COSTA: A COST and Termination Analyzer for Java (bytecode) Programs

Elvira Albert and Germán Puebla

École Normale Supérieure (ENS)

Paris, Sept 2, 2011
The COSTA Team

**Staff:**
E. Albert  P. Arenas  J. Correas  S. Genaim  G-Zamalloa  G. Puebla  Zanardini
UCM  UCM  UCM  UCM  UCM  UPM  UPM

**PhD Students:**
D. Alonso  A. Flores  A. Masud  D. Ramirez  J. Rojas  G. Roman
UCM  UCM  UPM  UPM  UPM  UPM
Part 1: Resource Usage Analysis (Elvira)

- Introduction to Resource Usage Analysis
- Overview of COSTA
- Generation of cost relations
- Closed-form upper and lower bounds
- Conclusions
- Future work
- Main publications
Outline of the Talk

1. **Part 1: Resource Usage Analysis (Elvira)**
   - Introduction to Resource Usage Analysis
   - Overview of COSTA
   - Generation of cost relations
   - Closed-form upper and lower bounds
   - Conclusions
   - Future work
   - Main publications

2. **Part 2: Tool Demo (Germán)**
   - Simple complexity classes
   - Memory consumption
   - Libraries
Part 1: Resource Usage Analysis
static cost analysis

bound the cost of executing program $P$ on any input data $\overline{x}$ without having to actually run $P(\overline{x})$
Introduction: Resource Usage Analysis

**static cost analysis**

Bound the cost of executing program $P$ on any input data $\overline{x}$ without having to actually run $P(\overline{x})$

- Reasoning about execution cost is difficult and error-prone.
- Cost analysis, or resource analysis or complexity analysis should be automatic.
Introduction: Resource Usage Analysis

**static cost analysis**

bound the cost of executing program $P$ on any input data $\overline{x}$ without having to actually run $P(\overline{x})$

- reasoning about execution cost is difficult and error-prone
- cost analysis, or resource analysis or complexity analysis should be automatic
- The resources considered
  - number of executed (bytecode) instructions
  - memory usage
  - billable events (number of calls to a specific method)
  - *Termination* (it guarantees the existence of an upper bound)
Different kinds of cost can be considered:
- worst case $\rightarrow$ upper bound
- average case $\rightarrow$ requires probabilistic study
- best case $\rightarrow$ lower bound
Different kinds of cost can be considered:
- worst case → upper bound
- average case → requires probabilistic study
- best case → lower bound

Two classes of upper bounds can be considered:
- non-asymptotic (or concrete, or micro-analysis)
- asymptotic (or macro-analysis)
Different kinds of cost can be considered:
- worst case \(\rightarrow\) upper bound
- average case \(\rightarrow\) requires probabilistic study
- best case \(\rightarrow\) lower bound

Two classes of upper bounds can be considered:
- non-asymptotic (or concrete, or micro-analysis)
- asymptotic (or macro-analysis)

Analysis results can be
- platform-independent
- platform-dependent \(\rightarrow\) WCET
Work on automatic cost analysis dates back to 1975, with the seminal work of Wegbreit.

His system, metric was able to compute:
- interesting results, but for
- restricted class of functional programs

Also, the seminal work on abstract interpretation [Cousot & Cousot'77] mentions performance analysis as an application.

Since then, a number of analyses and systems have been built which extend the capabilities of cost analysis:
- functional programs [Le Metayer’88, Rosendahl’89, Wadler’88, Sands’95, Benzinger’04]
- logic programs [Debray and Lin’93, Navas et al’07]
- imperative programs [Adachi et al’79, Albert et al’07]
A classical approach [Wegbreit’75] to cost analysis consists of:

1. expressing the cost of a program part in terms of other program parts, thus obtaining *recurrence relations*
2. solving the relations by obtaining a *closed-form* for the cost in terms of the input arguments
A classical approach [Wegbreit’75] to cost analysis consists of:

1. expressing the cost of a program part in terms of other program parts, thus obtaining *recurrence relations*
2. solving the relations by obtaining a *closed-form* for the cost in terms of the input arguments

The current situation is that

- Most work has concentrated on the first phase
- The difficulties of the second phase have been overseen
- Practical usage of cost analysis requires both!
- In COSTA we address both phases.
**COSTA**: COSt and Termination analyzer for Java Bytecode

**INPUT**: bytecode + cost model

**OUTPUT**: upper bound on resource consumption + termination info
List f(String x[]) {
    List l=null;
    int i=0;
    while (i<x.length) {
        l = new List(g(x,i),l);
        i += COND?1:3;
    }
    return l;
}

Integer g(String x[], int i) {
    int r=0;
    for(int j=0; j<i; j++) {
        Integer t = new Integer(x[i]);
        r += t.intValue();
    }
    return new Integer(r);
}
List f(String x[]) {
  List l=null;
  int i=0;
  while (i<x.length) {
    l = new List(g(x,i),l);
    i += COND?1:3;
  }
  return l;
}

Integer g(String x[], int i) {
  int r=0;
  for(int j=0; j<i; j++) {
    Integer t = new Integer(x[i]);
    r += t.intValue();
  }
  return new Integer(r);
}
List f(String x[]) {
   List l=null;
   int i=0;
   while (i<x.length) {
      l = new List(g(x,i),l);
      i += COND?1:3;
   }

   return l;
}

Integer g(String x[], int i) {
   int r=0;
   for(int j=0; j<i; j++) {
      Integer t = new Integer(x[i]);
      r += t.intValue();
   }

   return new Integer(r);
}
List f(String x[]) {
    List l = null;
    int i = 0;
    while (i < x.length) {
        l = new List(g(x, i), l);
        i += COND ? 1 : 3;
    }
    return l;
}

Integer g(String x[], int i) {
    int r = 0;
    for (int j = 0; j < i; j++) {
        Integer t = new Integer(x[i]);
        r += t.intValue();
    }
    return new Integer(r);
}
List f(String x[]) {
  List l = null;
  int i = 0;
  while (i < x.length) {
    l = new List(g(x, i), l);
    i += COND ? 1 : 3;
  }
  return l;
}

Integer g(String x[], int i) {
  int r = 0;
  for (int j = 0; j < i; j++) {
    Integer t = new Integer(x[j]);
    r += t.intValue();
  }
  return new Integer(r);
}
List f(String x[]) {
    List l=null;
    int i=0;
    while (i<x.length) {
        l = new List(g(x,i),l);
        i += COND?1:3;
    }

    return l;
}

Integer g(String x[], int i) {
    int r=0;
    for(int j=0; j<i; j++) {
        Integer t = new Integer(x[i]);
        r += t.intValue();
    }

    return new Integer(r);
}
List f(String x[]) {
    List l=null;
    int i=0;
    while (i<x.length) {
        l = new List(g(x,i),l);
        i += COND?1:3;
    }
    return l;
}

Integer g(String x[], int i) {
    int r=0;
    for(int j=0; j<i; j++) {
        Integer t = new Integer(x[i]);
        r += t.intValue();
    }
    return new Integer(r);
}
List f(String x[]) {
    List l=null;
    int i=0;
    while (i<x.length) {
        l = new List(g(x,i),l);
        i += COND?1:3;
    }
    return l;
}

Integer g(String x[], int i) {
    int r=0;
    for (int j=0; j<i; j++) {
        Integer t = new Integer(x[i]);
        r += t.intValue();
    }
    return new Integer(r);
}
List f(String x[]) {  
    List l=null;  
    int i=0;  
    while (i<x.length) {  
        l = new List(g(x,i),l);  
        i += COND?1:3;  
    }  
    return l;  
}

Integer g(String x[], int i) {  
    int r=0;  
    for(int j=0; j<i; j++) {  
        Integer t = new Integer(x[i]);  
        r += t.intValue();  
    }  
    return new Integer(r);  
}

Worst-Case (UB)
\[ f(x) = \frac{x(x+1)}{2} + c \cdot x + a \cdot x \]
List \( f(\text{String } x[]) \) {
    List l=null;
    int i=0;
    while (i<x.length) {
        l = new List(g(x,i),l);
        i += COND?1:3;
    }
    return l;
}

Integer g(\text{String } x[], \text{int } i) {
    int r=0;
    for (int j=0; j<i; j++) {
        Integer t = new Integer(x[j]);
        r += t.intValue();
    }
    return new Integer(r);
}

Worst-Case (UB)

\[
f(x) = b \cdot \frac{x(x+1)}{2} + c \cdot x + a^*x^3
\]

Best-Case (LB)

\[
f(x) = b \cdot \frac{x(x+3)}{18} + c \cdot \frac{x}{3} + a^*x^3
\]
COSTA - Worst/Best Case

1. Program
2. Static Analysis
3. Cost Relations
4. CRs Solver
5. Best/Worst Case
List f(String x[]) {
    List l=null;
    int i=0;
    while (i<x.length) {
        l = new List(g(x,i),l);
        i += COND?1:3;
    }
    return l;
}
List $f(String \ x[])$ {
    List $l$ = null;
    int $i$ = 0;
    while ($i < x.length$) {
        $l$ = new List($g(x,i), l$);
        $i$ += COND?1:3;
    }
    return $l$;
}
List \( f(\text{String } x[]) \) 
\[
\begin{align*}
\text{List } l &= \text{null}; \\
\text{int } i &= 0; \\
\text{while } (i < x.\text{length}) \
\quad l &= \text{new List}(g(x,i),l); \\
\quad i &= + \text{COND}?1:3; \\
\end{align*}
\]
\[
\text{return } l;
\]
\]

\[
f(x) = A(x, i) \quad \{i=0, x \geq 0\}
\]
\[
A(x, i) = 0 \quad \{i \geq x\}
\]
\[
A(x, i) = g(x, i) + A(x, i') \quad \{i < x, i+1 \leq i' \leq i+3\}
\]
\[
g(x, i) = B(i, j) + \quad \{j=0\}
\]
\[
B(i, j) = 0 \quad \{j \geq i\}
\]
\[
B(i, j) = B(i, j') \quad \{j < i, j' = j+1\}
\]
Program

Static Analysis

Cost Relations

CRs Solver

Best/Worst Case

List \( f(String \ x[]) \) {
    List \( l = null \);
    int \( i = 0 \);
    while \( (i < x.length) \) {
        \( l = new \) List \( (g(x,i),l) \);
        \( i += \) COND?1:3;
    }

    return \( l \);
}

\( f(x) = A(x,i) \) \hspace{1cm} \{ i = 0, x \geq 0 \}

\( A(x,i) = 0 \) \hspace{1cm} \{ i \geq x \}

\( A(x,i) = g(x,i) + A(x,i') \) \hspace{1cm} \{ i < x, i+1 \leq i' \leq i+3 \}

\( g(x,i) = B(i,j) + \) \hspace{1cm} \{ j = 0 \}

\( B(i,j) = 0 \) \hspace{1cm} \{ j \geq i \}

\( B(i,j) = B(i,j') + \) \hspace{1cm} \{ j < i, j' = j+1 \}
List $f(String \ x[])$ {
    List $l=null;$
    int $i=0;$
    while ($i<x.length$) {
        $l = new \ List(g(x,i),l);$  \[a\]
        $i += COND?1:3;$
    }
    return $l;$
}

$f(x) = A(x, i)$ \[\{i=0, x \geq 0\}\]

$A(x, i) = 0$ \[\{i \geq x\}\]

$A(x, i) = g(x, i) + A(x, i')$ \[\{i < x, i+1 \leq i' \leq i+3\}\]

$g(x, i) = B(i, j) + c$ \[\{j=0\}\]

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

$B(i, j) = B(i, j') + B(i, j''')$ \[\{j < i, j' = j+1\}\]
List $f(String \ x[])$ {
    $List \ l=null;$
    int $i=0;$
    while ($i<x.length$) {
        $l = new \ List(g(x,i),l);$
        $i += COND?1:3;$
    }
    return $l;$
}

$$f(x) = A(x, i) \quad \{i=0, x \geq 0\}$$

$$A(x, i) = 0 \quad \{i \geq x\}$$
$$A(x, i) = g(x, i) + A(x, i') \quad \{i < x, i+1 \leq i' \leq i+3\}$$

$$g(x, i) = B(i, j) + \quad \{j=0\}$$
$$B(i, j) = 0 \quad \{j \geq i\}$$
$$B(i, j) = B(i, j') \quad \{j < i, j' = j+1\}$$
List f(String x[]) {
    List l = null;
    int i = 0;
    while (i < x.length) {
        l = new List(g(x,i),l);
        i += COND ? 1 : 3;
    }
    return l;
}

f(x) = A(x, i) \quad \{i=0, x \geq 0\}
A(x, i) = 0 \quad \{i \geq x\}
A(x, i) = g(x, i) + A(x, i') \quad \{i < x, i+1 \leq i' \leq i+3\}
g(x, i) = B(i, j) + \circ \quad \{j=0\}
B(i, j) = 0 \quad \{j \geq i\}
B(i, j) = \bullet + B(i, j') \quad \{j < i, j' = j+1\}
Program

Static Analysis

Cost Relations

CRs Solver

Best/Worst Case

Integer g(String x[], int i) {
    int r=0;
    for(int j=0; j<i; j++) {
        Integer t = new Integer(x[i]);
        r += t.intValue();
    }
    return new Integer(r);
}

f(x) = A(x, i) \quad \{i=0, x\geq 0\}
A(x, i) = 0 \quad \{i\geq x\}
A(x, i) = g(x, i) + A(x, i') \quad \{i\leq x, x+1 \leq i' \leq x+3\}
g(x, i) = B(i, j) \quad \{j=0\}
B(i, j) = 0 \quad \{j \geq i\}
B(i, j) = B(i, j) + B(i, j') \quad \{j<i, j'=j+1\}
Program

Static Analysis

Cost Relations

CRs Solver

Best/Worst Case

Integer g(String x[], int i) {
    int r=0;
    for(int j=0; j<i; j++) {
        Integer t = new Integer(x[i]);
        r += t.intValue();
    }
    return new Integer(r);
}

f(x) = A(x, i) \quad \{i=0, x \geq 0\}
A(x, i) = 0 \quad \{i \geq x\}
A(x, i) = g(x, i) + A(x, i') \quad \{i < x, i+1 \leq i' \leq i+3\}
g(x, i) = B(i, j) \quad \{j=0\}
B(i, j) = 0 \quad \{j \geq i\}
B(i, j) = B(i, j') + B(i', j') \quad \{j < i, j' = j+1\}
Program

Static Analysis

Cost Relations

CRs Solver

Best/Worst Case

Integer g(String x[], int i) {
  int r=0;
  for(int j=0; j<i; j++) {
    Integer t = new Integer(x[i]);
    r += t.intValue();
  }
  return new Integer(r);
}

f(x) = A(x, i) \quad \{i=0, x \geq 0\}
A(x, i) = 0 \quad \{i \geq x\}
A(x, i) = g(x, i) + A(x, i') \quad \{i < x, i+1 \leq i' \leq i+3\}

\begin{align*}
  g(x, i) &= B(i, j) + \textcolor{red}{c} \quad \{j=0\} \\
  B(i, j) &= 0 \quad \{j \geq i\} \\
  B(i, j) &= \textcolor{blue}{b} + B(i, j') \quad \{j < i, j' = j+1\}
\end{align*}
Program \[\rightarrow\] Static Analysis \[\rightarrow\] Cost Relations \[\rightarrow\] CRs Solver \[\rightarrow\] Best/Worst Case

**Worst-Case (UB)**

\[f(x) = b \times x^2 + c \times x + a \times x\]
\[A(x, i) = b \times (x - i) \times x + c \times (x - i) + a \times x\]
\[g(x, i) = b \times i + c\]
\[B(i, j) = b \times (i - j)\]

\[f(x) = A(x, i)\quad \{i=0, x \leq 0\}\]
\[A(x, i) = 0\quad \{i \geq x\}\]
\[A(x, i) = g(x, i) + a + A(x, i')\quad \{i < x, i+1 \leq i' \leq i+3\}\]

\[g(x, i) = B(i, j) + c\quad \{j=0\}\]
\[B(i, j) = 0\quad \{j \geq i\}\]
\[B(i, j) = b + B(i, j')\quad \{j < i, j' = j+1\}\]
Program

Static Analysis

Cost Relations

CRs Solver

Best/Worst Case

```
Integer g(String x[], int i) {
    int r = 0;
    for (int j = 0; j < i; j++) {
        Integer t = new Integer(x[i]);
        r += t.intValue();
    }
    return new Integer(r);
}
```

```
f(x) = A(x, i) \quad \{i=0, x \geq 0\}
A(x, i) = 0 \quad \{i \geq x\}
A(x, i) = g(x, i) + A(x, i') \quad \{i < x, i+1 \leq i' \leq i+3\}
g(x, i) = B(i, j) + \quad \{j=0\}
B(i, j) = 0 \quad \{j \geq i\}
B(i, j) = g(i, j') + \quad \{j < i, j' = j+1\}
```
Solving CRs - Computer Algebra Systems

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

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

\[ A(x, i) = g(x, i) + A(x, i') \quad \{ i < x, i + 1 \leq i' \leq i+3 \} \]

\[ g(x, i) = B(i, j) + \quad \{ j=0 \} \]

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

\[ B(i, j) = b + B(i, j') \quad \{ j < i, j' = j+1 \} \]
Solving CRs - Computer Algebra Systems

\[
f(x) = A(x, i) \quad \{i=0, x \geq 0\}
\]
\[
A(x, i) = 0 \quad \{i \geq x\}
\]
\[
A(x, i) = g(x, i) + A(x, i') \quad \{i < x, i + 1 \leq i' \leq i + 3\}
\]
\[
g(x, i) = B(i, j) + a \quad \{j=0\}
\]
\[
B(i, j) = 0 \quad \{j \geq i\}
\]
\[
B(i, j) = b + B(i, j') \quad \{j < i, j' = j + 1\}
\]

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

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

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

\[ A(x, i) = g(x, i) + a + A(x, i') \] \quad \{i < x, i + 1 \leq i' \leq i + 3\}

\[ g(x, i) = B(i, j) + c \] \quad \{j = 0\}

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

\[ B(i, j) = b + B(i, j') \] \quad \{j < i, j' = j + 1\}

CAS can obtain an exact closed-form solution for:

\[ P(0) = 0 \]

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

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

\[ f(x) = A(x, i) \quad \{ i=0, x \geq 0 \} \]
\[ A(x, i) = 0 \quad \{ i \geq x \} \]
\[ A(x, i) = g(x, i) + A(x, i') \quad \{ i < x, i + 1 \leq i' \leq i+3 \} \]
\[ g(x, i) = B(i, j) + c \quad \{ j = 0 \} \]
\[ B(i, j) = 0 \quad \{ j \geq i \} \]
\[ B(i, j) = b + B(i, j') \quad \{ j < i, j' = j+1 \} \]

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

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

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

\[ A(x, i) = g(x, i) + A(x, i') \quad \{i < x, i + 1 \leq i' \leq i + 3\} \]

\[ g(x, i) = B(i, j) + c \quad \{j = 0\} \]

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

\[ B(i, j) = b + B(i, j') \quad \{j < i\} \]

Two possible runs for \( A(10, 0) \)

- \( A(10, 0) \rightarrow A(10, 1) \rightarrow A(10, 4) \rightarrow \cdots \)
- \( A(10, 0) \rightarrow A(10, 4) \rightarrow A(10, 6) \rightarrow \cdots \)

CRs are not deterministic

CRs have multiple arguments

CRs have multiple (not mutually exclusive) equations

Thus, CRs often do not have an exact solution

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

\[ f(x) = A(x, i) \]

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

\[ A(x, i) = g(x, i) + a + A(x, i') \quad \{ i > x \} \]

\[ g(x, i) = B(i, j) + c \quad \{ j = 0 \} \]

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

\[ B(i, j) = b + B(i, j') \quad \{ j < i, j' = j + 1 \} \]

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

\[ f(x) = A(x, i) \quad \{i=0, x \geq 0\} \]
\[ A(x, i) = 0 \quad \{i \geq x\} \]
\[ A(x, i) = g(x, i) + a \cdot A(x, i') \quad \{i < x, i + 1 \leq i' \leq i + 3\} \]
\[ g(x, i) = B(i, j) + c \quad \{j = 0\} \]
\[ B(i, j) = 0 \quad \{j \geq i\} \]
\[ B(i, j) = b \cdot B(i, j') \quad \{j < i, j' = j + 1\} \]

- Why not using directly Computer Algebra Systems?
- CRs are not deterministic
- CRs have multiple arguments
- CRs have multiple (not mutually exclusive) equations
\[ f(x) = A(x, i) \] \quad \{i=0, x \geq 0\}

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

\[ A(x, i) = g(x, i) + a + A(x, i') \] \quad \{i < x, i + 1 \leq i' \leq i + 3\}

\[ g(x, i) = B(i, j) + c \] \quad \{j = 0\}

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

\[ B(i, j) = b + B(i, j') \] \quad \{j < i, j' = j + 1\}

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

\[ f(x) = A(x, i) \quad \{ i=0, x \geq 0 \} \]
\[ A(x, i) = 0 \quad \{ i \geq x \} \]
\[ A(x, i) = g(x, i) + a + A(x, i') \quad \{ i < x, i + 1 \leq i' \leq i + 3 \} \]
\[ g(x, i) = B(i, j) + c \quad \{ j = 0 \} \]
\[ B(i, j) = 0 \quad \{ j \geq i \} \]
\[ B(i, j) = b + B(i, j') \quad \{ j < i, j' = j + 1 \} \]

CAS can obtain an exact closed-form solution for:

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

deterministic, 1 base-case, 1 recursive case, 1 argument
\[ f(x) = A(x, i) \]

\[ A(x, i) = 0 \]
\[ A(x, i) = g(x, i) + A(x, i') \]
\[ g(x, i) = B(i, j) + c \]
\[ B(i, j) = 0 \]
\[ B(i, j) = b + B(i, j') \]

\[ \{i=0, x \geq 0\} \]
\[ \{i \geq x\} \]
\[ \{i < x, i + 1 \leq i' \leq i + 3\} \]
\[ \{j = 0\} \]
\[ \{j \geq i\} \]
\[ \{j < i, j' = j + 1\} \]
\[ f(x) = A(x, i) \]
\[ A(x, i) = 0 \quad \{i=0, x \geq 0\} \]
\[ A(x, i) = g(x, i) + a + A(x, i') \quad \{i \geq x\} \]
\[ g(x, i) = B(i, j) + c \quad \{j=0\} \]
\[ B(i, j) = 0 \quad \{j \geq i\} \]
\[ B(i, j) = b + B(i, j') \quad \{j < i, j' = j + 1\} \]

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

\[ b \rightarrow b \rightarrow b \rightarrow b \rightarrow b \]
Solving CRs (UBs) - PUBLS

\[ f(x) = A(x, i) \]
\[ A(x, i) = 0 \]
\[ A(x, i) = g(x, i) + a + A(x, i') \]
\[ g(x, i) = B(i, j) + c \]
\[ B(i, j) = 0 \]
\[ B(i, j) = b + B(i, j') \]

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

- How many \( b \) has this chain?
\[ f(x) = A(x, i) \]
\[ A(x, i) = 0 \quad \{i = 0, x \geq 0\} \]
\[ A(x, i) = g(x, i) + a + A(x, i') \quad \{i \geq x\} \]
\[ g(x, i) = B(i, j) + c \quad \{j = 0\} \]
\[ B(i, j) = 0 \quad \{j \geq i\} \]
\[ B(i, j) = b + B(i, j') \quad \{j < i, j' = j + 1\} \]

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

  ![Diagram](image)

- How many \( b \) has this chain?
Solving CRs (UBs) - PUBS

\[ f(x) = A(x, i) \quad \{ i = 0, x > 0 \} \]

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

\[ A(x, i) = g(x, i) + a + A(x, i') \quad \{ i < x, i + 1 \leq i' \leq i + 3 \} \]

\[ g(x, i) = B(i, j) + c \quad \{ j = 0 \} \]

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

\[ B(i, j) = b + B(i, j') \quad \{ j < i, j' = j + 1 \} \]

\[ \exists \hat{f}. \forall i, j, j'. \varphi \models \hat{f}(i, j) \geq 0 \land \hat{f}(i, j) - \hat{f}(i, j') \geq 1 \]

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

- How many \( b \) has this chain?
\[ f(x) = A(x, i) \]
\[ A(x, i) = 0 \quad \{ i = 0, x \geq 0 \} \]
\[ A(x, i) = g(x, i) + a + A(x, i') \quad \{ i \geq x \} \]
\[ g(x, i) = B(i, j) + c \quad \{ j = 0 \} \]
\[ B(i, j) = 0 \quad \{ j \geq i \} \]
\[ B(i, j) = b + B(i, j') \quad \{ j < i, j' = j + 1 \} \]

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

\[ \hat{f}(i_0, j_0) = \text{nat}(i_0 - j_0) \]

- How many \( b \) has this chain?
Solving CRs (UBs) - PUBS

\[ f(x) = A(x, i) \]
\[ A(x, i) = 0 \]
\[ A(x, i) = g(x, i) \]
\[ g(x, i) = B(i, j) \]

- \( f(x) = A(x, i) \) for \( \{ i=0, x>0 \} \)
- \( A(x, i) = 0 \)
- \( A(x, i) = g(x, i) \)
- \( g(x, i) = B(i, j) \)

\[ B(i_0, j_0) = b \cdot \text{nat}(i_0 - j_0) \]
\[ \text{Worst-Case} \]

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

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

\[ b \rightarrow b \rightarrow b \rightarrow b \rightarrow b \]

- How many \( b \) has this chain?

\[ \hat{f}(i_0, j_0) = \text{nat}(i_0 - j_0) \]
Solving CRs (UBs) - PUBS

\[ f(x) = A(x, i) \quad \text{Worst-Case} \quad \{i=0, x>0\} \]

\[ A(x, i) = 0 \]
\[ A(x, i) = g(x, i) \]
\[ g(x, i) = B(i, j) \]

\[ B(i_0, j_0) = b \times 2^{\text{nat}(i_0-j_0)} \quad \leq i' \leq i+3 \]

\[ B(i, j) = 0 \quad \{j \geq i\} \]
\[ B(i, j) = b + B(i, j') + B(i, j') \quad \{j < i, j' = j+1\} \]

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

  \[
  \begin{array}{cccc}
  b & b & b & b \\
  \end{array}
  \]

- How many \( b \) has this chain?

  \[ \hat{f}(i_0, j_0) = \text{nat}(i_0-j_0) \]
Solving CRs (UBs) - PUBS

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

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

\[
A(x, i) = g(x, i) + a + A(x, i') \quad \{i < x, i + 1 \leq i' \leq i+3\}
\]

\[
g(x, i) = B(i, j) + c \quad \{j=0\}
\]
Solving CRs (UBs) - PUNS

\[ f(x) = A(x, i) \]

\[ A(x, i) = A(x, i) = 0 \]

\[ A(x, i) = g(x, i) + A(x, i') \]

\[ g(x, i) = b \times \text{nat}(i - j) + c \]

\{i = 0, x \geq 0\}

\{i \geq x\}

\{i < x, i + 1 \leq i' \leq i + 3\}

\{j = 0\}
Solving CRs (UBs) - PUBS

\[ f(x) = A(x, i) \]
\[ A(x, i) = 0 \]
\[ A(x, i) = g(x, i) + a + A(x, i') \]
\[ g(x, i) = b \cdot \text{nat}(i-j) + c \]

Worst-Case for \( g \)

\[ g(x_0, i_0) = b \cdot \text{nat}(i_0) + c \]
Solving CRs (UBs) - PUBS

\[ f(x) = A(x, i) \]

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

\[
A(x, i) = g(x, i) + a + A(x, i') \quad \{ i \geq x \}
\]

\[
g(x, i) = b \ast \text{nat}(i-j) + c \quad \{ j = 0 \}
\]

Worst-Case for \( g \)

\[
g(x_0, i_0) = b \ast \text{nat}(i_0) + c
\]
\[
f(x) = A(x, i) \quad \{i=0, x \geq 0\}
\]
\[
A(x, i) = 0 \quad \{i \geq x\}
\]
\[
A(x, i) = b \ast \text{nat}(i) + c + a + A(x, i') \quad \{i < x, i + 1 \leq i' \leq i + 3\}
\]
\[
g(x, i) = b \ast \text{nat}(i - j) + c \quad \{j = 0\}
\]
\[ f(x) = A(x, i) \]
\[ A(x, i) = 0 \quad \{ i = 0, x \geq 0 \} \]
\[ A(x, i) = b \times \text{nat}(i) + c + a + A(x, i') \quad \{ i \geq x \} \]
\[ g(x, i) = b \times \text{nat}(i - j) + c \quad \{ j = 0 \} \]

- An evaluation for \( A(x_0, i_0) \) looks like:
\[ f(x) = A(x, i) \]

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

\[ A(x, i) = b \cdot \text{nat}(i) + \text{c} + a + A(x, i') \quad \{ i \geq x \} \]

\[ A(x, i) = b \cdot \text{nat}(i) + \text{c} \quad \{ i < x, i + 1 \leq i' \leq i + 3 \} \]

\[ g(x, i) = b \cdot \text{nat}(i - j) + \text{c} \quad \{ j = 0 \} \]

- An evaluation for \( A(x_0, i_0) \) looks like:

  ![Evaluation Diagram](image)

- There are at most \( \text{nat}(x_0 - i_0) \) circles (\textit{ranking function})
\[ f(x) = A(x, i) \]
\[
A(x, i) = 0 \quad \{i = 0, x \geq 0\}
\]
\[
A(x, i) = b \cdot \text{nat}(i) + c + a + A(x, i') \quad \{i \geq x\}
\]
\[
g(x, i) = b \cdot \text{nat}(i - j) + c \quad \{j = 0\}
\]

- An evaluation for \( A(x_0, i_0) \) looks like:

- There are at most \( \text{nat}(x_0 - i_0) \) circles (*ranking function*)
Solving CRs (UBs) - PUBS

\[ f(x) = A(x, i) \]

\[ A(x, i) = 0 \]
\[ A(x, i) = b \times \text{nat}(i) + c + a + A(x, i') \]
\[ g(x, i) = b \times \text{nat}(i-j) + c \]

- An evaluation for \( A(x_0, i_0) \) looks like:

- There are at most \( \text{nat}(x_0 - i_0) \) circles (\textit{ranking function})

- Worst-case is \( \ast \text{nat}(x_0 - i_0) \)
Solving CRs (UBs) - PUBS

\[ f(x) = A(x, i) \]

\[ A(x, i) = \begin{cases} 0 & \{i = 0, x \geq 0\} \\ b \ast \text{nat}(i) + c + a + A(x, i') & \{i \geq x\} \\ a + A(x, i') & \{i < x, i + 1 \leq i' \leq i + 3\} \end{cases} \]

\[ g(x, i) = b \ast \text{nat}(i - j) + c \]

\[ \{j = 0\} \]

- Inferring box (or maximizing expression)

  - What is the maximum value that \( i \) can take in terms of the initial values \( \langle x_0, i_0 \rangle \)? It is \( x_0 \).
Solving CRs (UBs) - PUBS

\[ f(x) = A(x, i) \]
\[ A(x, i) = 0 \quad \{i = 0, x \geq 0\} \]
\[ A(x, i) = \textcolor{red}{b} \ast \text{nat}(i) + \textcolor{green}{c} + \textcolor{blue}{a} + A(x, i') \quad \{i \geq x\} \]
\[ A(x, i') = b \ast \text{nat}(i) + c + A(x, i') \quad \{i < x, i + 1 \leq i' \leq i + 3\} \]
\[ g(x, i) = \textcolor{red}{b} \ast \text{nat}(i-j) + \textcolor{green}{c} \quad \{j = 0\} \]

- An evaluation for \( A(x_0, i_0) \) looks like:
  - There are at most \( \text{nat}(x_0 - i_0) \) circles (ranking function)
  - Worst-case is \( \ast \text{nat}(x_0 - i_0) \)

- What is the maximum value that \( i \) can take in terms of the initial values \( \langle x_0, i_0 \rangle \)? It is \( x_0 \).

- Infer an invariant \( \langle A(x_0, i_0) \leadsto A(x, i), \Psi \rangle \)
\[ f(x) = A(x, i) \]
\[ A(x, i) = \begin{cases} 0 & \{i = 0, x \geq 0\} \\ \text{nat}(i) + c + a + A(x, i') & \{i \geq x\} \\ \text{nat}(i - j) + c & \{j = 0\} \end{cases} \]
\[ g(x, i) = \text{nat}(i - j) + c \]

- An invariant: \(\langle A(x_0, i_0) \leadsto A(x, i), \Psi \rangle\)
- Use (parametric) integer programming to maximize \(i\) w.r.t \(\Psi \land \varphi\) and the parameters \(\langle x_0, i_0 \rangle\).

What is the maximum value that \(i\) can take in terms of the initial values \(\langle x_0, i_0 \rangle\)? It is \(x_0\).
Solving CRs (UBs) - PUBS

\[ f(x) = A(x, i) \]

\[ A(x, i) = 0 \quad \{i = 0, x \geq 0\} \]
\[ A(x, i) = b \times \text{nat}(i) + c + a + A(x, i') \quad \{i \geq x\} \]
\[ A(x, i) = b \times \text{nat}(i) + c + a + A(x, i') \quad \{i < x, i + 1 \leq i' \leq i + 3\} \]
\[ g(x, i) = b \times \text{nat}(i - j) + c \quad \{j = 0\} \]

- An evaluation for \( A(x_0, i_0) \) looks like:

- There are at most \( \text{nat}(x_0 - i_0) \) circles (\textit{ranking function})

- \( A(x_0, i_0) = (b \times \text{nat}(x_0) + c + a) \times \text{nat}(x_0 - i_0) \)
Cost and Termination

- Termination: find ranking functions for all loops in the program
- Termination $\rightarrow$ Bounded resource consumption
- Cost (for number of instructions) $\rightarrow$ Termination
Conclusions

- **Cost and Termination**
  - Termination: find ranking functions for all loops in the program
  - Termination $\rightarrow$ Bounded resource consumption
  - Cost (for number of instructions) $\rightarrow$ Termination

- **COSTA** has been the first cost and termination analyzer for sequential Java Bytecode
  - It deals with Java libraries
  - It checks termination and computes upper bounds
  - It allows assertions on upper bounds (and thus termination)
Conclusions

Cost and Termination
- Termination: find ranking functions for all loops in the program
- Termination $\rightarrow$ Bounded resource consumption
- Cost (for number of instructions) $\rightarrow$ Termination

**COSTA** has been the first cost and termination analyzer for sequential Java Bytecode
- It deals with Java libraries
- It checks termination and computes upper bounds
- It allows assertions on upper bounds (and thus termination)

**PUBS** has been the first generic cost relation solver
- It is written in Prolog and uses PPL
- It can be connected to Maxima for further precision in some cases
- It is currently able to compute upper and lower bounds
European Projects:
  UPM (Germán Puebla) + UCM (Elvira Albert, Lou 83)

MEC Projects:
  UPM (Manuel Hermenegildo)

CAM Projects:
  UCM (Francisco J. López-Fraguas)
Some advanced topics:

- Modular and incremental analysis
- Support dynamically allocated data (numeric and reference fields) in cost/termination analysis of OO bytecode
- Asymptotic upper bound
- Checking of resource usage specifications

Present and future work:

- Handle concurrency during static analysis
  - X10 language (Java-like syntax, different concurrency)
  - ABS (successor of Creol)
- Combine static and dynamic techniques
- Combine testing and resource usage
Main Publications of COSTA

- ESOP’07/LNCS + TCS’11/Elsevier: Generation of cost relations
- SAS’08/LNCS + JAR’11/Springer: Closed-form upper bounds
- ISMM’07/ACM + ISMM’09/ACM + ISMM’10/ACM: Memory consumption
- FM’09/LNCS + SAS’10/LNCS: Treatment of Heap-Allocated Data
- FOPARA’09/LNCS: Checking upper bounds
- APLAS’09/LNCS: Asymptotic bounds
- VMCAI’11/LNCS: Lower-bounds
- LCTES’11/ACM: Concurrency in cost analysis
- FM’11/LNCS: Combination of static and dynamic techniques
- PEPM’11/ACM: Verified resource guarantees
- Several tool demos and system descriptions: Bytecode, PROLE, etc.