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)

Outline

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

What is Cost Analysis?

static

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

What is Cost Analysis?

automatic

static

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

What is Cost Analysis?

automatic static

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

What is Cost Analysis?

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

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

WHAT IS COST ANALYSIS?

The aim of ~~static~~ *automatic* COST ANALYSIS is to estimate the ~~bound~~ of resource consumption (aka cost) of executing a given program P on ~~a given~~ *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?

automatic
static

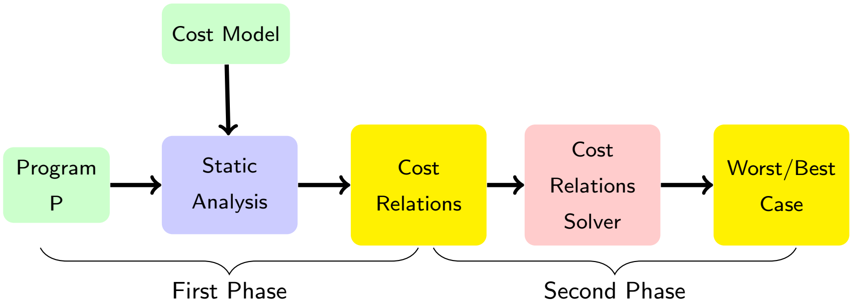
The aim of COST ANALYSIS is to estimate the bound of resource consumption (aka cost) of executing a given program P on ~~a given~~ any input data *without actually executing* P

What is Cost Analysis?

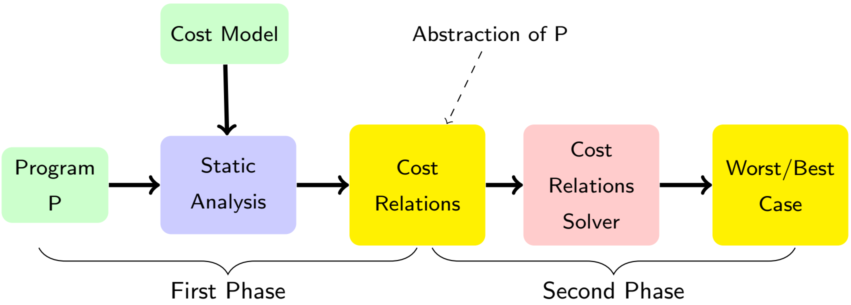
The aim of ~~static~~ *automatic* *static* COST ANALYSIS is to estimate the bound of resource consumption (aka cost) of executing a given program P on ~~a given~~ *any* input data *without actually executing* P

- Execution steps
- Visits to specific program points
- Memory (possibly with garbage collection)

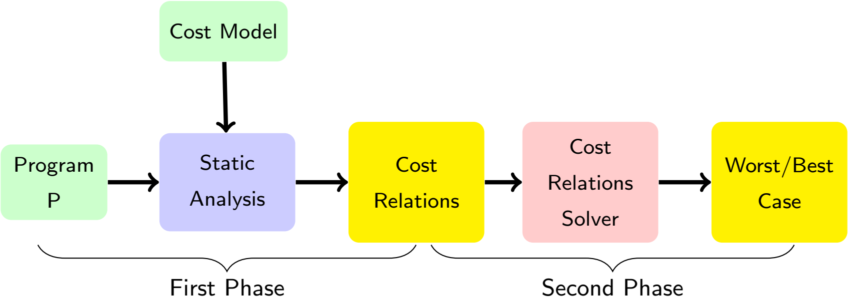
Phases in Cost Analysis



Phases in Cost Analysis



Phases in Cost Analysis



Quality of the solution is affected by the

- Precision issue in the first phase AND
- precision-applicability tradeoff in the second phase

Precision, Applicability and Expressiveness

- CRs are abstract programs and there is no unified terminology or syntax of these abstract programs in cost analysis

Precision, Applicability and Expressiveness

- 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

Precision, Applicability and Expressiveness

- CRs [partially obscured] no unified terr[partially obscured] programs in cost ana[partially obscured]

- Som[partially obscured]
  1. [obscured]
  2. [obscured] ols
  3. but the applicability is limited
  4. Example - recurrence relations which are solved using symbolic computations

$$
\begin{aligned}
P(0) &= 0 \\
P(x) &= x + P(x-1)
\end{aligned}
$$

Precision, Applicability and Expressiveness

- 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

Precision, Applicability and Expressiveness

- CRs are abstract programs and there is no unified

$$
\begin{aligned}
P(x,y) &= 0 && \{x = 0, y = 0\} \\
P(x,y) &= x + P(x',y') && \{x > 0, x' < x, y' = y\}
\end{aligned}
$$

  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

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.

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.

- Computin[...]mination problems [...]

Cost analysis requires inferring the bound on loop iterations or recursion depths.

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.

- Computin … mination problems … he … ur-

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 arrise 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 arrise in the context of cost analysis.

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

Outline

```
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;
  }
}
```

```
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)

$n_0+j_0-k_0$

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)

$n_0+j_0-k_0$

$(i_0-j_0)*(n_0+i_0-1)$

BEST-CASE (LB)

$n_0+j_0-k_0$

$\frac{i_0-j_0}{3}*(n_0+i_0-1)$

```
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)

$n_0+j_0-k_0$

$(i_0-j_0)*(n_0+i_0-1)$

$\frac{n_0-i_0}{2}*(n_0-1)*(2n_0-2)$

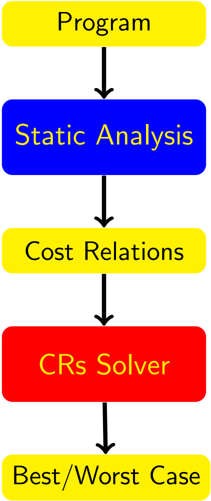
Best-Case (LB)

$n_0+j_0-k_0$

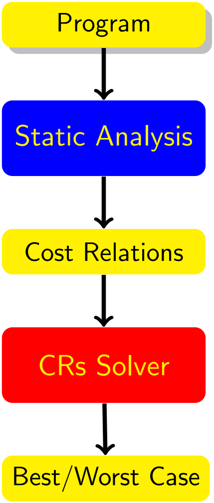
$\frac{i_0-j_0}{3}*(n_0+i_0-1)$

$\frac{n_0-i_0}{4}*\frac{n_0-1}{3}*(2n_0-2)$

COST ANALYSIS - WEGBREIT 1975

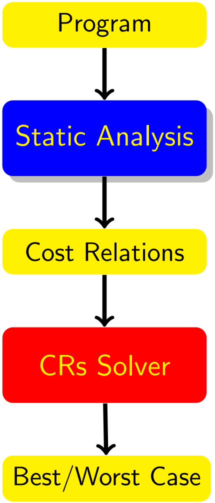


COST ANALYSIS - WEGBREIT 1975

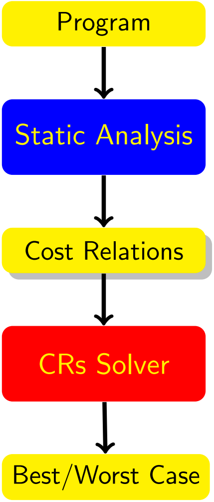



```
while ( i<n ) {
  int j=0;
  while ( j<i ) {
    for(int k=0;k<n+j;k++)
      l=new List(i*k*j,l);
    j+= (*) ? 1:3;
  }
  i+= (*) ? 2:4;
}
```

COST ANALYSIS - WEGBREIT 1975



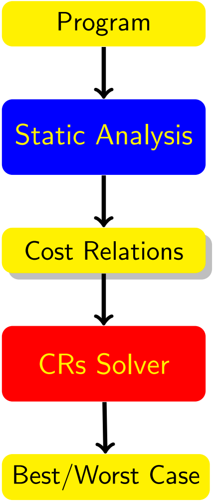

```
while ( i<n ) {
  int j=0;
  while ( j<i ) {
    for(int k=0;k<n+j;k++)
      l=new List(i*k*j,l);
    j+= (*) ? 1:3;
  }
  i+= (*) ? 2:4;
}
```




```
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;
}
```

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

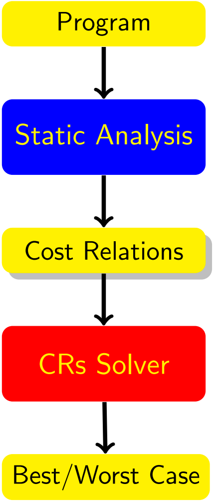
COST ANALYSIS - WEGBREIT 1975




```
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;
}
```

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

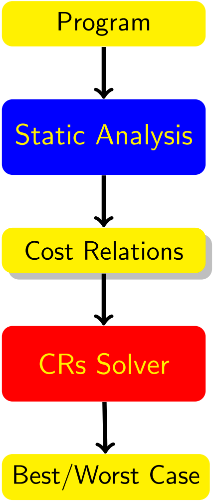
COST ANALYSIS - WEGBREIT 1975




```
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;
}
```

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

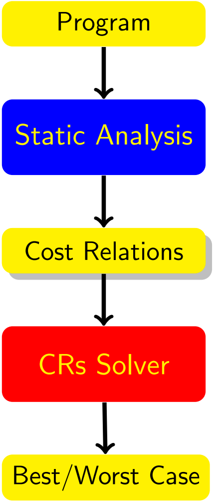
COST ANALYSIS - WEGBREIT 1975




```
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;
}
```

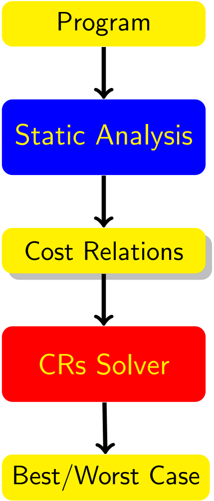
$$
\begin{array}{llll}
A(i,n) & = & 0 & \{i \geq n\} \\
A(i,n) & = & B(0,i,n)+A(i',n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\} \\
 & & & \\
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(j,i,n) & = & C(0,j,n)+B(j',i,n) & \{j+1 \leq i, j+1 \leq j' \leq j+3\} \\
 & & & \\
C(k,j,n) & = & 0 & \{k \geq n+j\} \\
C(k,j,n)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

COST ANALYSIS - WEGBREIT 1975




```
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;
}
```

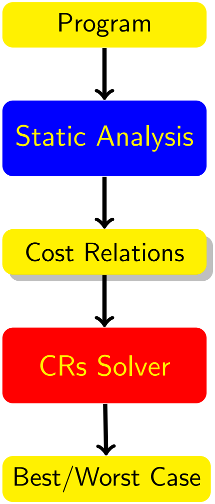
$$
\begin{array}{llll}
A(i,n) & = & 0 & \{i \geq n\} \\
A(i,n) & = & B(0,i,n)+A(i',n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\} \\
 & & & \\
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(j,i,n) & = & C(0,j,n)+B(j',i,n) & \{j+1 \leq i, j+1 \leq j' \leq j+3\} \\
 & & & \\
C(k,j,n) & = & 0 & \{k \geq n+j\} \\
C(k,j,n)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




```
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;
}
```

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

COST ANALYSIS - WEGBREIT 1975

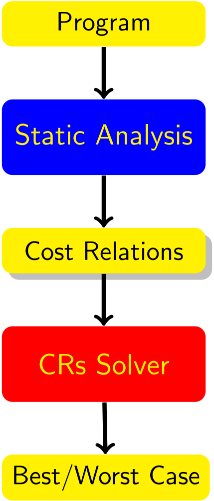



```
while ( i<n ) {
  int j=0;
  while ( j<i ) {
    for(int k=0;k<n+j;k++)
      l=new List(i*k*j,l);
    j+= (*) ? 1:3;
  }
  i+= (*) ? 2:4;
}
```

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

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

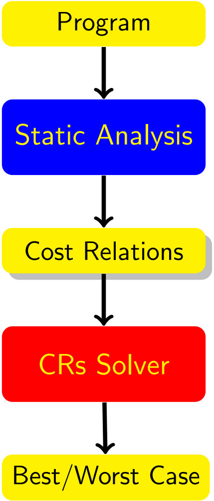
$$
\begin{aligned}
C(k,j,n) &= 0 && \{k \geq n+j\} \\
C(k,j,n) &= 1 + C(k',j,n) && \{k' = k+1,\ k+1 \leq n+j\}
\end{aligned}
$$




```
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;
}
```

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

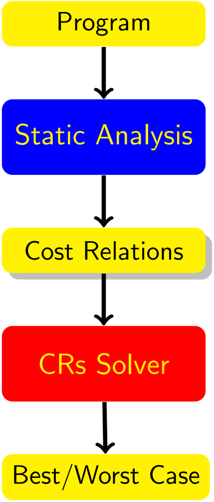
Cost analysis - Wegbreit 1975




```
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;
}
```

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

COST ANALYSIS - WEGBREIT 1975



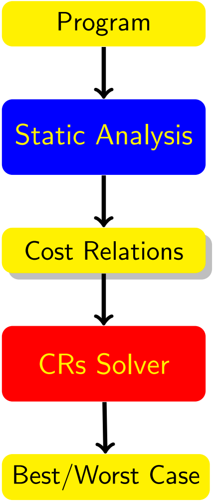

```
while ( i<n ) {
  int j=0;
  while ( j<i ) {
    for(int k=0;k<n+j;k++)
      l=new List(i*k*j,l);
    j+= (*) ? 1:3;
  }
  i+= (*) ? 2:4;
}
```

$$
\begin{array}{llll}
A(i,n) & = & 0 & \{i \geq n\} \\
A(i,n) & = & B(0,i,n)+A(i',n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\}
\end{array}
$$

$$
\begin{array}{llll}
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(j,i,n) & = & C(0,j,n)+B(j',i,n) & \{j+1 \leq i, j+1 \leq j' \leq j+3\}
\end{array}
$$

$$
\begin{array}{llll}
C(k,j,n) & = & 0 & \{k \geq n+j\} \\
C(k,j,n) & = & 1+C(k',j,n) & \{k' = k+1, k+1 \leq n+j\}
\end{array}
$$

COST ANALYSIS - WEGBREIT 1975



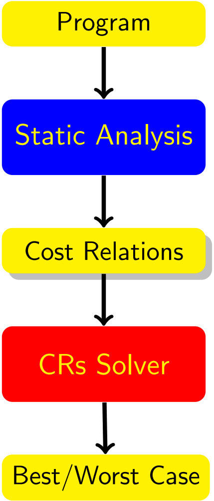

```
while ( i<n ) {
  int j=0;
  while ( j<i ) {
    for(int k=0;k<n+j;k++)
      l=new List(i*k*j,l);
    j+= (*) ? 1:3;
  }
  i+= (*) ? 2:4;
}
```

$$
\begin{array}{lcll}
A(i,n) & = & 0 & \{i \geq n\} \\
A(i,n) & = & B(0,i,n) + A(i',n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\}
\end{array}
$$

$$
\begin{array}{lcll}
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(j,i,n) & = & C(0,j,n) + B(j',i,n) & \{j+1 \leq i, j+1 \leq j' \leq j+3\}
\end{array}
$$

$$
\begin{array}{lcll}
C(k,j,n) & = & 0 & \{k \geq n+j\} \\
C(k,j,n) & = & 1 + C(k',j,n) & \{k' = k+1, k+1 \leq n+j\}
\end{array}
$$

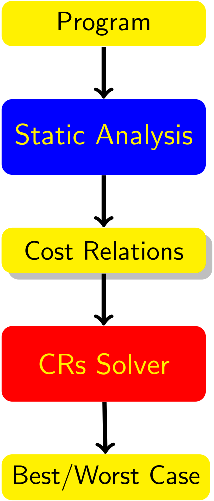
COST ANALYSIS - WEGBREIT 1975




```
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;
}
```

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

COST ANALYSIS - WEGBREIT 1975



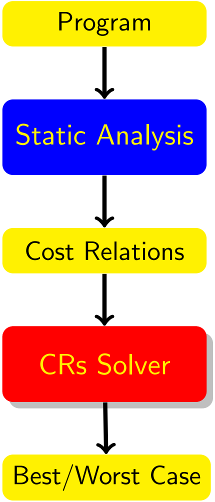

```
while ( i<n ) {
  int j=0;
  while ( j<i ) {
    for(int k=0;k<n+j;k++)
      l=new List(i*k*j,l);
    j+= (*) ? 1:3;
  }
  i+= (*) ? 2:4;
}
```

$$
\begin{array}{lcll}
A(i,n) & = & 0 & \{i \geq n\} \\
A(i,n) & = & B(0,i,n)+A(i',n) & \{i+1 \leq n,\ i+2 \leq i' \leq i+4\}
\end{array}
$$

$$
\begin{array}{lcll}
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(j,i,n) & = & C(0,j,n)+B(j',i,n) & \{j+1 \leq i,\ j+1 \leq j' \leq j+3\}
\end{array}
$$

$$
\begin{array}{lcll}
C(k,j,n) & = & 0 & \{k \geq n+j\} \\
C(k,j,n) & = & 1+C(k',j,n) & \{k' = k+1,\ k+1 \leq n+j\}
\end{array}
$$

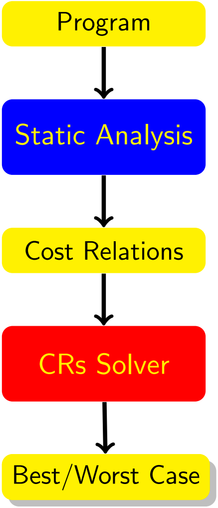
Cost analysis - Wegbreit 1975




```
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;
}
```

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

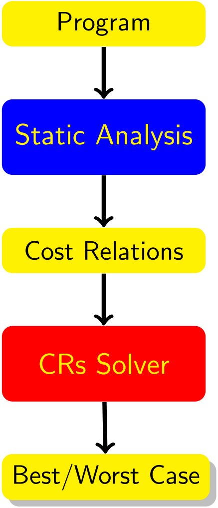
COST ANALYSIS - WEGBREIT 1975



Worst-Case (UB)

$$\begin{aligned} C(k_0, j_0, n_0) &= n_0 + j_0 - k_0 \\ B(j_0, i_0, n_0) &= \ldots\ldots \\ A(i_0, n_0) &= \ldots\ldots \end{aligned}$$

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



Worst-Case (UB)

$$C(k_0, j_0, n_0) = n_0 + j_0 - k_0$$
$$B(j_0, i_0, n_0) = \dots\dots$$
$$A(i_0, n_0) = \dots\dots$$

Best-Case (LB)

$$C(k_0, j_0, n_0) = n_0 + j_0 - k_0$$
$$B(j_0, i_0, n_0) = \dots\dots$$
$$A(i_0, n_0) = \dots\dots$$

$A(i, \dots$ $\qquad \leq i+4\}$
$A(i, n \dots$

$B(j, i, \dots$
$B(j, i, \dots$ $\qquad \leq j+3\}$

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

Solving CRs - Computer Algebra Systems

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

Solving CRs - Computer Algebra Systems

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

- Why not using directly Computer Algebra Systems?

Solving CRs - Computer Algebra Systems

$$
\begin{array}{lll}
A(i,n) & = & 0 \qquad \{i \geq n\} \\
A(i,n) & = & B(0,i,n)+A(i',n) \qquad \{i+1 \leq n, i+2 \leq i' \leq i+4\} \\
B(j,i,n) & = & 0 \qquad \{j \geq i\} \\
B(j,i,n) & = & C(0,j,n)+B(j',i,n) \qquad \{j+1 \leq i, j+1 \leq j' \leq j+3\} \\
C(k,j,n) & = & 0 \qquad \{k \geq n+j\} \\
C(k,j,n) & = & 1+C(k',j,n) \qquad \{k' = k+1, k+1 \leq n+j\}
\end{array}
$$

CAS can obtain an exact closed-form solution for:

$$
\begin{array}{l}
P(0) = 0 \\
P(x) = E + P(x-1) + \cdots + P(x-1)
\end{array}
$$

deterministic, 1 base-case, 1 recursive case, 1 argument

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

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

Solving CRs - Computer Algebra Systems

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

- Why not using directly Computer Algebra Systems?
- C

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

1: $B(1,5,3) \rightarrow B(2,5,3) \rightarrow B(5,5,3)$

2: $B(1,5,3) \rightarrow B(2,5,3) \rightarrow B(4,5,3) \rightarrow B(5,5,3)$

$$
\begin{array}{lcll}
A(i,n) & = & 0 & \{i \geq n\} \\
A(i,n) & = & B(0,i,n) + A(i',n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\} \\
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(\underline{j,i,n}) & = & C(0,j,n) + B(\underline{j',i,n}) & \{j+1 \leq i, j+1 \leq j' \leq j+3\} \\
C(k,j,n) & = & 0 & \{k \geq n+j\} \\
C(k,j,n) & = & 1 + C(k',j,n) & \{k' = k+1, k+1 \leq n+j\}
\end{array}
$$

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

Solving CRs - Computer Algebra Systems

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

- 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

$$
\begin{array}{lcll}
A(i,n) & = & 0 & \{i \geq n\} \\
A(i,n) & = & B(0,i,n) + A(i',n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\}
\end{array}
$$

$$
\left\{
\begin{array}{lcll}
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(j,i,n) & = & C(0,j,n) + B(j',i,n) & \{j+1 \leq i, j+1 \leq j' \leq j+2\} \\
B(j,i,n) & = & C(0,j,n) + B(j',i,n) & \{j+1 \leq i, j' = j+3\}
\end{array}
\right.
$$

$$
\begin{array}{lcll}
C(k,j,n) & = & 0 & \{k \geq n+j\} \\
C(k,j,n) & = & 1 + C(k',j,n) & \{k' = k+1, k+1 \leq n+j\}
\end{array}
$$

- 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

$$
\begin{array}{lcll}
A(i,n) & = & 0 & \{i \geq n\} \\
A(i,n) & = & B(0,i,n)+A(i',n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\}
\end{array}
$$

$$
\left\{
\begin{array}{lcll}
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(j,i,n) & = & C(0,j,n)+B(j',i,n) & \{j+1 \leq i, j+1 \leq j' \leq j+2\} \\
B(j,i,n) & = & C(0,j,n)+B(j',i,n) & \{j+1 \leq i, j'=j+3\}
\end{array}
\right.
$$

$$
\begin{array}{lcll}
C(k,j,n) & = & 0 & \{k \geq n+j\} \\
C(k,j,n) & = & 1+C(k',j,n) & \{k' = k+1, k+1 \leq n+j\}
\end{array}
$$

- 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

$$
\begin{array}{lcll}
A(i,n) & = & 0 & \{i \geq n\} \\
A(i,n) & = & B(0,i,n) + A(i',n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\}
\end{array}
$$

$$
\left\{
\begin{array}{lcll}
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(j,i,n) & = & C(0,j,n) + B(j',i,n) & \{j+1 \leq i, j+1 \leq j' \leq j+2\} \\
B(j,i,n) & = & C(0,j,n) + B(j',i,n) & \{j+1 \leq i, j' = j+3\}
\end{array}
\right.
$$

$$
\begin{array}{lcll}
C(k,j,n) & = & 0 & \{k \geq n+j\} \\
C(k,j,n) & = & 1 + C(k',j,n) & \{k' = k+1, k+1 \leq n+j\}
\end{array}
$$

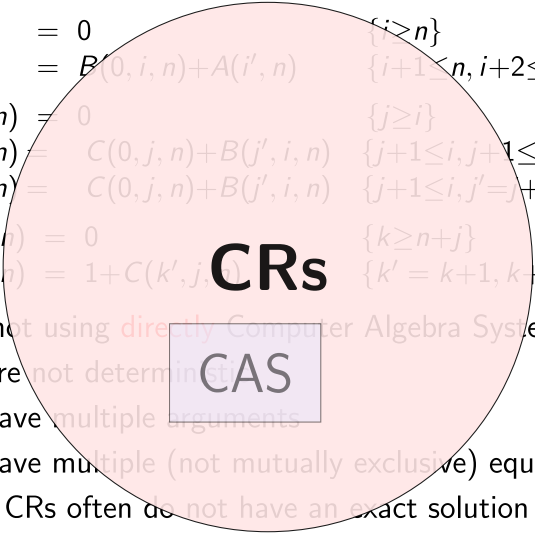
CRs

- 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

$$
\begin{array}{lcll}
A(i,n) & = & 0 & \{i \geq n\} \\
A(i,n) & = & B(0,i,n) + A(i',n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\}
\end{array}
$$

$$
\left\{
\begin{array}{lcll}
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(j,i,n) & = & C(0,j,n) + B(j',i,n) & \{j+1 \leq i, j+1 \leq j' \leq j+2\} \\
B(j,i,n) & = & C(0,j,n) + B(j',i,n) & \{j+1 \leq i, j' = j+3\}
\end{array}
\right.
$$

$$
\begin{array}{lcll}
C(k,j,n) & = & 0 & \{k \geq n+j\} \\
C(k,j,n) & = & 1 + C(k',j,n) & \{k' = k+1, k+1 \leq n+j\}
\end{array}
$$



- 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

Inferring Upper Bounds (CRs) - PUBS

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

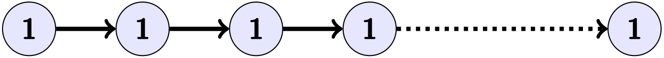
Inferring Upper Bounds (CRs) - PUBS

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

Inferring Upper Bounds (CRs) - PUBS

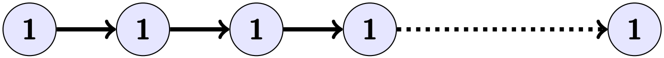
$$
\begin{array}{llll}
A(i,n) & = & 0 & \{i \geq n\} \\
A(i,n) & = & B(0,i,n) + A(i',n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\} \\
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(j,i,n) & = & C(0,j,n) + B(j',i,n) & \{j+1 \leq i, j+1 \leq j' \leq j+3\} \\
C(k,j,n) & = & 0 & \{k \geq n+j\} \\
C(k,j,n) & = & 1 + C(k',j,n)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

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



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

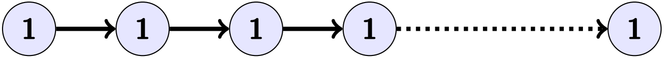
- An evaluation for $C(k_0, j_0, n_0)$ looks like:



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

$$
\begin{array}{lcll}
A(i,n) & = & 0 & \{i \geq n\} \\
A(i,n) & = & B(0,i,n) + A(i',n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\} \\
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(j,i,n) & = & C(0,j,n) + B(j',i,n) & \{j+1 \leq i, j+1 \leq j' \leq j+3\} \\
C(k,j,n) & = & 0 & \{k \geq n+j\} \\
C(k,j,n) & = & 1 + C(k',j,n) & \underbrace{\{k' = k+1, k+1 \leq n+j\}}_{\varphi}
\end{array}
$$

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



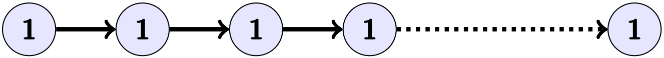
- What is the maximum length of this chain of **1** ?

Ranking function

$$\exists \hat{f}.\ \varphi \models \hat{f}(k,j,n) \geq 0 \ \wedge \ \hat{f}(k,j,n) - \hat{f}(k',j',n') \geq 1$$

$$
\begin{array}{lcll}
C(k,j,n) & = & 0 & \{k \geq n+j\} \\
C(k,j,n) & = & 1 + C(k',j,n) & \underbrace{\{k' = k+1, k+1 \leq n+j\}}_{\varphi}
\end{array}
$$

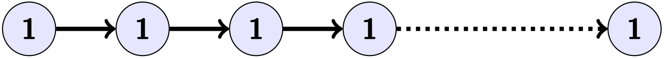
- An evaluation for $C(k_0, j_0, n_0)$ looks like:



- What is the maximum length of this chain of 1 ?

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

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



- What is the maximum length of this chain of **1** ?

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

$A(i,n)$ … $\{i \geq n\}$

$A(i,n)$ … $\{i+1 \leq n, i+2 \leq i' \leq i+4\}$

$B(j,i,n)$ … $\{j \geq i\}$

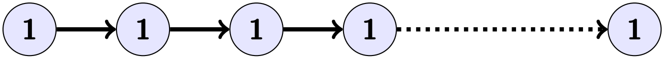
$B(j,i,n)$ … $\{j+1 \leq i, j+1 \leq j' \leq j+3\}$

$\|a\| = max(a,0)$

$$C(k,j,n) = 0 \qquad \{k \geq n+j\}$$

$$C(k,j,n) = 1 + C(k',j,n) \qquad \{k' = k+1, k+1 \leq n+j\}$$

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



- What is the maximum length of this chain of **1** ?

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

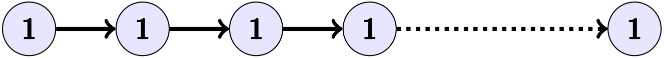
$A(i,$ … $\{i \ge n\}$
$A(i,$ … $\le i+4\}$
$B(j,$ …
$B(j,$ … $\le j+3\}$

Worst-Case

$$C^{ub}(k_0, j_0, n_0) = \textcircled{\textbf{1}} * \|n_0 + j_0 - k_0\|$$

$$\begin{array}{lcll} C(k,j,n) & = & 0 & \{k \ge n+j\} \\ C(k,j,n) & = & 1 + C(k',j,n) & \{k' = k+1, k+1 \le n+j\} \end{array}$$

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



- What is the maximum length of this chain of $\textcircled{\textbf{1}}$?

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

Inferring Upper Bounds (CRs) - PUBS

$A(i, \ldots$ $\quad \{\ldots\}$
$A(i, \ldots$ $\quad \ldots \leq i+4\}$

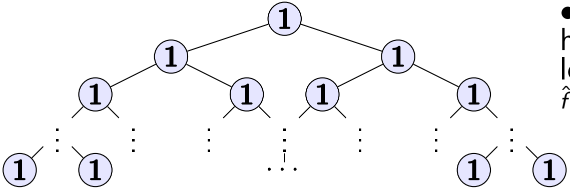
$B(j, \ldots$
$B(j, \ldots$ $\quad \ldots \leq j+3\}$

Worst-Case

$$C^{ub}(k_0, j_0, n_0) = \textcircled{1} * 2^{\|n_0+j_0-k_0\|}$$

$$
\begin{array}{lcll}
C(k,j,n) & = & 0 & \{k \geq n+j\} \\
C(k,j,n) & = & 1 + C(k',j,n) + C(k',j,n) & \{k' = k+1, k+1 \leq n+j\}
\end{array}
$$

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



- How many ① has from root to leaf?

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

Inferring Upper Bounds (CRs) - PUBS

$$
\begin{array}{lcll}
A(i,n) & = & 0 & \{i \geq n\} \\
A(i,n) & = & B(0,i,n) + A(i',n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\} \\
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(j,i,n) & = & C(0,j,n) + B(j',i,n) & \{j+1 \leq i, j+1 \leq j' \leq j+3\}
\end{array}
$$

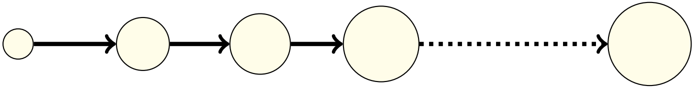
Inferring Upper Bounds (CRs) - PUBS

$$
\begin{array}{lcll}
A(i,n) & = & 0 & \{i \geq n\} \\
A(i,n) & = & B(0,i,n) + A(i',n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\} \\
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(j,i,n) & = & \|n+j\| + B(j',i,n) & \{j+1 \leq i, j+1 \leq j' \leq j+3\}
\end{array}
$$

Inferring Upper Bounds (CRs) - PUBS

$$
\begin{array}{llll}
A(i,n) & = & 0 & \{i \geq n\} \\
A(i,n) & = & B(0,i,n) + A(i',n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\} \\
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(j,i,n) & = & \|n+j\| \quad + B(j',i,n) & \{j+1 \leq i, j+1 \leq j' \leq j+3\}
\end{array}
$$

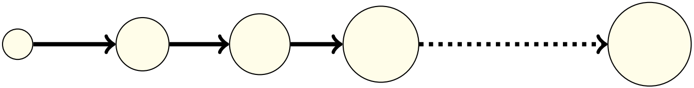
- An evaluation for $B(j_0, i_0, n_0)$ looks like:



Inferring Upper Bounds (CRs) - PUBS

$$
\begin{array}{lcll}
A(i,n) & = & 0 & \{i \geq n\} \\
A(i,n) & = & B(0,i,n) + A(i',n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\} \\
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(j,i,n) & = & \|n+j\| + B(j',i,n) & \{j+1 \leq i, j+1 \leq j' \leq j+3\}
\end{array}
$$

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

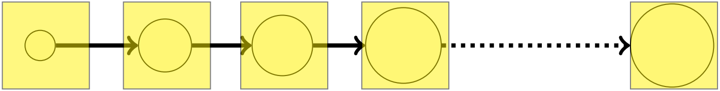


- There are at most $\|i_0 - j_0\|$ circles (*ranking function*)

Inferring Upper Bounds (CRs) - PUBS

$$
\begin{array}{lcll}
A(i,n) & = & 0 & \{i \geq n\} \\
A(i,n) & = & B(0,i,n) + A(i',n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\} \\
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(j,i,n) & = & \|n+j\| \quad + B(j',i,n) & \{j+1 \leq i, j+1 \leq j' \leq j+3\}
\end{array}
$$

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

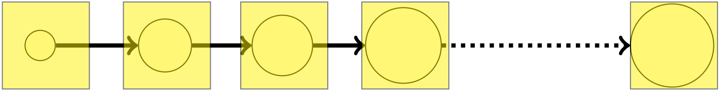


- There are at most $\|i_0 - j_0\|$ circles (*ranking function*)

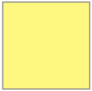
Inferring Upper Bounds (CRs) - PUBS

$$
\begin{array}{llll}
A(i,n) & = & 0 & \{i \geq n\} \\
A(i,n) & = & B(0,i,n) + A(i',n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\} \\
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(j,i,n) & = & \|n+j\| \quad + B(j',i,n) & \{j+1 \leq i, j+1 \leq j' \leq j+3\}
\end{array}
$$

- 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  $* \|i_0 - j_0\|$

Inferring Upper Bounds (CRs) - PUBS

$$
\begin{array}{llll}
A(i,n) & = & 0 & \{i \geq n\} \\
A(i,n) & = & B(0,i,n) + A(i',n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\} \\
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(j,i,n) & = & \underline{\|n+j\|} + B(j',i,n) & \underbrace{\{j+1 \leq i, j+1 \leq j' \leq j+3\}}_{\varphi}
\end{array}
$$

- Ar

Inferring box (or maximizing expression)

- 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

$$
\begin{array}{llll}
A(i,n) & = & 0 & \{i \geq n\} \\
A(i,n) & = & B(0,i,n) + A(i',n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\} \\
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(j,i,n) & = & \|n+j\| \quad + B(j',i,n) & \underbrace{\{j+1 \leq i, j+1 \leq j' \leq j+3\}}_{\varphi}
\end{array}
$$

• A

•

•

Inferring box (or maximizing expression)

- 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 $\langle B(j_0, i_0, n_0) \rightsquigarrow B(j,i,n), \Psi \rangle$

Inferring Upper Bounds (CRs) - PUBS

$$
\begin{array}{lcll}
A(i,n) & = & 0 & \{i \geq n\} \\
A(i,n) & = & B(0,i,n) + A(i',n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\} \\
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(j,i,n) & = & \underline{\|n+j\|} \quad + B(j',i,n) & \underbrace{\{j+1 \leq i, j+1 \leq j' \leq j+3\}}_{\varphi}
\end{array}
$$

- Ar

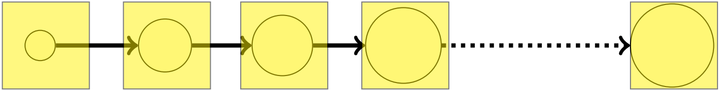
Inferring box (or maximizing expression)

- 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 $\langle B(j_0, i_0, n_0) \rightsquigarrow B(j,i,n), \Psi \rangle$
- Use (parametric) integer programming to maximize $n+j$ w.r.t $\Psi \wedge \varphi$ and the parameters $\langle j_0, i_0, n_0 \rangle$.

Inferring Upper Bounds (CRs) - PUBS

$$
\begin{array}{lcll}
A(i,n) & = & 0 & \{i \geq n\} \\
A(i,n) & = & B(0,i,n) + A(i',n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\} \\
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(j,i,n) & = & \|n+j\| + B(j',i,n) & \{j+1 \leq i, j+1 \leq j' \leq j+3\}
\end{array}
$$

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

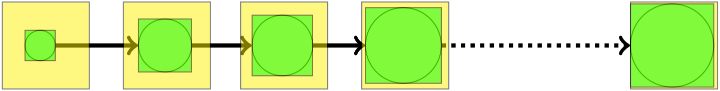


- There are at most $\|i_0 - j_0\|$ circles (*ranking function*)
- $B^{ub}(j_0, i_0, n_0) = \|n_0 + i_0 - 1\| * \|i_0 - j_0\|$

Inferring Upper Bounds (CRs) - PUBS

$$
\begin{array}{llll}
A(i,n) & = & 0 & \{i \geq n\} \\
A(i,n) & = & B(0,i,n) + A(i',n) & \{i+1 \leq n, i+2 \leq i' \leq i+4\} \\
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(j,i,n) & = & \|n+j\| \quad + B(j',i,n) & \{j+1 \leq i, j+1 \leq j' \leq j+3\}
\end{array}
$$

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



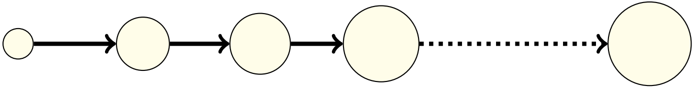
- There are at most $\|i_0 - j_0\|$ circles (*ranking function*)

Outline

Inferring Upper Bounds - Static analysis + CAS

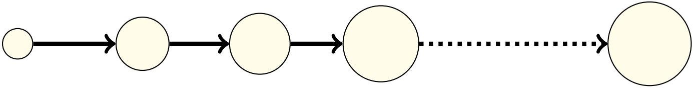
$$B(j,i,n) = 0 \qquad \{j \geq i\}$$
$$B(j,i,n) = \|n+j\| + B(j',i,n) \qquad \{j+1 \leq i, j+1 \leq j' \leq j+3\}$$

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



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

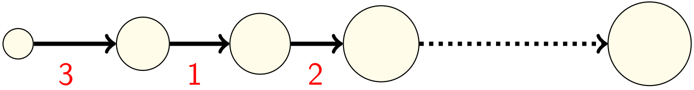
- An evaluation for $B(j_0, i_0, n_0)$ looks like:



- How these circles progress?

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

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

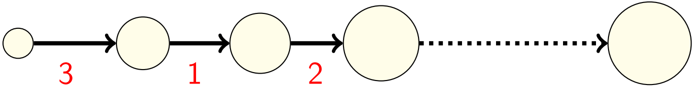


- How these circles progress? $\check{d} = 1$ and $\hat{d} = 3$

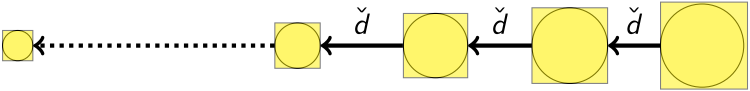
Inferring Upper Bounds - Static analysis + CAS

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

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

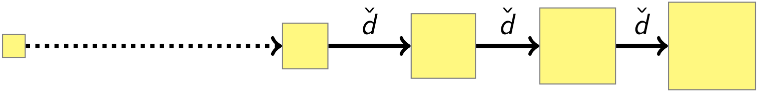


- How these circles progress? $\check{d} = 1$ and $\hat{d} = 3$

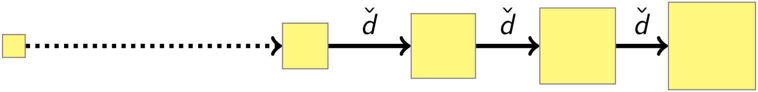


Inferring Upper Bounds - Static analysis + CAS

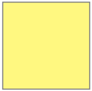
$$B(j,i,n) = 0 \qquad \{j \geq i\}$$
$$B(j,i,n) = \|n+j\| + B(j',i,n) \qquad \{j+1 \leq i, j+1 \leq j' \leq j+3\}$$



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

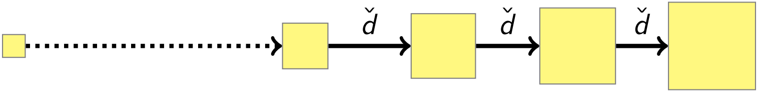


$$P(0) = 0$$
$P(x) =$  $+ (\|i_0 - j_0\| - x) * \check{d} + P(x-1)$

 $= \quad \|I\|$

 $= \quad \|I - (i_0 - j_0 - 1) * \check{d}\|$

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

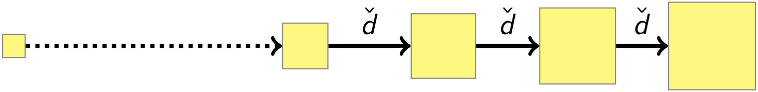


$$P(0) = 0$$
$$P(x) = \square + (\|i_0 - j_0\| - x) * \check{d} \; + \; P(x-1)$$

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

Inferring Upper Bounds - Static analysis + CAS

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



$$
\begin{aligned}
P(0) &= 0 \\
P(x) &= \square + (\|i_0 - j_0\| - x) * \check{d} \; + \; P(x-1)
\end{aligned}
$$

- $P(\|i_0 - j_0\|)$ 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

$$P(x) = \|n_0+j_0-1\|*x+\|i_0-j_0\|*x+\frac{x}{2}-\frac{x^2}{2}$$

$$P(\|i_0-j_0\|) = \|n_0+j_0-1\|*\|i_0-j_0\|+\|i_0-j_0\|^2+\frac{\|i_0-j_0\|}{2}-\frac{\|i_0-j_0\|^2}{2}$$

$$P(0) = 0$$
$$P(x) = \square + (\|i_0-j_0\|-x)*\check{d} + P(x-1)$$

- $P(\|i_0-j_0\|)$ is the sum of all boxes
- If E is a closed-form solution for $P(x)$ obtained by CAS, then $E[x/\|i_0-j_0\|]$ is an UB on the worst-case of B

CRs with Complex Nat Expression

$$
\begin{aligned}
A(i,n) &= 0 & \varphi_0 \\
A(i,n) &= B(0,i,n) + A(i',n) & \varphi_1
\end{aligned}
$$

CRs with Complex Nat Expression

$$
\begin{aligned}
A(i,n) &= 0 && \varphi_0 \\
A(i,n) &= \|n_0 - 1\| * \|i_0\| + \frac{\|i_0\|}{2} * (\|i_0\| + 1) + A(i', n) && \varphi_1
\end{aligned}
$$

CRs with Complex Nat Expression

$$A(i,n) = 0 \qquad \varphi_0$$

$$A(i,n) = \|n_0 - 1\| * \|i_0\| + \frac{\|i_0\|}{2} * (\|i_0\| + 1) + A(i', n) \qquad \varphi_1$$

Inferring Upper Bounds - Static analysis + CAS

CRs with Complex Nat Expression

$$A(i,n) = 0 \qquad \varphi_0$$

$$A(i,n) = \|n_0 - 1\| * \|i_0\| + \frac{\|i_0\|}{2} * (\|i_0\| + 1) + A(i', n) \qquad \varphi_1$$

$$P(0) = 0$$

$$P(x) = E_1 * E_2 + \frac{E_3}{2} * (E_4 + 1) \quad + P(x-1)$$

$$E_1 = \square + (\|n_0 - i_0\| - x) * \check{d}_1$$

$$E_2 = \square + (\|n_0 - i_0\| - x) * \check{d}_2$$

$$E_3 = \square + (\|n_0 - i_0\| - x) * \check{d}_3$$

$$E_4 = \square + (\|n_0 - i_0\| - x) * \check{d}_4$$

Geometrically Progressive nat

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

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

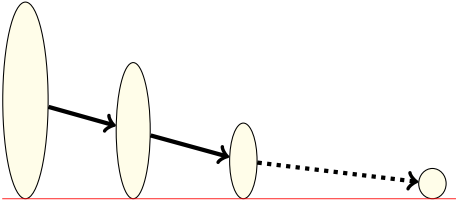
CRs for Mergesort

Geometrically Progressive nat

$$Ms(l,h) = 0, \qquad \{\ h \le l, h \ge 0, l \ge 0\ \}$$

$$Ms(l,h) = \|h-l+1\| + Ms(l,m) + Ms(m+1,h), \ \{\ h \ge l+1, l+h-1 \le 2*m \le l+h\ \}$$

CRs for Mergesort

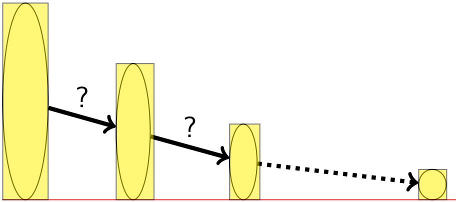


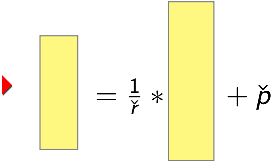
Geometrically Progressive nat

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

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

CRs for Mergesort





CRs with Multiple Recursive Equations

$$
\begin{array}{lll}
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(j,i,n) & = & \|n+j\| + B(j',i,n) & \{j+1 \leq i, j+1 \leq j' \leq j+3\} \\
B(j,i,n) & = & \|j\| * \|j\| + B(j',i,n) & \{j+1 \leq i, j' = j+1\}
\end{array}
$$

CRs with Multiple Recursive Equations - First Alternative

$$
\begin{array}{lll}
B(j,i,n) & = & 0 \qquad \{j \geq i\} \\
B(j,i,n) & = & \|n+j\| + B(j',i,n) \qquad \{j+1 \leq i, j+1 \leq j' \leq j+3\} \\
B(j,i,n) & = & \|j\| * \|j\| + B(j',i,n) \qquad \{j+1 \leq i, j' = j+1\}
\end{array}
$$

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

$$B_T(j,i,n) = \|n+j\| * \|j+1\| + B_T(j',i,n) \qquad \{j+1 \leq i, j+1 \leq j' \leq j+3\}$$

CRs with Multiple Recursive Equations - First Alternative

$$B(j,i,n) = 0 \qquad \{j \geq i\}$$

$$B(j,i,n) = \underbrace{\|n+j\| +}_{e_1} B(j',i,n) \qquad \{j+1 \leq i, j+1 \leq j' \leq j+3\}$$

$$B(j,i,n) = \underbrace{\|j\| * \|j\| +}_{e_2} B(j',i,n) \qquad \{j+1 \leq i, j' = j+1\}$$

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

$$B_T(j,i,n) = \underbrace{\|n+j\| * \|j+1\|}_{e_3} + B_T(j',i,n) \qquad \{j+1 \leq i, j+1 \leq j' \leq j+3\}$$

CRs with Multiple Recursive Equations - First Alternative

$$B(j,i,n) = 0 \qquad \{j \geq i\}$$

$$B(j,i,n) = \underbrace{\|n+j\|}_{e_1} + B(j',i,n) \qquad \underbrace{\{j+1 \leq i, j+1 \leq j' \leq j+3\}}_{\varphi_1}$$

$$B(j,i,n) = \underbrace{\|j\| * \|j\|}_{e_2} + B(j',i,n) \qquad \underbrace{\{j+1 \leq i, j' = j+1\}}_{\varphi_2}$$

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

$$B_T(j,i,n) = \underbrace{\|n+j\| * \|j+1\|}_{e_3} + B_T(j',i,n) \qquad \underbrace{\{j+1 \leq i, j+1 \leq j' \leq j+3\}}_{\text{conv.hull } \{\varphi_1, \varphi_2\}}$$

CRs with Multiple Recursive Equations - First Alternative

$$
\begin{aligned}
B(j,i,n) &= 0 && \{j \geq i\} \\
B(j,i,n) &= \|n+j\| + B(j',i,n) && \{j+1 \leq i, j+1 \leq j' \leq j+3\} \\
B(j,i,n) &= \|j\| * \|j\| + B(j',i,n) && \{j+1 \leq i, j' = j+1\}
\end{aligned}
$$

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

$$
B_T(j,i,n) = \underline{\|n+j\|} * \underline{\|j+1\|} + B_T(j',i,n) \quad \{j+1 \leq i, j+1 \leq j' \leq j+3\}
$$

$$
\begin{aligned}
P(x) &= 0 \\
P(x) &= \underline{E_1} * \underline{E_2} + P(x-1)
\end{aligned}
$$

$$
\begin{aligned}
E_1 &= \square + (\|i_0 - j_0\| - x) * \check{d}_1 \\
E_2 &= \square + (\|i_0 - j_0\| - x) * \check{d}_2
\end{aligned}
$$

CRs with Multiple Recursive Equations - Second Alternative

$$
\begin{array}{lll}
B(j,i,n) & = & 0 & \{j \geq i\} \\
B(j,i,n) & = & \|n+j\| + B(j',i,n) & \{j+1 \leq i, j+1 \leq j' \leq j+3\} \\
B(j,i,n) & = & \|j\| * \|j\| + B(j',i,n) & \{j+1 \leq i, j' = j+1\}
\end{array}
$$

$$\Downarrow$$

$$
\begin{array}{lll}
B_1(j,i,n) & = & 0 & \{j \geq i\} \\
B_1(j,i,n) & = & \|n+j\| + B_1(j',i,n) & \{j+1 \leq i, j+1 \leq j' \leq j+3\} \\
B_1(j,i,n) & = & 0 + B_1(j',i,n) & \{j+1 \leq i, j' = j+1\}
\end{array}
$$

$$
\begin{array}{lll}
B_2(j,i,n) & = & 0 & \{j \geq i\} \\
B_2(j,i,n) & = & 0 + B_2(j',i,n) & \{j+1 \leq i, j+1 \leq j' \leq j+3\} \\
B_2(j,i,n) & = & \|j\| * \|j\| + B_2(j',i,n) & \{j+1 \leq i, j' = j+1\}
\end{array}
$$

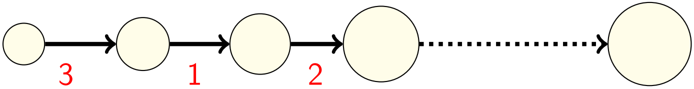
$$B^{ub}(j_0,i_0,n_0) = B_1^{ub}(j_0,i_0,n_0) + B_2^{ub}(j_0,i_0,n_0)$$

Inferring Lower Bounds

$$
\begin{array}{ll}
B(j,i,n) = 0 & \{j \geq i\} \\
B(j,i,n) = \|n+j\| + B(j',i,n) & \{j+1 \leq i, j+1 \leq j' \leq j+3\}
\end{array}
$$

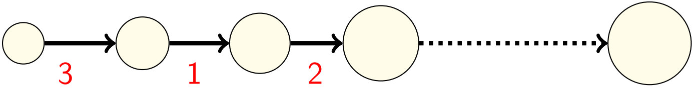
$$B(j,i,n) = 0 \qquad \{j \geq i\}$$
$$B(j,i,n) = \|n+j\| + B(j',i,n) \qquad \{j+1 \leq i, j+1 \leq j' \leq j+3\}$$

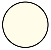
- An evaluation for $B(j_0, i_0, n_0)$ looks like:



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

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

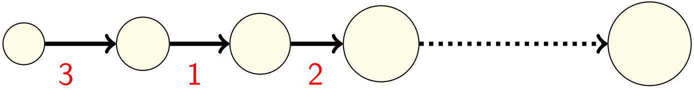


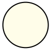
- What is the minimum number of  ?

Inferring Lower Bounds - Static analysis + CAS

$$B(j,i,n) = 0 \qquad \{j \geq i\}$$
$$B(j,i,n) = \|n+j\| + B(j',i,n) \qquad \{j+1 \leq i, \underline{j+1 \leq j' \leq j+3}\}$$

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

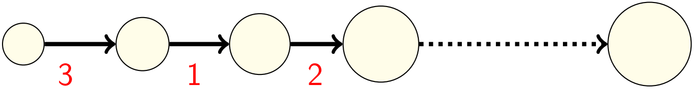


- What is the minimum number of  ? $\left\| \frac{i_0 - j_0}{3} \right\|$

λ $\lambda+1$

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

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

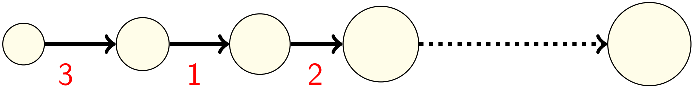


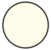
- What is the minimum number of ◯ ? $\left\| \frac{i_0 - j_0}{3} \right\|$
- Add loop counter λ to the CR

$\lambda \qquad \lambda+1$

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

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

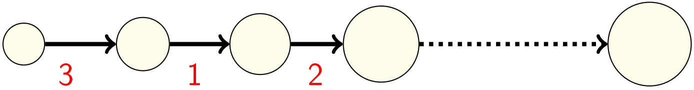


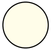
- What is the minimum number of  ? $\left\|\frac{i_0 - j_0}{3}\right\|$
- Add loop counter λ to the CR
- Infer an invariant $\langle B(j_0, i_0, n_0, 0) \rightsquigarrow B(j, i, n, \lambda), \Psi\rangle$

λ $\lambda+1$

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

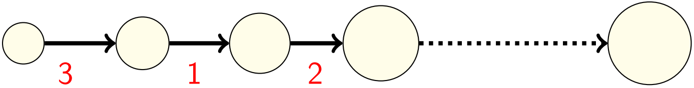
- An evaluation for $B(j_0, i_0, n_0)$ looks like:

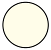
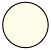


- What is the minimum number of  ? $\left\|\frac{i_0 - j_0}{3}\right\|$
- Add loop counter λ to the CR
- Infer an invariant $\langle B(j_0, i_0, n_0, 0) \rightsquigarrow B(j, i, n, \lambda), \Psi \rangle$
- Minimize λ w.r.t $\Psi \wedge \{j \geq i\}$ and the parameters $\langle j_0, i_0, n_0 \rangle$

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

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

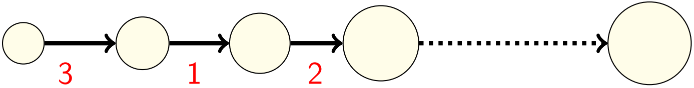


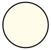
- What is the minimum number of  ? $\|\frac{i_0 - j_0}{3}\|$
- What is the LB of ? i.e. on $\|n+j\|$

Inferring Lower Bounds - Static analysis + CAS

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

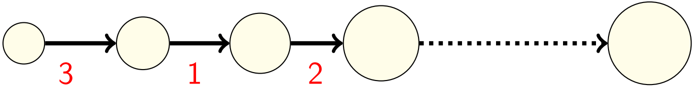
- An evaluation for $B(j_0, i_0, n_0)$ looks like:

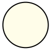


- What is the minimum number of  ? $\| \frac{i_0 - j_0}{3} \|$
- What is the LB of ? i.e. on $\|n+j\|$

$$B(j,i,n) = 0 \qquad \{j \geq i\}$$
$$B(j,i,n) = \|n+j\| + B(j',i,n) \qquad \{j+1 \leq i, \underline{j+1 \leq j' \leq j+3}\}$$

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

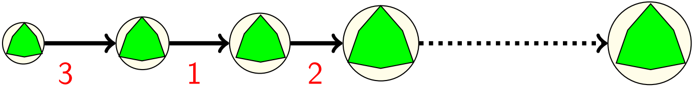


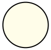
- What is the minimum number of  ? $\left\| \frac{i_0 - j_0}{3} \right\|$
- What is the LB of ? i.e. on $\|n+j\|$

 $* \left\| \frac{i_0 - j_0}{3} \right\|$

$$B(j,i,n) = 0 \qquad \{j \geq i\}$$
$$B(j,i,n) = \|n+j\| + B(j',i,n) \qquad \{j+1 \leq i, \underline{j+1 \leq j' \leq j+3}\}$$

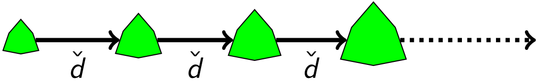
- An evaluation for $B(j_0, i_0, n_0)$ looks like:



- What is the minimum number of  ? $\|\frac{i_0 - j_0}{3}\|$
- What is the LB of ? i.e. on $\|n+j\|$

Inferring Lower Bounds - Static analysis + CAS

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

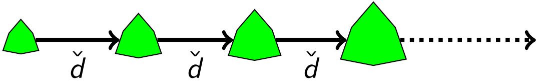


$$P(0) = 0$$

$P(x) =$  $+ (\|\frac{i_0 - j_0}{3}\| - x) * \check{d} \; + \; P(x-1)$

Inferring Lower Bounds - Static analysis + CAS

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

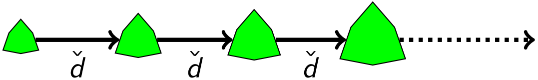


$$P(0) = 0$$
$$P(x) = $$  $$+ (\|\tfrac{i_0 - j_0}{3}\| - x) * \check{d} + P(x-1)$$

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

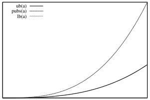
Inferring Lower Bounds - Static analysis + CAS

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

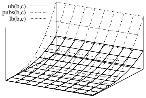


$$P(0) = 0$$
$$P(x) = $$  $$+ (\|\frac{i_0 - j_0}{3}\| - x) * \check{d} \; + \; P(x-1)$$

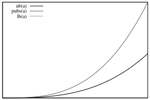
- $P(\frac{\|i_0 - j_0\|}{3})$ is the sum of all 
- If E is a closed-form solution for $P(x)$ obtained by CAS, then $E[x/\|\frac{i_0 - j_0}{3}\|]$ is an LB on the best-case of B



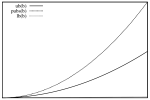
DetEval



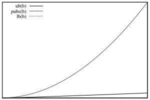
LinEqSolve



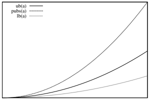
MatrixInv



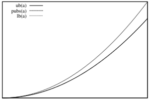
InsertSort



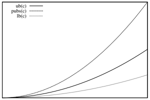
MergeSort



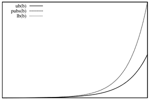
SelectSort



PascalTriangle

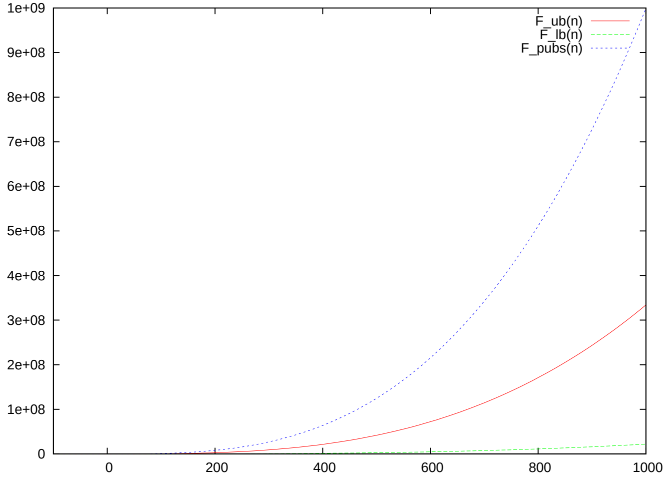


BubbleSort



NestedRecIter

Comparing Results with PUBS



Comparison - PUBS vs Static Analysis + CAS

	PUBS	PUBS+CAS
Precision	★★★☆☆	★★★★☆
Applicability	★★★★★	★★★★☆
Non-Determinism	★★★★★	★★★☆☆
Performance	★★★★★	★★★★★
Lower Bounds	★☆☆☆☆	★★★★☆

Comparison with RAML (Hoffmann et al. TOPLAS 2012)

`apAll`	**(A)** $\|a\|\cdot\|b\| + 2\cdot\|a\| + 1$ **(B)** $a\cdot b + 2\cdot a + 1$	64 46
`isort`	**(A)** $\frac{1}{2}\cdot\|a\|^2 + \frac{5}{2}\cdot\|a\| + 1$ **(B)** $\frac{1}{2}\cdot a^2 + \frac{3}{2}\cdot a + 1$	33 74
`dyade`	**(A)** $\|a\|\cdot\|b\| + 2\cdot\|a\| + 1$ **(B)** $2\cdot a\cdot b + 2\cdot a + 1$	40 44
`mult3`	**(A)** $2\cdot\|a\|^2 + 8\cdot\|a\| + 3$ **(B)** $4\cdot a\cdot b + 6\cdot a + 3$	70 193
`msort`	**(A)** $log_2(\|2\cdot a - 3\|+1)\cdot\|a-\frac{1}{2}\|+4\cdot\|2\cdot a-3\|+ log_2(\|2\cdot a-3\|+1)\cdot\|\frac{a}{2}\|+1$ **(B)** $\frac{7}{2}\cdot a^2 - \frac{5}{2}\cdot a + 1$	76 73
`qsort`	**(A)** $8 \cdot 2^{\|a\|} - 2 \cdot \|a\| - 7$ **(B)** $a^2 + 3 \cdot a + 1$	83 76

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.

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

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 $\frac{(n+1)*n}{2}$ 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 $O(n * log(n))$ for merge-sort
  - it allows inferring lower bounds on the best-case
  - `http://costa.ls.fi.upm.es/pubs`

Summary of Contributions - I

Our contributions are published in the following papers:

- Elvira Albert, Samir Genaim, and Abu Naser Masud. *More precise yet widely applicable cost analysis.* In *VMCAI 2011, USA, January, 2011. Proceedings*, volume 6538 of *LNCS*, pages 38–53.

- Elvira Albert, Samir Genaim, and Abu Naser Masud. *On the inference of resource usage upper and lower bounds. ACM Transactions on Computational Logic*. **To Appear**.

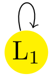
Outline

Integer linear constraints loops - motivation

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

Integer linear constraints loops - motivation

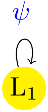
$$
\begin{array}{lll}
C(k,j,n) & = & 0 \qquad\qquad\qquad\quad \{k \geq n+j\} \\
C(k,j,n) & = & 1 + C(k',j,n) \; \{k' = k+1, k+1 \leq n+j\}
\end{array}
$$



Integer linear constraints loops - motivation

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

ψ



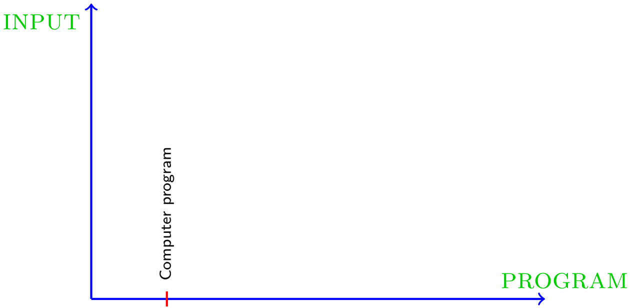
Integer Linear-Constraint (ILC) loops

The Termination Problem

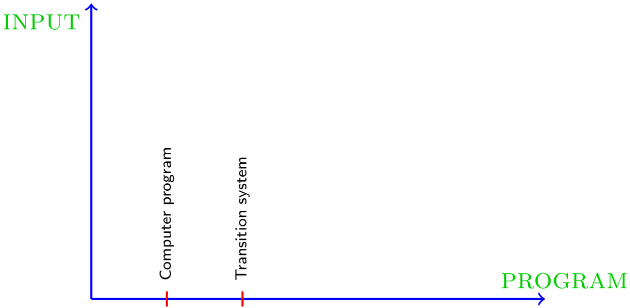
Given a program P, decide whether it will finish running or could run forever

The Termination Problem

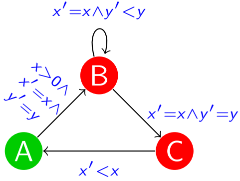
The Termination Problem

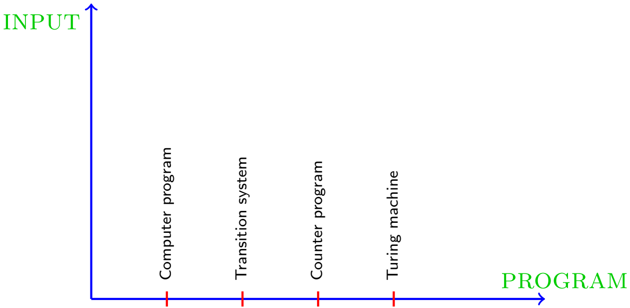



```
while (i<j) {
   ...
   i = i+1;
}
```


INPUT
PROGRAM
Computer program
Transition system

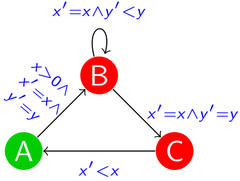
```
while (i<j) {
    ...
    i = i+1;
}
```


x′=x∧y′<y
x>0∧ x′=x∧ y′=y
x′=x∧y′=y
x′<x
A
B
C

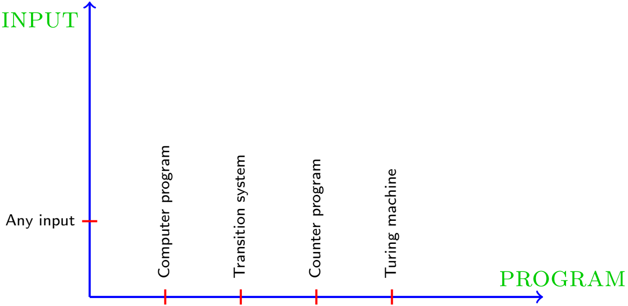


INPUT
Computer program
Transition system
Counter program
Turing machine
PROGRAM

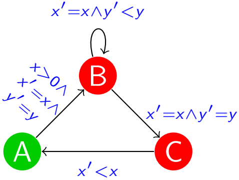

```
while (i<j) {
    ...
    i = i+1;
}
```



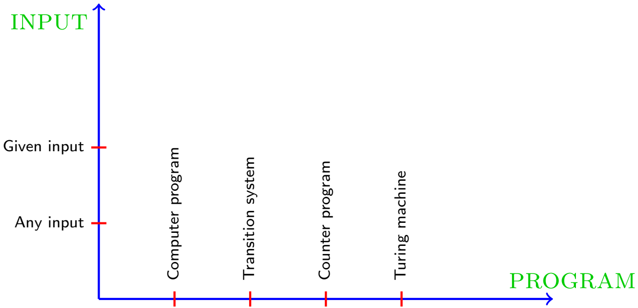
x'=x∧y'<y
x>0∧
x'=x∧
y'=y
x'=x∧y'=y
x'<x
A
B
C


INPUT
PROGRAM
Any input
Computer program
Transition system
Counter program
Turing machine

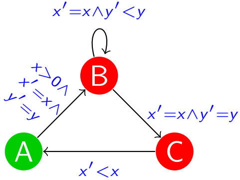
```
while (i<j) {
    ...
    i = i+1;
}
```


$x'=x \wedge y'<y$
B
$x>0 \wedge x'=x \wedge y'=y$
$x'=x \wedge y'=y$
A
C
$x'<x$

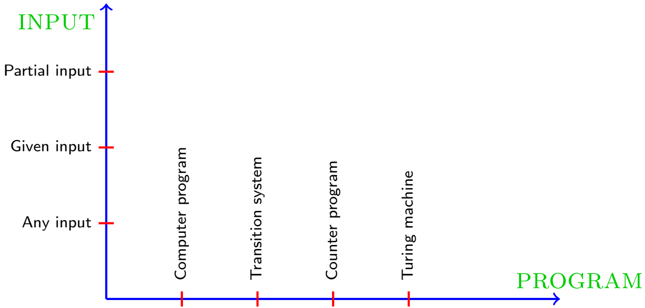
The Termination Problem



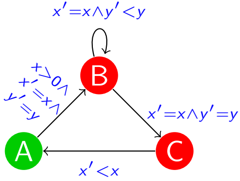

```
while (i<j) {
    ...
    i = i+1;
}
```



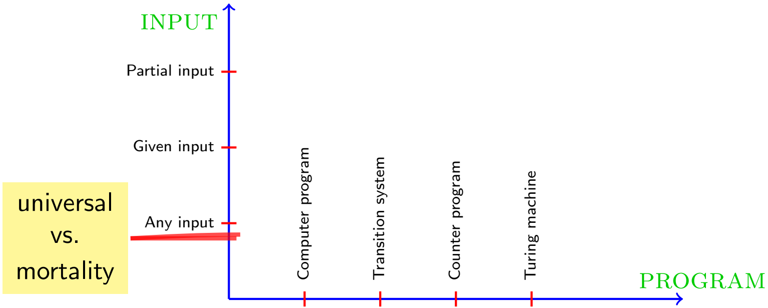
The Termination Problem



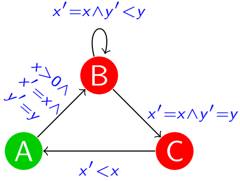

```
while (i<j) {
    ...
    i = i+1;
}
```



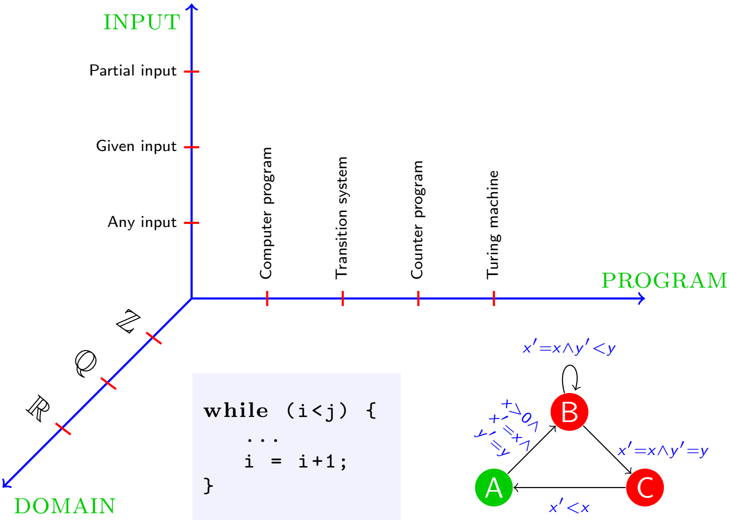
The Termination Problem




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



The Termination Problem




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

Integer Linear while loops

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

Integer Linear while loops

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

- **while** (B**x**>**b**) **x**=A**x**+**c** [TIWARI'04]
  - For *any input*, it is decidable over $\mathbb{R}$
- **while** (B**x**>**b** $\wedge$ D**x**$\geq$**d**) **x**=A**x**+**c** [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

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

- **while** (B**x**>**b**) **x**=A**x**+**c** [TIWARI'04]
  - For *any input*, it is decidable over $\mathbb{R}$
- **while** (B**x**>**b** $\wedge$ D**x**$\geq$**d**) **x**=A**x**+**c** [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 (ILW) loops

Termination of ILW Loops

Termination of Integer Linear while loops

general cases

Termination of ILW Loops

special cases

Termination of Integer Linear while loops

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

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

general cases

Termination of ILW Loops

special cases

Termination of Integer Linear while loops

general cases

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

Termination of ILW Loops

special cases

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

general cases

Termination of ILW Loops

spe

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
```

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
```

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
```

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
```

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
```

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
```

- 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:  if x>0 then 1 else 4
4:  end
```

```
while (A1>=0&& ··· && A3>=0 && A1+···+A3=1 && x>=0 && y>=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 (A1>=0&& ··· && A3>=0 && A1+···+A3=1 && x>=0 && y>=0)
{



}
```

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 (A1>=0&& ··· && A3>=0 && A1+···+A3=1 && x>=0 && y>=0)
{

   x:=x-A1;

}
```

- 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 (A1>=0&& ··· && A3>=0 && A1+···+A3=1 && x>=0 && y>=0)
{

   x:=x-A1;
   y:=y+A2;

}
```

- 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 (A1>=0&& ··· && A3>=0 && A1+···+A3=1 && x>=0 && y>=0)
{
    N0:=0; N1:=A1; N2:=A2; N3=A3;

    x:=x-A1;
    y:=y+A2;




    A1:=N0; A2:=N1; A3:=N2;
}
```

- 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 (A1>=0&& ··· && A3>=0 && A1+···+A3=1 && x>=0 && y>=0)
{
    N0:=0; N1:=A1; N2:=A2; N3=A3;
    F1:=isPositive(x);
    x:=x-A1;
    y:=y+A2;
    T3:=isPositive(A3+F1-1);
    R3:=isPositive(A3-F1);
    N3:=N3-A3;
    N0:=N0+T3;
    N3:=N3+R3;
    A1:=N0; A2:=N1; A3:=N2;
}
```

Termination of Integer Linear while loops

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

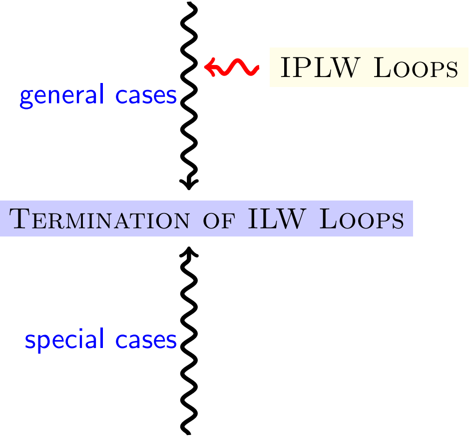
```
1:  x=x-1
2:  y=y+1
3:  if x>0 then 1 else 4
4:  end
```

```
while (A1>=0&& ··· && A3>=0 && A1+···+A3=1 && x>=0 && y>=0)
{
    N0:=0; N1:=A1; N2:=A2; N3=A3;
    ...sitive(x);
    ...
    N3:=N3+R3;
    A1:=N0; A2:=N1; A3:=N2;
}
```

Theorem

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

Termination of Integer Linear while loops



Termination of Integer Linear while loops

general cases

Undecidable for two linear pieces

```
while ( CONDITION ) {
    if ( x>0 ) then B1 else B2
}
```

Termination of ILW Loops

special cases

general cases

Undecidable for two linear pieces

```
while ( CONDITION ) {
    if ( x>0 ) then B1 else B2
}
```

Termination of ILW Loops

special cases

[Bozga et al., TACAS'12]

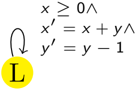
general cases

Termination of ILC Loops

special cases

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

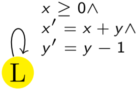
```
while (x>=0) {
    x=x+y;
    y=y-1;
}
```



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;
}
```

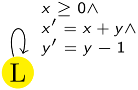


- Then, if we succeed to simulate the isPositive function with integer linear constraints we prove undecidability

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;
}
```

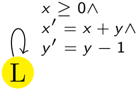


- Then, if we succeed to simulate the isPositive function with integer linear constraints we prove undecidability

$$\mathrm{T_k} = \mathrm{isPositive}(\mathrm{A_k} + \mathrm{F_k} - 1)$$

- 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

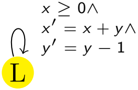
$$\mathrm{T_k} = \mathrm{isPositive}(\mathrm{A_k} + \mathrm{F_k} - 1)$$

$$\mathrm{F_k} + \mathrm{A_k} - 1 \leq 2 \cdot \mathrm{T_k} \leq \mathrm{F_k} + \mathrm{A_k}$$

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;
}
```

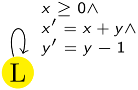


- Then, if we succeed to simulate the isPositive function with integer linear constraints we prove undecidability

$$\mathrm{R_k} = \mathrm{isPositive}(\mathrm{A_k} - \mathrm{F})$$

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

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



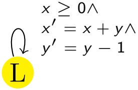
- Then, if we succeed to simulate the isPositive function with integer linear constraints we prove undecidability

$$\mathrm{R_k} = \mathrm{isPositive}(\mathrm{A_k} - \mathrm{F})$$

$$\mathrm{A_k} - \mathrm{F} \leq 2{\cdot}\mathrm{R_k} \leq \mathrm{A_k} - \mathrm{F} + 1 \wedge 0 \leq \mathrm{R_k} \leq 1$$

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

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



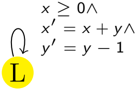
- Then, if we succeed to simulate the isPositive function with integer linear constraints we prove undecidability

$$\mathrm{F_k} = \mathrm{isPositive(x)}$$

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

$$\mathrm{F_k} = \mathrm{isPositive(x)}$$

$$\begin{array}{l} \Psi \wedge \mathrm{x} = 0 \rightarrow \mathrm{F_k} = 0 \quad \wedge \\ \Psi \wedge \mathrm{x} \geq 1 \rightarrow \mathrm{F_k} = 1 \end{array}$$

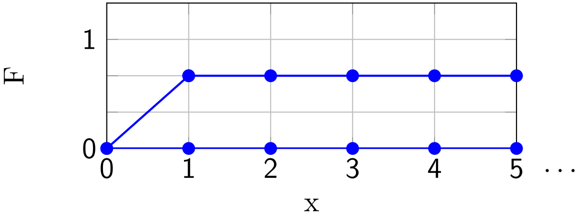
Termination of ILC Loops – The search for Ψ

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

$$\Psi_1 \equiv \mathrm{F} \leq \mathrm{x} \wedge 0 \leq \mathrm{F} \leq 1$$

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

$$\Psi_1 \equiv \mathrm{F} \leq \mathrm{x} \wedge 0 \leq \mathrm{F} \leq 1$$



Termination of ILC Loops – The search for Ψ

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

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

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

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

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

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

-

$$\Psi_2 \equiv \begin{array}{rcccl} \sqrt{2}\cdot x & \leq & A & \leq & \sqrt{2}\cdot x + 1 \quad \wedge \\ -\sqrt{2}\cdot x & \leq & B & \leq & -\sqrt{2}\cdot x + 1 \quad \wedge \\ F = A + B & & & & \end{array}$$

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

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

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

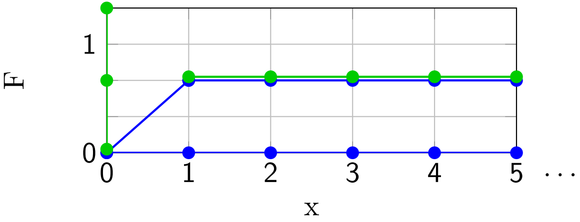
$$\Psi_2 \equiv \begin{array}{rcccl} \sqrt{2} \cdot x & \leq & A & \leq & \sqrt{2} \cdot x + 1 \quad \wedge \\ -\sqrt{2} \cdot x & \leq & B & \leq & -\sqrt{2} \cdot x + 1 \quad \wedge \\ F = A + B & & & & \end{array}$$

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

$$\Psi_1 \equiv F \le x \wedge 0 \le F \le 1$$

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

$$\Psi_2 \equiv \begin{array}{ccccc} \sqrt{2} \cdot x & \le & A & \le & \sqrt{2} \cdot x + 1 & \wedge \\ -\sqrt{2} \cdot x & \le & B & \le & -\sqrt{2} \cdot x + 1 & \wedge \\ F = A + B \end{array}$$



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

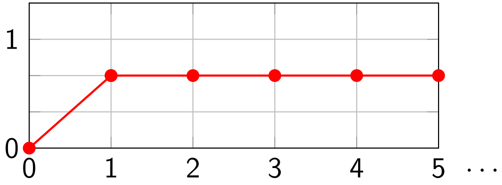
$$\Psi_1 \equiv F \leq x \wedge 0 \leq F \leq 1$$

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

-

$$\Psi_2 \equiv \begin{array}{ccccc} \sqrt{2}{\cdot}x & \leq & A & \leq & \sqrt{2}{\cdot}x + 1 & \wedge \\ -\sqrt{2}{\cdot}x & \leq & B & \leq & -\sqrt{2}{\cdot}x + 1 & \wedge \\ F = A + B \end{array}$$

- $\Psi \equiv \Psi_1 \wedge \Psi_2$



Termination of ILC Loops – The search for Ψ

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

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

- $\text{isPositive}(x) = \lceil \sqrt{2} \cdot x \rceil + \lceil -\sqrt{2} \cdot x \rceil$
- $$\Psi_2 \equiv \begin{array}{ccccc} \sqrt{2} \cdot x & \leq & A & \leq & \sqrt{2} \cdot x + 1 & \wedge \\ -\sqrt{2} \cdot x & \leq & B & \leq & -\sqrt{2} \cdot x + 1 & \wedge \\ F = A + B \end{array}$$
- $\Psi \equiv \Psi_1 \wedge \Psi_2$
- $a_0 + a_1 \cdot x_1 + \cdots + a_n \cdot x_n \leq 0$, all constants are taken from $\mathbb{Q}$

Termination of ILC Loops – The search for Ψ

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

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

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

$$\Psi_2 \equiv \begin{array}{ccccc} \sqrt{2} \cdot x & \leq & A & \leq & \sqrt{2} \cdot x + 1 & \wedge \\ -\sqrt{2} \cdot x & \leq & B & \leq & -\sqrt{2} \cdot x + 1 & \wedge \\ F = A + B \end{array}$$

- $\Psi \equiv \Psi_1 \wedge \Psi_2$
- $a_0 + a_1 \cdot x_1 + \cdots + a_n \cdot x_n \leq 0$, all constants are taken from $\mathbb{Q}$
- Undecidable when constants are taken from $\mathbb{Q} \cup \{\sqrt{2}, -\sqrt{2}\}$

Termination of ILC Loops – The search for Ψ

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

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

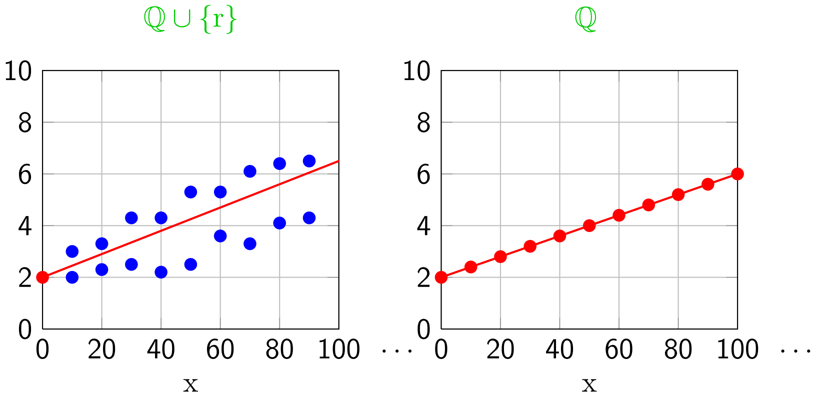
- $\text{isPositive}(x) = \lceil \sqrt{2} \cdot x \rceil + \lceil -\sqrt{2} \cdot x \rceil$
-

$$\Psi_2 \equiv \begin{array}{ccccc} \sqrt{2} \cdot x & \leq & A & \leq & \sqrt{2} \cdot x + 1 & \wedge \\ -\sqrt{2} \cdot x & \leq & B & \leq & -\sqrt{2} \cdot x + 1 & \wedge \\ F = A + B \end{array}$$

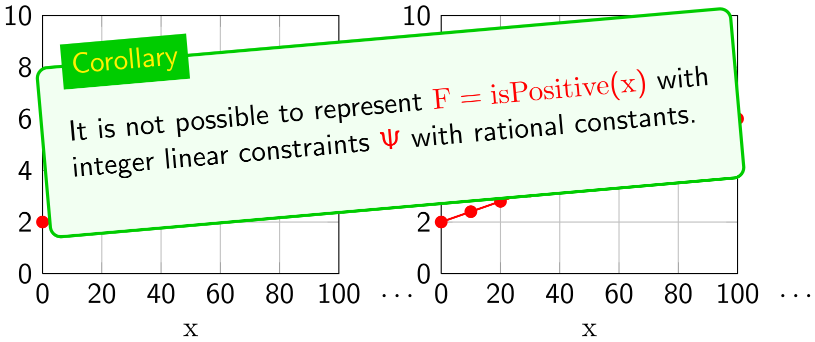
- $\Psi \equiv \Psi_1 \wedge \Psi_2$
- $a_0 + a_1 \cdot x_1 + \cdots + a_n \cdot x_n \leq 0$, all constants are taken from $\mathbb{Q}$
- Undecidable when constants are taken from $\mathbb{Q} \cup \{\sqrt{2}, -\sqrt{2}\}$
- $\mathbb{Q} \cup \{r\}$ for any irrational r

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

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



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



Corollary

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

Termination of ILC loops – a lower bound

general cases

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

Termination of ILC Loops

special cases

Termination of ILC loops – a lower bound

general cases

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

Termination of ILC Loops

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 $\mathbb{Q} \cup \{r\}$

Termination of ILC Loops

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

general cases

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

TERMINATION OF ILC LOOPS

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

general cases

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

Termination of ILC Loops

special cases

Still at least EXPSPACE-hard for deterministic constraints, but for *partial input*

Termination for *a given input* is at least EXPSPACE-hard

Decidable for octagons [Bozga et. al. TACAS'09, Iosif et. al. TACAS'12]

Summary of Contributions - II

Summary of Contributions - II

- Termination Integer Linear While loops

Summary of Contributions - II

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

Summary of Contributions - II

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

Summary of Contributions - II

- Termination 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

Summary of Contributions - II

- Termination 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
  - it is not possible to model $\text{isPositive}(x)$ with rational constants
  - but possible when allowing a *single* irrational constant
  - this leaves some hope for a positive answer, ...

Summary of Contributions - II

- Termination 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, ...
  - EXPSPACE-hard lower bound by simulating Petri-nets

Limits of cost analysis

- 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.

Summary of Contributions - II

- Amir M. Ben-Amram, Samir Genaim, and Abu Naser Masud. On the termination of integer loops. In *VMCAI 2012, Philadelphia, USA, January 25-27, 2012. Proceedings*, Lecture Notes in Computer Science. Springer, January 2012.

- Amir M. Ben-Amram, Samir Genaim, and Abu Naser Masud. On the termination of integer loops. On *ACM Transactions on Programming Langauges and Systems*, 34(4), December 2012.

Outline

Conclusion and Future Work

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

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.

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.

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.