[
\text{for (int } k = 0; k < n + j; k++)
\]
\[
\text{1 = new List(} i \times k \times j, 1); \]
\[
j += (\text{(*) } ? 1:3);
\]
\[
i += (\text{(*) } ? 2:4);
\]

\[
A(i, n) = 0 \quad \{i \ge n\}
\]
\[
A(i, n) = B(0, i, n) + A(i', n) \quad \{i + 1 \le n, i + 2 \le i' \le i + 4\}
\]

\[
B(j, i, n) = 0 \quad \{j \ge i\}
\]
\[
B(j, i, n) = C(0, j, n) + B(j', i, n) \quad \{j + 1 \le i, j + 1 \le j' \le j + 3\}
\]

\[
C(k, j, n) = 0 \quad \{k \ge n + j\}
\]
\[
C(k, j, n) = 1 + C(k', j, n) \quad \{k' = k + 1, k + 1 \le n + j\}
\]
Program

while ( i<n ) {
    int j=0;
    while ( j<i ) {
        for (int k=0; k<n+j; k++)
            l = new List(i*k*j, l);
        j+= (*) ? 1:3;
    }
    i+= (*) ? 2:4;
}

A(i, n) = 0 \{i\geq n\}
A(i, n) = B(0, i, n) + A(i', n) \{i+1\leq n, i+2\leq i'\leq i+4\}

B(j, i, n) = 0 \{j\geq i\}
B(j, i, n) = C(0, j, n) + B(j', i, n) \{j+1\leq i, j+1\leq j'\leq j+3\}

C(k, j, n) = 0 \{k\geq n+j\}
C(k, j, n) = 1+C(k', j, n) \{k' = k+1, k+1\leq n+j\}
Program

Static Analysis

Cost Relations

CRs Solver

Best/Worst Case

while ( i < n ) {
    int j = 0;
    while ( j < i ) {
        for (int k = 0; k < n + j; k++)
            l = new List(i * k * j, l);
        j += (*) ? 1 : 3;
    }
    i += (*) ? 2 : 4;
}

\[
A(i, n) = \begin{cases}
0 & \text{if } i \geq n \\
B(0, i, n) + A(i', n) & \text{if } i + 1 \leq n, i + 2 \leq i' \leq i + 4
\end{cases}
\]

\[
B(j, i, n) = \begin{cases}
0 & \text{if } j \geq i \\
C(0, j, n) + B(j', i, n) & \text{if } j + 1 \leq i, j + 1 \leq j' \leq j + 3
\end{cases}
\]

\[
C(k, j, n) = \begin{cases}
0 & \text{if } k \geq n + j \\
1 + C(k', j, n) & \text{if } k' = k + 1, k + 1 \leq n + j
\end{cases}
\]
Program

Static Analysis

Cost Relations

CRs Solver

Best/Worst Case

while ( i<n ) {
  int j=0;
  while ( j<i ) {
    for (int k=0; k<n+j; k++)
      l=new List(i*k*j,l);
    j+= (*) ? 1:3;
  }
  i+= (*) ? 2:4;
}

\[
A(i, n) = 0 \quad \{i \geq n\}
\]
\[
A(i, n) = B(0, i, n) + A(i', n) \quad \{i+1 \leq n, i+2 \leq i' \leq i+4\}
\]
\[
B(j, i, n) = 0 \quad \{j \geq i\}
\]
\[
B(j, i, n) = C(0, j, n) + B(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\}
\]
\[
C(k, j, n) = 0 \quad \{k \geq n+j\}
\]
\[
C(k, j, n) = 1 + C(k', j, n) \quad \{k' = k+1, k+1 \leq n+j\}
\]
Program

Static Analysis

Cost Relations

CRs Solver

Best/Worst Case

while ( i<n ) {
    int j=0;
    while ( j<i ) {
        for(int k=0;k<n+j;k++)
            l=new List(i*k*j,l);
        j+= (*) ? 1:3;
    }
    i+= (*) ? 2:4;
}

A(i, n) = 0 \quad \{i>n\}
A(i, n) = B(0, i, n)+A(i', n) \quad \{i+1\leq n, i+2\leq i'\leq i+4\}

B(j, i, n) = 0 \quad \{j\geq i\}
B(j, i, n) = C(0, j, n)+B(j', i, n) \quad \{j+1\leq i, j+1\leq j'\leq j+3\}

C(k, j, n) = 0 \quad \{k\geq n+j\}
C(k, j, n) = 1+C(k', j, n) \quad \{k' = k+1, k+1\leq n+j\}
while ( i<n ) {
    int j=0;
    while ( j<i ) {
        for(int k=0;k<n+j;k++)
            l=new List(i*k*j,l);
        j+= (*) ? 1:3;
    }
    i+= (*) ? 2:4;
}

A(i, n) = 0 \quad \{i\geq n\}
A(i, n) = B(0, i, n)+A(i', n) \quad \{i+1\leq n, i+2\leq i'\leq i+4\}

B(j, i, n) = 0 \quad \{j\geq i\}
B(j, i, n) = C(0, j, n)+B(j', i, n) \quad \{j+1\leq i, j+1\leq j'\leq j+3\}

C(k, j, n) = 0 \quad \{k\geq n+j\}
C(k, j, n) = 1+C(k', j, n) \quad \{k' = k+1, k+1\leq n+j\}
Program → Static Analysis → Cost Relations → CRs Solver → Best/Worst Case

Worst-Case (UB)

\[
\begin{align*}
C(k_0, j_0, n_0) &= n_0 + j_0 - k_0 \\
B(j_0, i_0, n_0) &= \cdots \\
A(i_0, n_0) &= \cdots
\end{align*}
\]

\[
\begin{align*}
A(i, n) &= 0 \quad \{i \geq n\} \\
A(i, n) &= B(0, i, n) + A(i', n) \quad \{i + 1 \leq n, i + 2 \leq i' \leq i + 4\}
\end{align*}
\]

\[
\begin{align*}
B(j, i, n) &= 0 \quad \{j \geq i\} \\
B(j, i, n) &= C(0, j, n) + B(j', i, n) \quad \{j + 1 \leq i, j + 1 \leq j' \leq j + 3\}
\end{align*}
\]

\[
\begin{align*}
C(k, j, n) &= 0 \quad \{k \geq n + j\} \\
C(k, j, n) &= 1 + C(k', j, n) \quad \{k' = k + 1, k + 1 \leq n + j\}
\end{align*}
\]
Program

Static Analysis

Cost Relations

CRs Solver

Best/Worst Case

Worst-Case (UB)

\[ C(k_0, j_0, n_0) = n_0 + j_0 - k_0 \]
\[ B(j_0, i_0, n_0) = \ldots \]
\[ A(i_0, n_0) = \ldots \]

Best-Case (LB)

\[ C(k_0, j_0, n_0) = n_0 + j_0 - k_0 \]
\[ B(j_0, i_0, n_0) = \ldots \]
\[ A(i_0, n_0) = \ldots \]

\[ A(i, n) = \begin{cases} 0 & \text{if } i = n \\ \infty & \text{otherwise} \end{cases} \]
\[ A(i, n) = \infty \}

\[ B(j, i, n) = \begin{cases} 0 & \text{if } j = i \\ \infty & \text{otherwise} \end{cases} \}
\[ B(j, i, n) = \infty \}

\[ C(k, j, n) = 1 + C(k', j, n) \quad \text{if } k \geq n+j \}
\[ \{ k' = k+1, k+1 \leq n+j \} \]
Solving CRs - Computer Algebra Systems

\[ A(i, n) = 0 \quad \{i \geq n\} \]
\[ A(i, n) = B(0, i, n) + A(i', n) \quad \{i+1 \leq n, i+2 \leq i' \leq i+4\} \]
\[ B(j, i, n) = 0 \quad \{j \geq i\} \]
\[ B(j, i, n) = C(0, j, n) + B(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\} \]
\[ C(k, j, n) = 0 \quad \{k \geq n+j\} \]
\[ C(k, j, n) = 1 + C(k', j, n) \quad \{k' = k+1, k+1 \leq n+j\} \]
Solving CRs - Computer Algebra Systems

\[ A(i, n) = 0 \quad \{i \geq n\} \]
\[ A(i, n) = B(0, i, n) + A(i', n) \quad \{i+1 \leq n, i+2 \leq i' \leq i+4\} \]
\[ B(j, i, n) = 0 \quad \{j \geq i\} \]
\[ B(j, i, n) = C(0, j, n) + B(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\} \]
\[ C(k, j, n) = 0 \quad \{k \geq n+j\} \]
\[ C(k, j, n) = 1 + C(k', j, n) \quad \{k' = k+1, k+1 \leq n+j\} \]

- Why not using directly Computer Algebra Systems?
Solving CRs - Computer Algebra Systems

\[
A(i, n) = 0 \quad \{i \geq n\}
\]
\[
A(i, n) = B(0, i, n) + A(i', n) \quad \{i+1 \leq n, i+2 \leq i' \leq i+4\}
\]
\[
B(j, i, n) = 0 \quad \{j \geq i\}
\]
\[
B(j, i, n) = C(0, j, n) + B(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\}
\]
\[
C(k, j, n) = 0 \quad \{k \geq n+j\}
\]
\[
C(k, j, n) = 1 + C(k', j, n) \quad \{k' = k+1, k+1 \leq n+j\}
\]

CAS can obtain an **exact closed-form solution** for:

\[
P(0) = 0
\]
\[
P(x) = E + P(x - 1) + \cdots + P(x - 1)
\]

*Deterministic, 1 base-case, 1 recursive case, 1 argument*
Solving CRs - Computer Algebra Systems

\[ A(i, n) = 0 \quad \{i \geq n\} \]
\[ A(i, n) = B(0, i, n) + A(i', n) \quad \{i+1 \leq n, i+2 \leq i' \leq i+4\} \]
\[ B(j, i, n) = 0 \quad \{j \geq i\} \]
\[ B(j, i, n) = C(0, j, n) + B(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\} \]
\[ C(k, j, n) = 0 \quad \{k \geq n+j\} \]
\[ C(k, j, n) = 1 + C(k', j, n) \quad \{k' = k+1, k+1 \leq n+j\} \]

- Why not using directly Computer Algebra Systems?
- CRs are not deterministic
Solving CRs - Computer Algebra Systems

\[
\begin{align*}
A(i, n) & = 0 & \{i \geq n\} \\
A(i, n) & = B(0, i, n) + A(i', n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\} \\
B(j, i, n) & = 0 & \{j \geq i\} \\
B(j, i, n) & = C(0, j, n) + B(j', i, n) & \{j+1 \leq i, j+1 \leq j' \leq j+3\} \\
C(k, j, n) & = 0 & \{k \geq n+j\} \\
C(k, j, n) & = 1 + C(k', j, n) & \{k' = k+1, k+1 \leq n+j\}
\end{align*}
\]

Why not using directly Computer Algebra Systems?

CRs are not deterministic
CRs have multiple arguments
CRs have multiple (not mutually exclusive) equations
Thus, CRs often do not have an exact solution

Two possible runs for \(B(1, 5, 3)\)

1: \(B(1, 5, 3) \rightarrow B(2, 5, 3) \rightarrow B(5, 5, 3)\)
2: \(B(1, 5, 3) \rightarrow B(2, 5, 3) \rightarrow B(4, 5, 3) \rightarrow B(5, 5, 3)\)
Solving CRs - Computer Algebra Systems

\[ A(i, n) = 0 \quad \{i \geq n\} \]
\[ A(i, n) = B(0, i, n) + A(i', n) \quad \{i+1 \leq n, i+2 \leq i' \leq i+4\} \]
\[ B(j, i, n) = 0 \quad \{j \geq i\} \]
\[ B(j, i, n) = C(0, j, n) + B(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\} \]
\[ C(k, j, n) = 0 \quad \{k \geq n+j\} \]
\[ C(k, j, n) = 1 + C(k', j, n) \quad \{k' = k+1, k+1 \leq n+j\} \]

- Why not using directly Computer Algebra Systems?
- CRs are not deterministic
- CRs have multiple arguments


\begin{align*}
A(i, n) & = 0 & \{i \geq n\} \\
A(i, n) & = B(0, i, n) + A(i', n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\} \\
\begin{cases}
B(j, i, n) = 0 & \{j \geq i\} \\
B(j, i, n) = C(0, j, n) + B(j', i, n) & \{j+1 \leq i, j+1 \leq j' \leq j+3\}
\end{cases} \\
C(k, j, n) & = 0 & \{k \geq n+j\} \\
C(k, j, n) & = 1 + C(k', j, n) & \{k' = k+1, k+1 \leq n+j\}
\end{align*}

\begin{itemize}
  \item Why not using directly Computer Algebra Systems?
  \item CRs are not deterministic
  \item CRs have multiple arguments
  \item CRs have multiple (not mutually exclusive) equations
\end{itemize}
Solving CRs - Computer Algebra Systems

\[ A(i, n) = 0 \quad \{ i \geq n \} \]
\[ A(i, n) = B(0, i, n) + A(i', n) \quad \{ i + 1 \leq n, i + 2 \leq i' \leq i + 4 \} \]
\[
\begin{cases}
B(j, i, n) = 0 & \{ j \geq i \} \\
B(j, i, n) = C(0, j, n) + B(j', i, n) & \{ j + 1 \leq i, j + 1 \leq j' \leq j + 2 \} \\
B(j, i, n) = C(0, j, n) + B(j', i, n) & \{ j + 1 \leq i, j' = j + 3 \}
\end{cases}
\]
\[ C(k, j, n) = 0 \quad \{ k \geq n + j \} \]
\[ C(k, j, n) = 1 + C(k', j, n) \quad \{ k' = k + 1, k + 1 \leq n + j \} \]

- Why not using directly Computer Algebra Systems?
- CRs are not deterministic
- CRs have multiple arguments
- CRs have multiple (not mutually exclusive) equations
**Solving CRs - Computer Algebra Systems**

\[
A(i, n) = 0 \quad \{i \geq n\}
\]
\[
A(i, n) = B(0, i, n) + A(i', n) \quad \{i+1 \leq n, i+2 \leq i' \leq i+4\}
\]
\[
B(j, i, n) = 0 \quad \{j \geq i\}
\]
\[
B(j, i, n) = C(0, j, n) + B(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+2\}
\]
\[
B(j, i, n) = C(0, j, n) + B(j', i, n) \quad \{j+1 \leq i, j' = j+3\}
\]
\[
C(k, j, n) = 0 \quad \{k \geq n+j\}
\]
\[
C(k, j, n) = 1 + C(k', j, n) \quad \{k' = k+1, k+1 \leq n+j\}
\]

- Why not using directly Computer Algebra Systems?
- CRs are not deterministic
- CRs have multiple arguments
- CRs have multiple (not mutually exclusive) equations
- Thus, CRs often do not have an exact solution
Solving CRs - Computer Algebra Systems

\[
A(i, n) = 0 \quad \{i \geq n\}
\]
\[
A(i, n) = B(0, i, n) + A(i', n) \quad \{i + 1 \leq n, i + 2 \leq i' \leq i + 4\}
\]
\[
\begin{align*}
B(j, i, n) &= 0 \quad \{j \geq i\} \\
B(j, i, n) &= C(0, j, n) + B(j', i, n) \quad \{j + 1 \leq i, j + 1 \leq j' \leq j + 2\} \\
B(j, i, n) &= C(0, j, n) + B(j', i, n) \quad \{j + 1 \leq i, j' = j + 3\}
\end{align*}
\]
\[
C(k, j, n) = 0 \quad \{k \geq n + j\}
\]
\[
C(k, j, n) = 1 + C(k', j, n) \quad \{k' = k + 1, k + 1 \leq n + j\}
\]

- Why not using directly Computer Algebra Systems?
- CRs are not deterministic
- CRs have multiple arguments
- CRs have multiple (not mutually exclusive) equations
- Thus, CRs often do not have an exact solution
Solving CRs - Computer Algebra Systems

\[ A(i, n) = 0 \quad \{i \geq n\} \]
\[ A(i, n) = B(0, i, n) + A(i', n) \quad \{i+1 \leq n, i+2 \leq i' \leq i+4\} \]

\[ \begin{align*}
B(j, i, n) &= 0 \quad \{j \geq i\} \\
B(j, i, n) &= C(0, j, n) + B(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+2\} \\
B(j, i, n) &= C(0, j, n) + B(j', i, n) \quad \{j+1 \leq i, j' = j+3\} \\
C(k, j, n) &= 0 \quad \{k \geq n+j\} \\
C(k, j, n) &= 1 + C(k', j, n) \quad \{k' = k+1, k+1 \leq n+j\}
\end{align*} \]

- Why not using directly Computer Algebra Systems?
- CRs are not deterministic
- CRs have multiple arguments
- CRs have multiple (not mutually exclusive) equations
- Thus, CRs often do not have an exact solution

Abu Naser Masud, UPM  
Termination and Cost Analysis: Complexity and Precision Issues  
11/34
Inferring Upper Bounds (CRs) - PUBS

\[
\begin{align*}
A(i, n) &= 0 & \{i \geq n\} \\
A(i, n) &= B(0, i, n) + A(i', n) & \{i + 1 \leq n, i + 2 \leq i' \leq i + 4\} \\
B(j, i, n) &= 0 & \{j \geq i\} \\
B(j, i, n) &= C(0, j, n) + B(j', i, n) & \{j + 1 \leq i, j + 1 \leq j' \leq j + 3\} \\
C(k, j, n) &= 0 & \{k \geq n + j\} \\
C(k, j, n) &= 1 + C(k', j, n) & \{k' = k + 1, k + 1 \leq n + j\}
\end{align*}
\]
Inferring Upper Bounds (CRs) - PUBS

\[
\begin{align*}
A(i, n) &= 0 \quad \{i \geq n\} \\
A(i, n) &= B(0, i, n) + A(i', n) \quad \{i + 1 \leq n, i + 2 \leq i' \leq i + 4\} \\
B(j, i, n) &= 0 \quad \{j \geq i\} \\
B(j, i, n) &= C(0, j, n) + B(j', i, n) \quad \{j + 1 \leq i, j + 1 \leq j' \leq j + 3\} \\
C(k, j, n) &= 0 \quad \{k \geq n + j\} \\
C(k, j, n) &= 1 + C(k', j, n) \quad \{k' = k + 1, k + 1 \leq n + j\}
\end{align*}
\]
Inferring Upper Bounds (CRs) - PUBS

\[
A(i, n) = 0 \quad \{i \geq n\}
\]

\[
A(i, n) = B(0, i, n) + A(i', n) \quad \{i+1 \leq n, \ i+2 \leq i' \leq i+4\}
\]

\[
B(j, i, n) = 0 \quad \{j \geq i\}
\]

\[
B(j, i, n) = C(0, j, n) + B(j', i, n) \quad \{j+1 \leq i, \ j+1 \leq j' \leq j+3\}
\]

\[
C(k, j, n) = 0 \quad \{k \geq n+j\}
\]

\[
C(k, j, n) = 1 + C(k', j, n) \quad \{k' = k+1, \ k+1 \leq n+j\}
\]

• An evaluation for \(C(k_0, j_0, n_0)\) looks like:
Inferring Upper Bounds (CRs) - PUBS

\[ A(i, n) = 0 \quad \{i \geq n\} \]
\[ A(i, n) = B(0, i, n) + A(i', n) \quad \{i+1 \leq n, i+2 \leq i' \leq i+4\} \]
\[ B(j, i, n) = 0 \quad \{j \geq i\} \]
\[ B(j, i, n) = C(0, j, n) + B(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\} \]
\[ C(k, j, n) = 0 \quad \{k \geq n+j\} \]
\[ C(k, j, n) = 1 + C(k', j, n) \quad \{k' = k+1, k+1 \leq n+j\} \]

- An evaluation for \( C(k_0, j_0, n_0) \) looks like:

1 → 1 → 1 → 1

- What is the maximum length of this chain of 1?
Inferring Upper Bounds (CRs) - PUBS

\[ A(i, n) = \begin{cases} 0 & \{i \geq n\} \\ B(0, i, n) + A(i', n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\} \end{cases} \]

\[ B(j, i, n) = \begin{cases} 0 & \{j \geq i\} \\ C(0, j, n) + B(j', i, n) & \{j+1 \leq i, j+1 \leq j' \leq j+3\} \end{cases} \]

\[ C(k, j, n) = \begin{cases} 0 & \{k \geq n+j\} \\ 1 + C(k', j, n) & \{k' = k+1, k+1 \leq n+j\} \end{cases} \]

- An evaluation for \( C(k_0, j_0, n_0) \) looks like:

  \[ \begin{array}{ccccccc} 1 & \to & 1 & \to & 1 & \to & 1 \end{array} \]

- What is the maximum length of this chain of \( 1 \)?
\[ A(i, n) = 0 \quad \{ i \geq n \} \]

\[ A(i, n) = B(0, i, n) + A(i', n) \quad \{ i + 1 \leq n, i + 2 \leq i' \leq i + 4 \} \]

\[ B(j, i, n) = 0 \quad \{ j \geq i \} \]

\[ B(j, i, n) = C(0, j, n) + B(j', i, n) \quad \{ j + 1 \leq i, j + 1 \leq j' \leq j + 3 \} \]

\[ C(k, j, n) = 0 \quad \{ k \geq n + j \} \]

\[ C(k, j, n) = 1 + C(k', j, n) \quad \{ k' = k + 1, k + 1 \leq n + j \} \]

- An evaluation for \( C(k_0, j_0, n_0) \) looks like:

\[ \begin{array}{cccccc}
1 & \rightarrow & 1 & \rightarrow & 1 & \rightarrow & 1 \\
\end{array} \]

- What is the maximum length of this chain of \( 1 \)?

\[ \exists \hat{f}. \; \varphi \models \hat{f}(k, j, n) \geq 0 \land \hat{f}(k, j, n) - \hat{f}(k', j', n') \geq 1 \]
Inferring Upper Bounds (CRs) - PUBS

\[ A(i, n) = 0 \quad \{i \geq n\} \]
\[ A(i, n) = B(0, i, n) + A(i', n) \quad \{i + 1 \leq n, i + 2 \leq i' \leq i + 4\} \]
\[ B(j, i, n) = 0 \quad \{j \geq i\} \]
\[ B(j, i, n) = C(0, j, n) + B(j', i, n) \quad \{j + 1 \leq i, j + 1 \leq j' \leq j + 3\} \]

\[ C(k, j, n) = 0 \quad \{k \geq n + j\} \]
\[ C(k, j, n) = 1 + C(k', j, n) \quad \{k' = k + 1, k + 1 \leq n + j\} \]

- An evaluation for \( C(k_0, j_0, n_0) \) looks like:

\[ \begin{array}{cccccc}
1 & \rightarrow & 1 & \rightarrow & 1 & \rightarrow & 1
\end{array} \]

- What is the maximum length of this chain of \( 1 \) ?

\[ \hat{f}(k_0, j_0, n_0) = \|n_0 + j_0 - k_0\| \]
Inferring Upper Bounds (CRs) - PUBS

\[ A(i, n) = \begin{cases} 0 & \{i \geq n\} \\ A(i, n) + B(0, i, n) + A(i', n) & \{i + 1 \leq n, i + 2 \leq i' \leq i + 4\} \end{cases} \]

\[ B(j, i, n) = \begin{cases} 0 & \{j \geq i\} \\ C(0, j, n) + B(j', i, n) & \{j + 1 \leq i, j + 1 \leq j' \leq j + 3\} \end{cases} \]

\[ C(k, j, n) = \begin{cases} 0 & \{k \geq n + j\} \\ 1 + C(k', j, n) & \{k' = k + 1, k + 1 \leq n + j\} \end{cases} \]

- An evaluation for \( C(k_0, j_0, n_0) \) looks like:

\[ \hat{f}(k_0, j_0, n_0) = \|n_0 + j_0 - k_0\| \]
Inferring Upper Bounds (CRs) - PUBS

\[ A(i, n) = \begin{cases} 0 & \text{if } i > n \\ B(0, i, n) + A(i', n) & \text{if } i+1 \leq i' \leq i+4 \\
\end{cases} \]

\[ B(j, i, n) = \begin{cases} 0 & \text{if } j > i \\ C(0, j, n) + B(j', i, n) & \text{if } j+1 \leq j' \leq j+3 \\
\end{cases} \]

\[ C(k, j, n) = \begin{cases} 0 & \text{if } k > n+j \\ 1 + C(k', j, n) & \text{if } k' = k+1, k+1 \leq n+j \\
\end{cases} \]

- An evaluation for \( C(k_0, j_0, n_0) \) looks like:

\[ C^{ub}(k_0, j_0, n_0) = 1 \cdot \|n_0 + j_0 - k_0\| \]

- What is the maximum length of this chain of \( 1 \)?

\[ \hat{f}(k_0, j_0, n_0) = \|n_0 + j_0 - k_0\| \]
Inferring Upper Bounds (CRs) - PUBS

\[ A(i, n) = \begin{cases} 0 & \text{if } i \geq n \\ \leq i+4 \end{cases} \]

\[ B(j, i, n) = \begin{cases} 0 & \text{if } j \geq i \\ \leq j+3 \end{cases} \]

\[ C^{ub}(k_0, j_0, n_0) = 1 \times 2^{||n_0+j_0-k_0||} \]

\[ C(k, j, n) = \begin{cases} 0 & \text{if } k \geq n+j \\ 1 + C(k', j, n) + C(k', j, n) & \text{if } k' = k+1, k+1 \leq n+j \end{cases} \]

- An evaluation for \( C(k_0, j_0, n_0) \) looks like:

\[ \hat{f}(k_0, j_0, n_0) = ||n_0+j_0-k_0|| \]

- How many \( 1 \) has from root to leaf?

Abu Naser Masud, UPM

Termination and Cost Analysis: Complexity and Precision Issues
$$A(i, n) = 0 \quad \{ i \geq n \}$$
$$A(i, n) = B(0, i, n) + A(i', n) \quad \{ i + 1 \leq n, i + 2 \leq i' \leq i + 4 \}$$

$$B(j, i, n) = 0 \quad \{ j \geq i \}$$
$$B(j, i, n) = C(0, j, n) + B(j', i, n) \quad \{ j + 1 \leq i, j + 1 \leq j' \leq j + 3 \}$$
Inferring Upper Bounds (CRs) - PUBS

\[ A(i, n) = 0 \quad \{i \geq n\} \]
\[ A(i, n) = B(0, i, n) + A(i', n) \quad \{i+1 \leq n, i+2 \leq i' \leq i+4\} \]

\[ B(j, i, n) = 0 \quad \{j \geq i\} \]
\[ B(j, i, n) = \|n+j\| + B(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\} \]
Inferring Upper Bounds (CRs) - PUBS

\[
A(i, n) = 0 \quad \{i \geq n\} \\
A(i, n) = B(0, i, n) + A(i', n) \quad \{i + 1 \leq n, i + 2 \leq i' \leq i + 4\}
\]

\[
B(j, i, n) = 0 \quad \{j \geq i\} \\
B(j, i, n) = \|n+j\| + B(j', i, n) \quad \{j + 1 \leq i, j + 1 \leq j' \leq j + 3\}
\]

- An evaluation for \(B(j_0, i_0, n_0)\) looks like:
Inferring Upper Bounds (CRs) - PUBS

\[ A(i, n) = 0 \quad \{ i \geq n \} \]
\[ A(i, n) = B(0, i, n) + A(i', n) \quad \{ i + 1 \leq n, i + 2 \leq i' \leq i + 4 \} \]

\[ B(j, i, n) = 0 \quad \{ j \geq i \} \]
\[ B(j, i, n) = \| n + j \| + B(j', i, n) \quad \{ j + 1 \leq i, j + 1 \leq j' \leq j + 3 \} \]

- An evaluation for \( B(j_0, i_0, n_0) \) looks like:

- There are at most \( \| i_0 - j_0 \| \) circles (ranking function)
Inferring Upper Bounds (CRs) - PUBS

\[
A(i, n) = 0 \quad \{i \geq n\} \\
A(i, n) = B(0, i, n) + A(i', n) \quad \{i+1 \leq n, i+2 \leq i' \leq i+4\}
\]

\[
B(j, i, n) = 0 \quad \{j \geq i\} \\
B(j, i, n) = \|n+j\| + B(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\}
\]

- An evaluation for \(B(j_0, i_0, n_0)\) looks like:

- There are at most \(\|i_0 - j_0\|\) circles (ranking function)
Inferring Upper Bounds (CRs) - PUBS

\[ A(i, n) = 0 \quad \{i \geq n\} \]
\[ A(i, n) = B(0, i, n) + A(i', n) \quad \{i+1 \leq n, i+2 \leq i' \leq i+4\} \]

\[ B(j, i, n) = 0 \quad \{j \geq i\} \]
\[ B(j, i, n) = ||n+j|| + B(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\} \]

- An evaluation for \( B(j_0, i_0, n_0) \) looks like:

- There are at most \( ||i_0 - j_0|| \) circles (ranking function)

- Worst-case is \(*||i_0 - j_0||*\)
\[
A(i, n) = 0 \quad \{i \geq n\}
\]
\[
A(i, n) = B(0, i, n) + A(i', n) \quad \{i+1 \leq n, i+2 \leq i' \leq i+4\}
\]
\[
B(j, i, n) = 0 \quad \{j \geq i\}
\]
\[
B(j, i, n) = \|n+j\| + B(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\}
\]

• An evaluation for \(B(j_0, i_0, n_0)\) looks like:

• What is the maximum value that \(n+j\) can take in terms of \(\langle j_0, i_0, n_0 \rangle\)? It is \(n_0 + i_0 - 1\).
Inferring Upper Bounds (CRs) - PUBS

\[
A(i, n) = \begin{cases} 
0 & \{i \geq n\} \\
B(0, i, n) + A(i', n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\}
\end{cases}
\]

\[
B(j, i, n) = \begin{cases} 
0 & \{j \geq i\} \\
\|n+j\| + B(j', i, n) & \{j+1 \leq i, j+1 \leq j' \leq j+3\}
\end{cases}
\]

- What is the maximum value that \(n+j\) can take in terms of \(\langle j_0, i_0, n_0 \rangle\)? It is \(n_0 + i_0 - 1\).
- Infer an invariant \(\langle B(j_0, i_0, n_0) \leadsto B(j, i, n), \Psi \rangle\)
Inferring Upper Bounds (CRs) - PUBS

\[ A(i, n) = 0 \quad \{i \geq n\} \]
\[ A(i, n) = B(0, i, n) + A(i', n) \quad \{i+1 \leq n, i+2 \leq i' \leq i+4\} \]

\[ B(j, i, n) = 0 \quad \{j \geq i\} \]
\[ B(j, i, n) = \|n+j\| + B(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\} \]

- An evaluation for \( B(j_0, i_0, n_0) \) looks like:
  - There are at most \( \|i_0 - j_0\| \) circles (ranking function)
  - Worst-case is \( \ast \|i_0 - j_0\| \)

What is the maximum value that \( n+j \) can take in terms of \( \langle j_0, i_0, n_0 \rangle \)? It is \( n_0 + i_0 - 1 \).

- Infer an invariant \( \langle B(j_0, i_0, n_0) \leadsto B(j, i, n), \Psi \rangle \)
- Use (parametric) integer programming to maximize \( n+j \) w.r.t \( \Psi \land \varphi \) and the parameters \( \langle j_0, i_0, n_0 \rangle \).
Inferring Upper Bounds (CRs) - PUBS

\[
A(i, n) = \begin{cases} 
0 & \{i \geq n\} \\
B(0, i, n) + A(i', n) & \{i + 1 \leq n, i + 2 \leq i' \leq i + 4\}
\end{cases}
\]

\[
B(j, i, n) = \begin{cases} 
0 & \{j \geq i\} \\
\|n + j\| + B(j', i, n) & \{j + 1 \leq i, j + 1 \leq j' \leq j + 3\}
\end{cases}
\]

- An evaluation for \(B(j_0, i_0, n_0)\) looks like:

- There are at most \(\|i_0 - j_0\|\) circles (\textit{ranking function})

\[
B^{ub}(j_0, i_0, n_0) = \|n_0 + i_0 - 1\| * \|i_0 - j_0\|
\]
Inferring Upper Bounds (CRs) - PUBS

\[ A(i, n) = 0 \quad \{i \geq n\} \]
\[ A(i, n) = B(0, i, n) + A(i', n) \quad \{i + 1 \leq n, i + 2 \leq i' \leq i + 4\} \]

\[ B(j, i, n) = 0 \quad \{j \geq i\} \]
\[ B(j, i, n) = \|n + j\| + B(j', i, n) \quad \{j + 1 \leq i, j + 1 \leq j' \leq j + 3\} \]

- An evaluation for \( B(j_0, i_0, n_0) \) looks like:

- There are at most \( \|i_0 - j_0\| \) circles (ranking function)
1. Introduction

2. Background on Cost Analysis

3. Precise Cost Analysis Techniques

4. Theoretical Complexity of Deciding Termination

5. Conclusions
Inferring Upper Bounds - Static analysis + CAS

\[ B(j, i, n) = 0 \quad \{ j \geq i \} \]
\[ B(j, i, n) = \| n + j \| + B(j', i, n) \quad \{ j + 1 \leq i, j + 1 \leq j' \leq j + 3 \} \]

- An evaluation for \( B(j_0, i_0, n_0) \) looks like:

![Diagram of evaluation process]
\[
B(j, i, n) = 0 \quad \{j \geq i\}
\]
\[
B(j, i, n) = \|n + j\| + B(j', i, n) \quad \{j + 1 \leq i, j + 1 \leq j' \leq j + 3\}
\]

- An evaluation for \(B(j_0, i_0, n_0)\) looks like:

- How these circles progress?
Inferring Upper Bounds - Static analysis + CAS

\[
B(j, i, n) = 0 \quad \{j \geq i\}
\]

\[
B(j, i, n) = \|n + j\| + B(j', i, n) \quad \{j + 1 \leq i, j + 1 \leq j' \leq j + 3\}
\]

- An evaluation for \(B(j_0, i_0, n_0)\) looks like:

  ![Diagram](image)

- How these circles progress? \(\ddot{d} = 1\) and \(\hat{d} = 3\)
Inferring Upper Bounds - Static analysis + CAS

\[ B(j, i, n) = 0 \]
\[ B(j, i, n) = \| n + j \| + B(j', i, n) \quad \{ j \geq i \} \]
\[ \{ j + 1 \leq i, j + 1 \leq j' \leq j + 3 \} \]

- An evaluation for \( B(j_0, i_0, n_0) \) looks like:

- How these circles progress? \( \hat{d} = 1 \) and \( \hat{d} = 3 \)
Inferring Upper Bounds - Static analysis + CAS

\[ B(j, i, n) = 0 \]
\[ B(j, i, n) = \| n + j \| + B(j', i, n) \]

\{ j \geq i \}
\{ j + 1 \leq i, j + 1 \leq j' \leq j + 3 \}
Inferring Upper Bounds - Static analysis + CAS

\[
B(j, i, n) = 0 \\
B(j, i, n) = \|n + j\| + B(j', i, n)
\]

\[
\begin{align*}
\tilde{d} & \quad \tilde{d} & \quad \tilde{d} \\
\end{align*}
\]

\[
P(0) = 0 \\
P(x) = + (\|i_0 - j_0\| - x) \ast \tilde{d} + P(x - 1)
\]

\[
= \|l\| \\
= \|l - (i_0 - j_0 - 1) \ast \tilde{d}\|
\]
Inferring Upper Bounds - Static analysis + CAS

\[ B(j, i, n) = 0 \]
\[ B(j, i, n) = \| n + j \| + B(j', i, n) \quad \{ j \geq i \} \]
\[ \{ j + 1 \leq i, j + 1 \leq j' \leq j + 3 \} \]

\[ P(0) = 0 \]
\[ P(x) = \sum (\| i_0 - j_0 \| - x) \cdot \dd + P(x - 1) \]

- \( P(\| i_0 - j_0 \|) \) is the sum of all boxes
Inferring Upper Bounds - Static analysis + CAS

\[ B(j, i, n) = 0 \]
\[ B(j, i, n) = \| n + j \| + B(j', i, n) \]

\{ j \geq i \}
\{ j + 1 \leq i, j + 1 \leq j' \leq j + 3 \}

\[ \tilde{d} \]

\[ \tilde{d} \]

\[ \tilde{d} \]

\[ \tilde{d} \]

\[ P(0) = 0 \]
\[ P(x) = (\| i_0 - j_0 \| - x) \cdot \tilde{d} + P(x - 1) \]

• \( P(\| i_0 - j_0 \|) \) is the sum of all boxes

• If \( E \) is a closed-form solution for \( P(x) \) obtained by CAS, then \( E[\tfrac{x}{\| i_0 - j_0 \|}] \) is an UB on the worst-case of \( B \)
\[ P(x) = \|n_0 + j_0 - 1\| \times x + \|i_0 - j_0\| \times x + \frac{x^2}{2} \]

\[ P(\|i_0 - j_0\|) = \|n_0 + j_0 - 1\| \times \|i_0 - j_0\| + \|i_0 - j_0\|^2 + \frac{\|i_0 - j_0\|}{2} - \frac{\|i_0 - j_0\|^2}{2} \]

\[ P(0) = 0 \]

\[ P(x) = P(x - 1) + (\|i_0 - j_0\| - x) \times \bar{d} \]

- \( P(\|i_0 - j_0\|) \) is the sum of all boxes

- If \( E \) is a closed-form solution for \( P(x) \) obtained by CAS, then \( E[x/\|i_0 - j_0\|] \) is an UB on the worst-case of \( B \)
CRs with Complex Nat Expression

\[
\begin{align*}
A(i, n) &= 0 & \varphi_0 \\
A(i, n) &= B(0, i, n) + A(i', n) & \varphi_1
\end{align*}
\]
CRs with Complex Nat Expression

\[ A(i, n) = 0 \quad \varphi_0 \]
\[ A(i, n) = \| n_0 - 1 \| \cdot \| i_0 \| + \frac{\| i_0 \|}{2} \cdot (\| i_0 \| + 1) + A(i', n) \quad \varphi_1 \]
Inferring Upper Bounds - Static analysis + CAS

CRs with Complex Nat Expression

\[ A(i, n) = \begin{cases} 0 & \varphi_0 \\ \|n_0 - 1\| * \|i_0\| + \frac{\|i_0\|}{2} * (\|i_0\| + 1) + A(i', n) & \varphi_1 \end{cases} \]
CRs with Complex Nat Expression

\[ A(i, n) = 0 \quad \varphi_0 \]

\[ A(i, n) = \| n_0 - 1 \| \| i_0 \| + \frac{\| i_0 \|}{2} \cdot (\| i_0 \| + 1) + A(i', n) \quad \varphi_1 \]

\[ P(0) = 0 \]

\[ P(x) = E_1 \cdot E_2 + \frac{E_3}{2} \cdot (E_4 + 1) + P(x - 1) \]

\[ E_1 = + (\| n_0 - i_0 \| - x) \cdot \tilde{d}_1 \]
\[ E_2 = + (\| n_0 - i_0 \| - x) \cdot \tilde{d}_2 \]
\[ E_3 = + (\| n_0 - i_0 \| - x) \cdot \tilde{d}_3 \]
\[ E_4 = + (\| n_0 - i_0 \| - x) \cdot \tilde{d}_4 \]
Geometrically Progressive nat

\[ \begin{align*}
    Ms(l, h) &= 0, & \{ h \leq l, h \geq 0, l \geq 0 \} \\
    Ms(l, h) &= \| h - l + 1 \| + Ms(l, m) + Ms(m+1, h), & \{ h \geq l+1, l+h-1 \leq 2\times m \leq l+h \}
\end{align*} \]

CRs for Mergesort
Inferring Upper Bounds - Static analysis + CAS

Geometrically Progressive nat

\[ M_s(l, h) = 0, \quad \{ h \leq l, h \geq 0, l \geq 0 \} \]

\[ M_s(l, h) = \|h-l+1\| + M_s(l, m) + M_s(m+1, h), \quad \{ h \geq l+1, l+h-1 \leq 2m \leq l+h \} \]

CRs for Mergesort
Inferring Upper Bounds - Static analysis + CAS

Geometrically Progressive nat

\[ Ms(l, h) = 0, \quad \{ h \leq l, h \geq 0, l \geq 0 \} \]

\[ Ms(l, h) = \|h-l+1\| + Ms(l, m) + Ms(m+1, h), \quad \{ h \geq l+1, l+h-1 \leq 2m \leq l+h \} \]

CRs for Mergesort
CRs with Multiple Recursive Equations

\[ B(j, i, n) = 0 \quad \{j \geq i\} \]
\[ B(j, i, n) = \|n + j\| + B(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\} \]
\[ B(j, i, n) = \|j\| \ast \|j\| + B(j', i, n) \quad \{j+1 \leq i, j' = j+1\} \]
CRs with Multiple Recursive Equations - First Alternative

\[
B(j, i, n) = 0 \quad \{j \geq i\}
\]

\[
B(j, i, n) = \| n + j \| + B(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\}
\]

\[
B(j, i, n) = \| j \| \times \| j \| + B(j', i, n) \quad \{j+1 \leq i, j' = j+1\}
\]

\[\Downarrow\] Generate a single recursive CRs \( B_T(j, i, n) \)

\[
B_T(j, i, n) = \| n + j \| \times \| j + 1 \| + B_T(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\}
\]
CRs with Multiple Recursive Equations - First Alternative

\[ B(j, i, n) = 0 \quad \{ j \geq i \} \]

\[ B(j, i, n) = \|n + j\| + B(j', i, n) \quad \{ j + 1 \leq i, j + 1 \leq j' \leq j + 3 \} \]

\[ B(j, i, n) = \|j\| \cdot \|j\| + B(j', i, n) \quad \{ j + 1 \leq i, j' = j + 1 \} \]

↓ Generate a single recursive CRs \( B_T(j, i, n) \)

\[ B_T(j, i, n) = \|n + j\| \cdot \|j + 1\| + B_T(j', i, n) \quad \{ j + 1 \leq i, j + 1 \leq j' \leq j + 3 \} \]
CRs with Multiple Recursive Equations - First Alternative

\[
B(j, i, n) = 0 \quad \{j \geq i\}
\]

\[
B(j, i, n) = \|n + j\| + B(j', i, n) \quad \{j + 1 \leq i, j + 1 < j' < j + 3\}
\]

\[
B(j, i, n) = j \|j\| + B(j', i, n) \quad \{j + 1 \leq i, j' = j + 1\}
\]

\[
\downarrow \text{Generate a single recursive CRs } B_T(j, i, n)
\]

\[
B_T(j, i, n) = \|n + j\| \|j + 1\| + B_T(j', i, n) \quad \{j + 1 \leq i, j + 1 < j' < j + 3\}
\]

\[
\text{conv.hull } \{\varphi_1, \varphi_2\}
\]
CRs with Multiple Recursive Equations - First Alternative

\[
B(j, i, n) = 0 \quad \{j \geq i\}
\]

\[
B(j, i, n) = \| n + j \| + B(j', i, n) \quad \{j + 1 \leq i, j + 1 \leq j' \leq j + 3\}
\]

\[
B(j, i, n) = \| j \| \ast \| j \| + B(j', i, n) \quad \{j + 1 \leq i, j' = j + 1\}
\]

\[\downarrow \text{Generate a single recursive CRs } B_T(j, i, n)\]

\[
B_T(j, i, n) = \| n + j \| \ast \| j + 1 \| + B_T(j', i, n) \quad \{j + 1 \leq i, j + 1 \leq j' \leq j + 3\}
\]

\[
P(x) = 0
\]

\[
P(x) = E_1 \ast E_2 + P(x - 1)
\]

\[
E_1 = \| i_0 - j_0 \| - x \ast \tilde{d}_1
\]

\[
E_2 = \| i_0 - j_0 \| - x \ast \tilde{d}_2
\]
**Inferring Upper Bounds - Static analysis + CAS**

**CRs with Multiple Recursive Equations - Second Alternative**

\[
B(j, i, n) = 0 \quad \{j \geq i\}
\]

\[
B(j, i, n) = \|n + j\| + B(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\}
\]

\[
B(j, i, n) = \|j\| \times \|j\| + B(j', i, n) \quad \{j+1 \leq i, j' = j+1\}
\]

\[
B_1(j, i, n) = 0 \quad \{j \geq i\}
\]

\[
B_1(j, i, n) = \|n + j\| + B_1(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\}
\]

\[
B_1(j, i, n) = 0 + B_1(j', i, n) \quad \{j+1 \leq i, j' = j+1\}
\]

\[
B_2(j, i, n) = 0 \quad \{j \geq i\}
\]

\[
B_2(j, i, n) = 0 + B_2(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\}
\]

\[
B_2(j, i, n) = \|j\| \times \|j\| + B_2(j', i, n) \quad \{j+1 \leq i, j' = j+1\}
\]

\[
B_{ub}(j_0, i_0, n_0) = B_{1ub}(j_0, i_0, n_0) + B_{2ub}(j_0, i_0, n_0)
\]
Inferring Lower Bounds

\[ B(j, i, n) = 0 \quad \{j \geq i\} \]

\[ B(j, i, n) = \|n + j\| + B(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\} \]
Inferring Lower Bounds - Static analysis + CAS

\[ B(j, i, n) = 0 \quad \{ j \geq i \} \]
\[ B(j, i, n) = \| n + j \| + B(j', i, n) \quad \{ j + 1 \leq i, j + 1 \leq j' \leq j + 3 \} \]

- An evaluation for \( B(j_0, i_0, n_0) \) looks like:

\[
\begin{align*}
&3 \rightarrow 1 \rightarrow 2 \rightarrow \ldots
\end{align*}
\]
Inferring Lower Bounds - Static analysis + CAS

\[ B(j, i, n) = 0 \]
\[ B(j, i, n) = \|n + j\| + B(j', i, n) \quad \{j \geq i, j + 1 \leq i, j + 1 \leq j' \leq j + 3\} \]

- An evaluation for \( B(j_0, i_0, n_0) \) looks like:

- What is the minimum number of circles?
Inferring Lower Bounds - Static analysis + CAS

\[ B(j, i, n) = 0 \quad \{j \geq i\} \]
\[ B(j, i, n) = \| n + j \| + B(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\} \]

• An evaluation for \( B(j_0, i_0, n_0) \) looks like:

• What is the minimum number of \( \text{circles} \) ? \( \| \frac{i_0 - j_0}{3} \| \)
\[ B(j, i, n) = 0 \quad \text{for} \quad j \geq i \]

\[ B(j, i, n) = \| n + j \| + B(j', i, n) \quad \text{for} \quad j + 1 \leq i, j + 1 \leq j' \leq j + 3 \]

- An evaluation for \( B(j_0, i_0, n_0) \) looks like:

- What is the minimum number of \( \| \frac{i_0 - j_0}{3} \| \)?

- Add loop counter \( \lambda \) to the CR
Inferring Lower Bounds - Static analysis + CAS

\[ B(j, i, n) = 0 \quad \lambda \quad \lambda + 1 \quad \{j \geq i\} \]
\[ B(j, i, n) = \|n + j\| + B(j', i, n) \quad \{j + 1 \leq i, j + 1 \leq j' \leq j + 3\} \]

- An evaluation for \( B(j_0, i_0, n_0) \) looks like:

- What is the minimum number of \( \| \frac{i_0 - j_0}{3} \| \)?
- Add loop counter \( \lambda \) to the CR
- Infer an invariant \( \langle B(j_0, i_0, n_0, 0) \leadsto B(j, i, n, \lambda), \Psi \rangle \)
Inferring Lower Bounds - Static analysis + CAS

\[ B(j, i, n) = 0 \quad \lambda \quad \lambda + 1 \quad \{ j \geq i \} \]
\[ B(j, i, n) = \| n + j \| + B(j', i, n) \quad \{ j + 1 \leq i, j + 1 \leq j' \leq j + 3 \} \]

- An evaluation for \( B(j_0, i_0, n_0) \) looks like:

- What is the minimum number of \( \| \frac{i_0 - j_0}{3} \| \) ?

- Add loop counter \( \lambda \) to the CR

- Infer an invariant \( \langle B(j_0, i_0, n_0, 0) \leadsto B(j, i, n, \lambda), \Psi \rangle \)

- Minimize \( \lambda \) w.r.t \( \Psi \wedge \{ j \geq i \} \) and the parameters \( \langle j_0, i_0, n_0 \rangle \)
\[ B(j, i, n) = 0 \quad \{j \geq i\} \]
\[ B(j, i, n) = \|n + j\| + B(j', i, n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\} \]

- An evaluation for \( B(j_0, i_0, n_0) \) looks like:

\[ \begin{array}{c}
3 \rightarrow 1 \rightarrow 2 \rightarrow \ldots \ldots \rightarrow \\
\end{array} \]

- What is the minimum number of \( \circ \)? \( \| \frac{i_0 - j_0}{3} \| \)

- What is the LB of \( \circ \)? i.e. on \( \| n + j \| \)
\[ B(j, i, n) = 0 \]
\[ B(j, i, n) = \|n + j\| + B(j', i, n) \quad \{j \geq i\} \]
\[ \{j + 1 \leq i, j + 1 \leq j' \leq j + 3\} \]

- An evaluation for \( B(j_0, i_0, n_0) \) looks like:

- What is the minimum number of \( \bigcirc \) ? \( \| \frac{i_0 - j_0}{3} \| \)

- What is the LB of \( \bigtriangleup \)? i.e. on \( \|n + j\| \)
Inferring Lower Bounds - Static analysis + CAS

\[ B(j, i, n) = 0 \{ j \geq i \} \]
\[ B(j, i, n) = \| n + j \| + B(j', i, n) \{ j+1 \leq i, j+1 \leq j' \leq j+3 \} \]

• An evaluation for \( B(j_0, i_0, n_0) \) looks like:

![Diagram with nodes 1, 2, 3 and a triangle marked with *]

• What is the minimum number of \( \| i_0 - j_0 \| \)?

• What is the LB of \( \bigcirc \) i.e. on \( \| n + j \| \)

\[ \bigcirc \quad * \quad \| \frac{i_0 - j_0}{3} \| \]
Inferring Lower Bounds - Static analysis + CAS

\[
B(j, i, n) = 0 \quad \{j \geq i\}
\]

\[
B(j, i, n) = \|n + j\| + B(j', i, n) \quad \{j + 1 \leq i, j + 1 \leq j' \leq j + 3\}
\]

- An evaluation for \(B(j_0, i_0, n_0)\) looks like:

- What is the minimum number of \(\bigcirc\) ? \(\|\frac{i_0 - j_0}{3}\|\)

- What is the LB of \(\bigtriangleup\)? i.e. on \(\|n + j\|\)
\[ B(j, i, n) = 0 \]
\[ B(j, i, n) = \|n + j\| + B(j', i, n) \quad \{j \geq i\} \]
\[ \{j' + 1 \leq i, j + 1 \leq j' \leq j + 3\} \]

\[ \hat{d} \]
\[ \hat{d} \]
\[ \hat{d} \]
\[ \hat{d} \]

\[ P(0) = 0 \]
\[ P(x) = \hat{d} + (\|\frac{i_0 - j_0}{3}\| - x) \hat{d} + P(x - 1) \]
\[ B(j, i, n) = 0 \]
\[ B(j, i, n) = \| n + j \| + B(j', i, n) \quad \{ j \geq i \} \]
\[ \{ j + 1 \leq i, j + 1 \leq j' \leq j + 3 \} \]

\[ P(0) = 0 \]
\[ P(x) = \text{\textbullet} + (\| i_0 - j_0 \| - x) \times \dd + P(x - 1) \]

- \( P(\| i_0 - j_0 \| / 3) \) is the sum of all \( \text{\textbullet} \)
Inferring Lower Bounds - Static analysis + CAS

\[ B(j, i, n) = 0 \]

\[ B(j, i, n) = \| n + j \| + B(j', i, n) \quad \{ j \geq i \} \]

\[ \{ j + 1 \leq i, j + 1 \leq j' \leq j + 3 \} \]

\[ \hat{P}(0) = 0 \]

\[ \hat{P}(\hat{x}) = (\frac{i_0 - j_0}{3} - \hat{x}) * \hat{d} + \hat{P}(\hat{x} - 1) \]

• \( \hat{P}(\frac{i_0 - j_0}{3}) \) is the sum of all

• If \( E \) is a closed-form solution for \( \hat{P}(\hat{x}) \) obtained by CAS, then \( E[\hat{x}/\| \frac{i_0 - j_0}{3} \|] \) is an LB on the best-case of \( B \).
Benchmarks

DetEval

LinEqSolve

MatrixInv

InsertSort

MergeSort

SelectSort

PascalTriangle

BubbleSort

NestedRecIter
Comparing Results with PUBS

![Graph comparing results]

- $F_{ub}(n)$
- $F_{lb}(n)$
- $F_{pubs}(n)$
## Comparison - PUBS vs Static Analysis + CAS

<table>
<thead>
<tr>
<th></th>
<th>PUBS</th>
<th>PUBS + CAS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Precision</strong></td>
<td>★★★☆☆☆☆☆☆</td>
<td>★★★☆☆☆☆☆☆</td>
</tr>
<tr>
<td><strong>Applicability</strong></td>
<td>★★★★★☆☆☆☆</td>
<td>★★★★★☆☆☆☆</td>
</tr>
<tr>
<td><strong>Non-Determinism</strong></td>
<td>★★★★★☆☆☆☆</td>
<td>★★★★★☆☆☆☆</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>★★★★★☆☆☆☆</td>
<td>★★★★★☆☆☆☆</td>
</tr>
<tr>
<td><strong>Lower Bounds</strong></td>
<td>★☆☆☆☆☆☆☆☆</td>
<td>★★★★★☆☆☆☆</td>
</tr>
</tbody>
</table>
## Comparison with RAML (Hoffmann et al. TOPLAS 2012)

<table>
<thead>
<tr>
<th>Function</th>
<th>Formula A</th>
<th>Formula B</th>
<th>Time Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>apAll</td>
<td>$|a| \cdot |b| + 2 \cdot |a| + 1$</td>
<td>$a \cdot b + 2 \cdot a + 1$</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>$\frac{1}{2} \cdot |a|^2 + \frac{5}{2} \cdot |a| + 1$</td>
<td>$\frac{1}{2} a^2 + \frac{3}{2} \cdot a + 1$</td>
<td>46</td>
</tr>
<tr>
<td>isort</td>
<td>$\frac{1}{2} \cdot |a|^2 + \frac{5}{2} \cdot |a| + 1$</td>
<td>$\frac{1}{2} a^2 + \frac{3}{2} \cdot a + 1$</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>$2 \cdot a \cdot b + 2 \cdot a + 1$</td>
<td>$74$</td>
<td></td>
</tr>
<tr>
<td>dyade</td>
<td>$|a| \cdot |b| + 2 \cdot |a| + 1$</td>
<td>$2 \cdot a \cdot b + 2 \cdot a + 1$</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>$2 \cdot |a|^2 + 8 \cdot |a| + 3$</td>
<td>$70$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$4 \cdot a \cdot b + 6 \cdot a + 3$</td>
<td>$193$</td>
<td></td>
</tr>
<tr>
<td>mult3</td>
<td>$\log_2(|2 \cdot a - 3| + 1) \cdot |a - \frac{1}{2}| + 4 \cdot |2 \cdot a - 3| + \log_2(|2 \cdot a - 3| + 1) \cdot |\frac{a}{2}| + 1$</td>
<td>$76$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\frac{7}{2} a^2 - \frac{5}{2} \cdot a + 1$</td>
<td>$73$</td>
<td></td>
</tr>
<tr>
<td>msort</td>
<td>$8 \cdot 2^{|a|} - 2 \cdot |a| - 7$</td>
<td>$83$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$a^2 + 3 \cdot a + 1$</td>
<td>$76$</td>
<td></td>
</tr>
</tbody>
</table>
Summary of Contributions - I

- CAS can be used for solving a small subclass of CRs
- This subclass is not enough when considering imperative languages, with heap, arrays, etc.

\[
\sum_{i=1}^{n} i \text{ is solved to } \frac{(n+1)n}{2} \text{ rather than approximating it by } n^2
\]

Our contributions are: our inferred bounds are very precise. For example, \( \sum_{i=1}^{n} i \) is solved to \( \frac{(n+1)n}{2} \) rather than approximating it by \( n^2 \). Our bounds are precise, yet widely applicable. It allows inferring lower bounds on the best-case behavior.

\[\text{UB in the order of } O(n \cdot \log(n)) \text{ for merge-sort}\]

http://costa.ls.fi.upm.es/pubs
CAS can be used for solving a small subclass of CRs
- This subclass is not enough when considering imperative languages, with heap, arrays, etc.
- Static analysis based solvers have been developed for CRs
  - Trade-off between applicability and precision

\[
\sum_{i=1}^{n} i = \frac{(n+1)n}{2} \text{ rather than approximating it by } n^2
\]

http://costa.ls.fi.upm.es/pubs
Summary of Contributions - I

- CAS can be used for solving a small subclass of CRs
- This subclass is not enough when considering imperative languages, with heap, arrays, etc.
- Static analysis based solvers have been developed for CRs
  - Trade-off between applicability and precision
- Our contributions are:
  - Our inferred bounds are very precise. For example, \( \sum_{i=1}^{n} i \) is solved to \( \frac{(n+1)\cdot n}{2} \) rather than approximating it by \( n^2 \)
  - Precise, yet widely applicable
  - Applicable to linear, geometric, etc, progression behavior
  - We obtain an UB in the order of \( O(n \cdot \log(n)) \) for merge-sort
  - It allows inferring lower bounds on the best-case
- http://costa.ls.fi.upm.es/pubs
Our contributions are published in the following papers:


1 Introduction

2 Background on Cost Analysis

3 Precise Cost Analysis Techniques

4 Theoretical Complexity of Deciding Termination

5 Conclusions
Integer linear constraints loops - motivation

\[ C(k, j, n) = 0 \quad \{ k \geq n+j \} \]
\[ C(k, j, n) = 1 + C(k', j, n) \quad \{ k' = k+1, k+1 \leq n+j \} \]
Integer linear constraints loops - motivation

\[ C(k, j, n) = 0 \quad \{ k \geq n+j \} \]
\[ C(k, j, n) = 1 + C(k', j, n) \quad \{ k' = k+1, k+1 \leq n+j \} \]
**INTEGER LINEAR CONSTRAINTS LOOPS - MOTIVATION**

\[
C(k, j, n) = 0 \quad \{k \geq n + j\}
\]

\[
C(k, j, n) = 1 + C(k', j, n) \quad \{k' = k + 1, k + 1 \leq n + j\}
\]

\[\psi\]

**INTEGER LINEAR-CONSTRAINT (ILC) loops**
Given a program P, decide whether it will finish running or could run forever
The Termination Problem
The Termination Problem

The diagram shows a graph with two axes: INPUT and PROGRAM. The program shown is:

```java
while (i<j) {
  ...
  i = i+1;
}
```
The Termination Problem

Input

Computer program

Transition system

Program

while (i<j) {
    
    
    i = i+1;

}
The Termination Problem

while (i<j) {
    ... 
    i = i+1;
}

x' = x \wedge y' < y

x' = x \wedge y' = y

x' < x

Abu Naser Masud, UPM
Termination and Cost Analysis: Complexity and Precision Issues
The Termination Problem

```
while (i<j) {
    ...
    i = i+1;
}
```
THE TERMINATION PROBLEM

**Given input**

- Computer program
- Transition system
- Counter program
- Turing machine

**Any input**

```
while (i<j) {
    ...
    i = i+1;
}
```

**Diagram**

- Node A: $x' = x \land y' < y$
- Node B: $x' = x \land y' = y$
- Node C: $x' < x$

Abu Naser Masud, UPM
Termination and Cost Analysis: Complexity and Precision Issues
26/34
**The Termination Problem**

```
while (i<j) {
    ...  
i = i+1;
}
```

---

**INPUT**
- Partial input
- Given input
- Any input

**PROGRAM**
- Computer program
- Transition system
- Counter program
- Turing machine

---

Given input

---

Any input

---

Partial input

---

while (i<j) {
    ...
i = i+1;
}
The Termination Problem

A

B

C

universal vs. mortality

Any input

Given input

Partial input

while (i<j) {
    ...
    i = i+1;
}
The Termination Problem

\textbf{while} (i<j) {
  ...
  i = i+1;
}\}

\textbf{DOMAIN} \rightarrow \textbf{PROGRAM}

\textbf{INPUT}

- Partial input
- Given input
- Any input

Computer program
Transition system
Counter program
Turing machine

Any input

\(x'=x \land y'<y\)
\(x'=x \land y'=y\)
\(x'<x\)

Abu Naser Masud, UPM
while ( x >= 0 && y >= 0 && 2*z+w >= 3 && ... ) {
    x=x-y;
    y=y-1;
    ... 
}

```cpp
while (Bx > b ∧ Dx ≥ d) x = Ax + c
```

For any input, it is decidable over \( \mathbb{R} \) and \( \mathbb{Q} \); and over \( \mathbb{Z} \) in the homogeneous case (\( b = 0, c = 0, d = 0 \)).
\textbf{Integer Linear while loops}

\begin{verbatim}
while ( x >= 0 && y >= 0 && 2*z+w >= 3 && ... ) {
    x=x-y;
    y=y-1;
    ...
}
\end{verbatim}

- \textbf{while (Bx>b) x=Ax+c} \quad \textbf{[TIWARI’04]}
  - For any input, it is decidable over $\mathbb{R}$
- \textbf{while (Bx>b \land Dx\geq d) x=Ax+c} \quad \textbf{[BRAVERMAN’05]}
  - For any input, it is decidable over $\mathbb{R}$ and $\mathbb{Q}$; and
  - over $\mathbb{Z}$ in the homogeneous case ($b=0,c=0,d=0$)
**Integer Linear While loops**

```plaintext
while ( x >= 0 && y >= 0 && 2*z+w >= 3 && ... ) {
    x=x-y;
    y=y-1;
    ...
}
```

- **while (Bx>b) x=Ax+c**
  - For *any input*, it is decidable over $\mathbb{R}$  
  - **while (Bx>b ∧ Dx≥d) x=Ax+c**  
    - For *any input*, it is decidable over $\mathbb{R}$ and $\mathbb{Q}$; and
    - over $\mathbb{Z}$ in the homogeneous case ($b=0,c=0,d=0$)

**Integer Linear While (ILW) loops**
Termination of Integer Linear while loops

Termination of ILW Loops
Termination of Integer Linear while loops

Termination of ILW Loops

- general cases
- special cases
**Termination of Integer Linear while loops**

*Termination of ILW Loops*

**IPLW Loops**: allow using the instruction $y = \text{isPositive}(x)$ where

$$
isPositive(x) = \begin{cases} 
0 & x \leq 0 \\
1 & x > 0 
\end{cases}
$$

**Termination of ILW Loops**

general cases

special cases
**Termination of Integer Linear while loops**

**IPLW Loops**: allow using the instruction $y = \text{isPositive}(x)$ where

$$\text{isPositive}(x) = \begin{cases} 
0 & x \leq 0 \\
1 & x > 0 
\end{cases}$$

**Termination of ILW Loops**

- **General cases**
- **Special cases**
Termination of Integer Linear while loops

General cases

Termination of ILW loops

IPLW loops: allow using the instruction \( y = \text{isPositive}(x) \) where

\[
\text{isPositive}(x) = \begin{cases} 
0 & x \leq 0 \\
1 & x > 0 
\end{cases}
\]

UNDECIDABLE

isPositive(x) can be simulated by Integer Division by Constant

\[
\text{isPositive}(x) = x - \frac{2 \cdot x - 1}{2}
\]
Termination of Integer Linear While Loops

The proof is done by a reduction from the termination of counter programs

1: x = x - 1
2: y = y + 1
3: if x > 0 then 1 else 4
4: end
The proof is done by a reduction from the termination of counter programs.

1. \( x = x - 1 \)
2. \( y = y + 1 \)
3. if \( x > 0 \) then 1 else 4
4. end
The proof is done by a reduction from the termination of counter programs

```
1: x=x-1
2: y=y+1
3: if x>0 then 1 else 4
4: end
```
The proof is done by a reduction from the termination of counter programs

1: x=x-1
2: y=y+1
3: if x>0 then 1 else 4
4: end
The proof is done by a reduction from the termination of counter programs

1: \( x = x - 1 \)
2: \( y = y + 1 \)
3: if \( x > 0 \) then 1 else 4
4: end
The proof is done by a reduction from the termination of counter programs

1: x = x - 1
2: y = y + 1
3: if x > 0 then 1 else 4
4: end
The proof is done by a reduction from the termination of counter programs:

1: \( x = x - 1 \)
2: \( y = y + 1 \)
3: if \( x > 0 \) then 1 else 4
4: end
The proof is done by a reduction from the termination of counter programs

1: x = x - 1  
2: y = y + 1  
3: if x > 0 then 1 else 4  
4: end

Termination for a given input and mortality are undecidable for counter programs
The proof is done by a reduction from the termination of counter programs

1: \( x = x - 1 \)
2: \( y = y + 1 \)
3: \( \text{if } x > 0 \text{ then } 1 \text{ else } 4 \)
4: \( \text{end} \)

while \((A_1 \geq 0 \&\& \cdots \&\& A_3 \geq 0 \&\& A_1 + \cdots + A_3 = 1 \&\& x \geq 0 \&\& y \geq 0)\) 
{

}
The proof is done by a reduction from the termination of counter programs:

1: \( x = x - 1 \)
2: \( y = y + 1 \)
3: \( \text{if } x > 0 \text{ then } 1 \text{ else } 4 \)
4: \( \text{end} \)

\[
\text{while } (A_1 \geq 0 \land \cdots \land A_3 \geq 0 \land A_1 + \cdots + A_3 = 1 \land x \geq 0 \land y \geq 0)
\{
\}
\]
The proof is done by a reduction from the termination of counter programs

1: \( x = x - 1 \)
2: \( y = y + 1 \)
3: if \( x > 0 \) then 1 else 4
4: end

while \( (A_1 \geq 0 \land \cdots \land A_3 \geq 0 \land A_1 + \cdots + A_3 = 1 \land x \geq 0 \land y \geq 0) \)
{x := x - A_1;
Termination of Integer Linear While Loops

The proof is done by a reduction from the termination of counter programs:

1: \( x = x - 1 \)
2: \( y = y + 1 \)
3: if \( x > 0 \) then 1 else 4
4: end

while \( (A_1 \geq 0 \land \cdots \land A_3 \geq 0 \land A_1 + \cdots + A_3 = 1 \land x \geq 0 \land y \geq 0) \) {

\( x := x - A_1 \);
\( y := y + A_2 \);

}
The proof is done by a reduction from the termination of counter programs

1: $x = x - 1$
2: $y = y + 1$
3: if $x > 0$ then 1 else 4
4: end

while $(A_1 >= 0 \land \cdots \land A_3 >= 0 \land A_1 + \cdots + A_3 = 1 \land x >= 0 \land y >= 0)$
{
    $N_0 := 0; N_1 := A_1; N_2 := A_2; N_3 := A_3;$
    
    $x := x - A_1;$
    $y := y + A_2;$

    $A_1 := N_0; A_2 := N_1; A_3 := N_2;$
}

Abu Naser Masud, UPM
The proof is done by a reduction from the termination of counter programs

1: x=x-1
2: y=y+1
3: if x>0 then 1 else 4
4: end

while (A_1 >= 0 && \ldots && A_3 >= 0 && A_1+\ldots+A_3=1 && x>=0 && y>=0) {
  N_0:=0; N_1:=A_1; N_2:=A_2; N_3=A_3;
  F_1:=isPositive(x);
  x:=x-A_1;
  y:=y+A_2;
  T_3:=isPositive(A_3+F_1-1);
  R_3:=isPositive(A_3-F_1);
  N_3:=N_3-A_3;
  N_0:=N_0+T_3;
  N_3:=N_3+R_3;
  A_1:=N_0; A_2:=N_1; A_3:=N_2;
}
The proof is done by a reduction from the termination of counter programs

1: \( x = x - 1 \)
2: \( y = y + 1 \)
3: \( \text{if } x > 0 \text{ then } 1 \text{ else } 4 \)
4: \( \text{end} \)

while \( (A_1 \geq 0 \&\& \cdots \&\& A_3 \geq 0 \&\& A_1 + \cdots + A_3 = 1 \&\& x \geq 0 \&\& y \geq 0) \)
{ 
\( N_0 := 0; N_1 := A_1; N_2 := A_2; N_3 := A_3; \) 
\( \text{isPositive}(x); \) 
\( N_3 := N_3 + R_3; \) 
\( A_1 := N_0; A_2 := N_1; A_3 := N_2; \) 
}

Termination, for any or a given input, of IPLW loops is UNDECIDABLE
Termination of Integer Linear While loops

- IPLW Loops
  - general cases
  - special cases

Termination of ILW Loops
Termination of Integer Linear while loops

Special cases

General cases

Undecidable for two linear pieces

while ( CONDITION ) {
    if ( x > 0 ) then B_1 else B_2
}

Termination of ILW Loops

Abu Naser Masud, UPM

Termination and Cost Analysis: Complexity and Precision Issues
Termination of Integer Linear while loops

Termination of ILW Loops

- general cases
- special cases

Undecidable for two linear pieces

while ( CONDITION ) {
    if ( x > 0 ) then B_1 else B_2
}

[Bozga et al., TACAS’12]
Termination of Integer Linear-Constraint loops

- General cases
- Special cases
An **INTEGER LINEAR WHILE** loop can be translated into an equivalent **INTEGER LINEAR-CONSTRAINT** loop.

```java
while (x >= 0) {
    x = x + y;
    y = y - 1;
}
```

\[
x \geq 0 \land 
\begin{align*}
    x' &= x + y \\
    y' &= y - 1
\end{align*}
\]
Termination of Integer Linear-Constraint loops

- An **INTEGER LINER WHILE** loop can be translated into an equivalent **INTEGER LINEAR-CONSTRAINT** loop

```plaintext
while (x >= 0) {
    x = x + y;
    y = y - 1;
}
```

- Then, if we succeed to simulate the `isPositive` function with integer linear constraints we prove undecidability
An **INTEGER LINEAR WHILE** loop can be translated into an equivalent **INTEGER LINEAR-CONSTRAINT** loop

```
while (x>=0) {
    x=x+y;
    y=y-1;
}
```

Then, if we succeed to simulate the **isPositive** function with integer linear constraints we prove undecidability

\[ T_k = \text{isPositive}(A_k + F_k - 1) \]
An **INTEGER LINEAR WHILE loop** can be translated into an equivalent **INTEGER LINEAR-CONSTRAINT loop**

```
while (x>=0) {
    x=x+y;
    y=y-1;
}
```

Then, if we succeed to simulate the **isPositive** function with integer linear constraints we prove undecidability

\[ T_k = \text{isPositive}(A_k + F_k - 1) \]

\[ F_k + A_k - 1 \leq 2 \cdot T_k \leq F_k + A_k \]
Termination of Integer Linear-Constraint loops

- An **INTEGER LINEAR WHILE** loop can be translated into an equivalent **INTEGER LINEAR-CONSTRAINT** loop

```plaintext
while (x >= 0) {
    x = x + y;
    y = y - 1;
}
```

- Then, if we succeed to simulate the `isPositive` function with integer linear constraints we prove undecidability

```plaintext
R_k = isPositive(A_k - F')
```
Termination of Integer Linear-Constraint loops

- An **integer linear while** loop can be translated into an equivalent **integer linear-constraint** loop

```
while (x >= 0) {
    x = x + y;
    y = y - 1;
}
```

Then, if we succeed to simulate the `isPositive` function with integer linear constraints we prove undecidability

\[ R_k = \text{isPositive}(A_k - F') \]

\[ A_k - F \leq 2 \cdot R_k \leq A_k - F + 1 \land 0 \leq R_k \leq 1 \]
An integer linear while loop can be translated into an equivalent integer linear-constraint loop

\[
\text{while } (x \geq 0) \{
\begin{align*}
    x &= x + y; \\
    y &= y - 1;
\end{align*}
\}
\]

Then, if we succeed to simulate the \texttt{isPositive} function with integer linear constraints we prove undecidability

\[
F_k = \text{isPositive}(x)
\]
An \text{INTEGER LINEAR WHILE} loop can be translated into an equivalent \text{INTEGER LINEAR-CONSTRAINT} loop

\begin{verbatim}
while (x>=0) {
    x=x+y;
    y=y-1;
}
\end{verbatim}

\[
\begin{align*}
x \geq 0 \land \\
x' &= x + y \\
y' &= y - 1
\end{align*}
\]

Then, if we succeed to simulate the \text{isPositive} function with integer linear constraints we prove undecidability

\[F_k = \text{isPositive}(x)\]

\[
\Psi \land x = 0 \rightarrow F_k = 0 \land \\
\Psi \land x \geq 1 \rightarrow F_k = 1
\]
After many (failing) attempts, the best we could get is

\[ \Psi_1 \equiv F \leq x \land 0 \leq F \leq 1 \]
After many (failing) attempts, the best we could get is

$$\Psi_1 \equiv F \leq x \land 0 \leq F \leq 1$$
After many (failing) attempts, the best we could get is

\[ \Psi_1 \equiv F \leq x \land 0 \leq F \leq 1 \]

\[ \text{isPositive}(x) = \lceil \sqrt{2} \cdot x \rceil + \lceil -\sqrt{2} \cdot x \rceil \]
Termination of ILC Loops – The search for $\Psi$

After many (failing) attempts, the best we could get is

$$\Psi_1 \equiv F \leq x \land 0 \leq F \leq 1$$

isPositive($x$) = $\lceil \sqrt{2} \cdot x \rceil + \lceil -\sqrt{2} \cdot x \rceil$

$$\Psi_2 \equiv -\sqrt{2} \cdot x \leq B \leq -\sqrt{2} \cdot x + 1 \land F = A + B$$
After many (failing) attempts, the best we could get is

$$\Psi_1 \equiv F \leq x \land 0 \leq F \leq 1$$

isPositive\( (x) = \lceil \sqrt{2} \cdot x \rceil + \lceil -\sqrt{2} \cdot x \rceil$$

$$\Psi_2 \equiv -\sqrt{2} \cdot x \leq B \leq -\sqrt{2} \cdot x + 1 \land F = A + B$$
After many (failing) attempts, the best we could get is

\[ \Psi_1 \equiv F \leq x \land 0 \leq F \leq 1 \]

\[ \text{isPositive}(x) = \lfloor \sqrt{2} \cdot x \rfloor + \lceil -\sqrt{2} \cdot x \rceil \]

\[ \Psi_2 \equiv \sqrt{2} \cdot x \leq A \leq \sqrt{2} \cdot x + 1 \quad \land \]
\[ -\sqrt{2} \cdot x \leq B \leq -\sqrt{2} \cdot x + 1 \quad \land \]
\[ F = A + B \]

\[ F \]

\[ 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad \ldots \]

\[ x \]
After many (failing) attempts, the best we could get is

$$\Psi_1 \equiv F \leq x \land 0 \leq F \leq 1$$

isPositive(x) = \lceil \sqrt{2} \cdot x \rceil + \lceil -\sqrt{2} \cdot x \rceil

$$\Psi_2 \equiv \sqrt{2} \cdot x \leq A \leq \sqrt{2} \cdot x + 1 \land -\sqrt{2} \cdot x \leq B \leq -\sqrt{2} \cdot x + 1 \land F = A + B$$

$$\Psi \equiv \Psi_1 \land \Psi_2$$
After many (failing) attempts, the best we could get is

\[ \Psi_1 \equiv F \leq x \land 0 \leq F \leq 1 \]

isPositive(x) = \[ \lceil \sqrt{2} \cdot x \rceil + \lceil -\sqrt{2} \cdot x \rceil \]

\[ \sqrt{2} \cdot x \leq A \leq \sqrt{2} \cdot x + 1 \land \]
\[ \Psi_2 \equiv -\sqrt{2} \cdot x \leq B \leq -\sqrt{2} \cdot x + 1 \land \]
\[ F = A + B \]

\[ \Psi \equiv \Psi_1 \land \Psi_2 \]

\[ a_0 + a_1 \cdot x_1 + \cdots + a_n \cdot x_n \leq 0, \text{ all constants are taken from } \mathbb{Q} \]
After many (failing) attempts, the best we could get is

\[ \Psi_1 \equiv F \leq x \land 0 \leq F \leq 1 \]

\[ \text{isPositive}(x) = \lceil \sqrt{2} \cdot x \rceil + \lceil -\sqrt{2} \cdot x \rceil \]

\[
\sqrt{2} \cdot x \leq A \leq \sqrt{2} \cdot x + 1 \quad \land \\
\Psi_2 \equiv -\sqrt{2} \cdot x \leq B \leq -\sqrt{2} \cdot x + 1 \quad \land \\
F = A + B
\]

\[ \Psi \equiv \Psi_1 \land \Psi_2 \]

\[ a_0 + a_1 \cdot x_1 + \cdots + a_n \cdot x_n \leq 0, \text{ all constants are taken from } \mathbb{Q} \]

Undecidable when constants are taken from \( \mathbb{Q} \cup \{ \sqrt{2}, -\sqrt{2} \} \)
After many (failing) attempts, the best we could get is
\[
\Psi_1 \equiv F \leq x \land 0 \leq F \leq 1
\]

\[
isPositive(x) = \lceil \sqrt{2} \cdot x \rceil + \lceil -\sqrt{2} \cdot x \rceil
\]
\[
\Psi_2 \equiv -\sqrt{2} \cdot x \leq B \leq -\sqrt{2} \cdot x + 1 \land F = A + B
\]
\[
\Psi \equiv \Psi_1 \land \Psi_2
\]
\[
a_0 + a_1 \cdot x_1 + \cdots + a_n \cdot x_n \leq 0, \text{ all constants are taken from } \mathbb{Q}
\]
\[
\text{Undecidable when constants are taken from } \mathbb{Q} \cup \{\sqrt{2}, -\sqrt{2}\}
\]
\[
\mathbb{Q} \cup \{r\} \text{ for any irrational } r
\]
Termination of ILC Loops – The search for $\Psi$

Why we succeeded with $\mathcal{Q} \cup \{r\}$ but not with $\mathcal{Q}$?
**Termination of ILC Loops – The search for $\Psi$**

**Why we succeeded with $Q \cup \{r\}$ but not with $Q$?**

![Graphs showing comparison between $Q \cup \{r\}$ and $Q$.](image)
Termination of ILC Loops – The search for $\Psi$

Why we succeeded with $\mathbb{Q} \cup \{r\}$ but not with $\mathbb{Q}$?

It is not possible to represent $F = \text{isPositive}(x)$ with integer linear constraints $\Psi$ with rational constants.

Corollary
Termination of ILC Loops – a lower bound

- General cases
  - Undecidable, for any or a given input, when the constants are taken from $\mathbb{Q} \cup \{r\}$

- Special cases
Termination of ILC loops – a lower bound

Termination of ILC Loops

General cases

Undecidable, for any or a given input, when the constants are taken from \( \mathbb{Q} \cup \{r\} \)

Special cases

Decidable for octagons [Bozga et al. TACAS’09, Iosif et al. TACAS’12]
Termination of ILC loops – a lower bound

Termination of ILC Loops

- General cases
  - Undecidable, for any or a given input, when the constants are taken from \( \mathbb{Q} \cup \{r\}\)

- Special cases
  - Termination for a given input is at least EXPSPACE-hard
  - Decidable for octagons [Bozga et. al. TACAS’09, Iosif et. al. TACAS’12]
Termination of ILC loops – a lower bound

Termination of ILC Loops

General cases

Undecidable, for any or a given input, when the constants are taken from $\mathbb{Q} \cup \{r\}$

Termination for a given input is at least EXPSPACE-hard

Special cases

Decidable for octagons [Bozga et. al. TACAS’09, Iosif et. al. TACAS’12]
**Termination of ILC Loops – a lower bound**

**General cases**

- **Undecidable**, for any or a given input, when the constants are taken from \( \mathbb{Q} \cup \{ r \} \)

**Special cases**

- **Decidable** for octagons [Bozga et al. TACAS’09, Iosif et al. TACAS’12]
- Termination for a given input is at least EXPSPACE-hard
- Still at least EXPSPACE-hard for deterministic constraints, but for partial input
Termination

Integer Linear While

loops

undecidable when allowing "minimal" amount of non-linearity

isPositive

Termination of Integer Linear constraints

it is not possible to model isPositive with rational constants

but possible when allowing a single irrational constant

this leaves some hope for a positive answer, ...

EXPSPACE-hard lower bound by simulating Petri-nets

still EXPSPACE-hard for deterministic constraints
Termination of 
**Integer Linear While** loops
Termination **INTEGER LINEAR WHILE** loops

- undecidable when allowing “minimal” amount of non-linearity
- integer division by constant, ... *isPositive(x)*
Summary of Contributions - II

- **Termination** of Integer Linear While loops
  - undecidable when allowing “minimal” amount of non-linearity
  - integer division by constant, ... \( \text{isPositive}(x) \)

- **Termination of** Integer Linear Constraints loops
Termination of **Integer Linear While** loops

- undecidable when allowing “minimal” amount of non-linearity
- integer division by constant, ... `isPositive(x)`

Termination of **Integer Linear Constraints** loops

- it is not possible to model `isPositive(x)` with rational constants
Termination of Integer Linear While loops
- undecidable when allowing “minimal” amount of non-linearity
- integer division by constant, ... isPositive(x)

Termination of Integer Linear Constraints loops
- it is not possible to model isPositive(x) with rational constants
- but possible when allowing a single irrational constant
- this leaves some hope for a positive answer, ...
Termination of \textit{Integer Linear While} loops
- undecidable when allowing “minimal” amount of non-linearity
- integer division by constant, \ldots isPositive(x)

Termination of \textit{Integer Linear Constraints} loops
- it is not possible to model isPositive(x) with rational constants
- but possible when allowing a \textit{single} irrational constant
- this leaves some hope for a positive answer, \ldots
- EXPSPACE-hard lower bound by simulating Petri-nets
The undecidability of termination for IPLW loops implies that there are certain classes of programs for which inference of cost bounds is not decidable.

Undecidability of termination for IPLW loops with two linear pieces implies solving CRs having two recursive equations is undecidable.

The EXPSPACE-hardness lower bound for ILC loops with a given or partially specified input implies that solving CRs having a single recursive equation, when the input is (partiality) specified, is also at least EXPSPACE-hard.

Outline

1 Introduction

2 Background on Cost Analysis

3 Precise Cost Analysis Techniques

4 Theoretical Complexity of Deciding Termination

5 Conclusions
Conclusion and Future Work

We have considered the practical and theoretical aspects of cost and termination analysis.

Possible extensions of this work would be to consider more expressive abstract programs possibly with nonlinear constraints and inferring techniques for solving those abstract programs.

Another possible extension would be to consider solving open problems regarding termination of ILW and ILC loops.
Conclusion and Future Work

- We have considered the practical and theoretical aspects of cost and termination analysis.

- As practical aspects, we provide cost analysis techniques from CRs that are precise, scalable and have wider applicability.
We have considered the practical and theoretical aspects of cost and termination analysis.

As practical aspects, we provide cost analysis techniques from CRs that are precise, scalable and have wider applicability.

The theoretical study on termination analysis gives us more insight into the termination as well as into the cost analysis in general.

Possible Extension of this work would be to consider more expressive abstract programs possibly with nonlinear constraints and inferring techniques for solving those abstract programs. Another possible extension would be to consider solving open problems regarding termination of ILW and ILC loops.
Conclusion and Future Work

- We have considered the practical and theoretical aspects of cost and termination analysis.
- As practical aspects, we provide cost analysis techniques from CRs that are precise, scalable and have wider applicability.
- The theoretical study on termination analysis gives us more insight into the termination as well as into the cost analysis in general.
- Possible Extension of this work would be to consider more expressive abstract programs possibly with nonlinear constraints and inferring techniques for solving those abstract programs.
Conclusion and Future Work

- We have considered the practical and theoretical aspects of cost and termination analysis.
- As practical aspects, we provide cost analysis techniques from CRs that are precise, scalable and have wider applicability.
- The theoretical study on termination analysis gives us more insight into the termination as well as into the cost analysis in general.
- Possible Extension of this work would be to consider more expressive abstract programs possibly with nonlinear constraints and inferring techniques for solving those abstract programs.
- Another possible extension would be to consider solving open problems regarding termination of ILW and ILC loops.