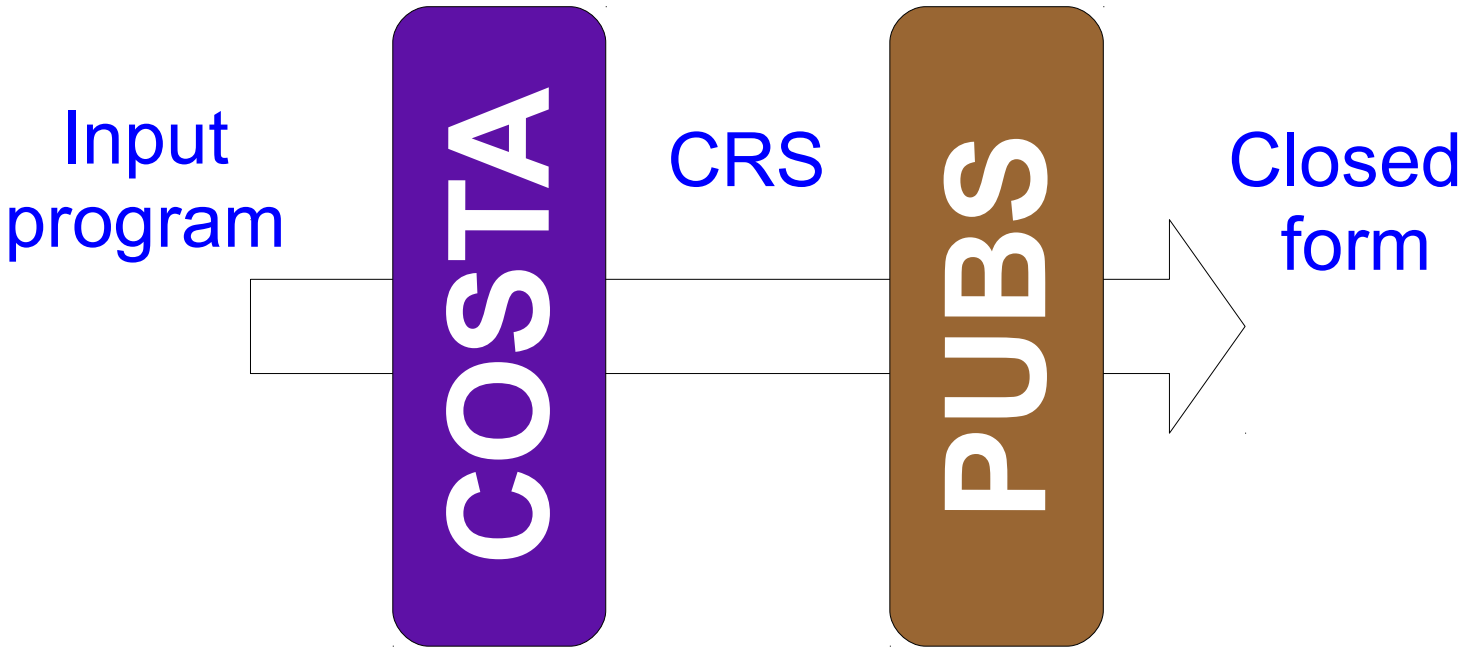
Introduction to the COSTA Web Interface

Manuel Montenegro

Dpto. Sistemas Informáticos y Computación (DSIC)
Facultad de Informática
Universidad Complutense de Madrid (UCM)

Reliability of Software Systems
Master Computing Science

Overview of COSTA arquitecture



Syntax of CRS

- A Cost Relation System (CRS) is a finite set of equations of the form:

$$C(\overline{x}) = \mathtt{exp} + \sum_{i=1}^{n} C(\overline{y_i}) \qquad \varphi$$

- The syntax of basic cost expressions is defined as follows:

$$\begin{aligned} \mathtt{exp} ::= & r \mid \mathsf{nat}(l) \mid \mathtt{exp} + \mathtt{exp} \mid \mathtt{exp} * \mathtt{exp} \mid \mathtt{exp}^r \mid log_n(\mathtt{exp}) \\ & \mid n^{\mathtt{exp}} \mid max(S) \mid \mathtt{exp} - r \end{aligned}$$

where $r \in \mathbb{Q}$, $n \in \mathbb{N}$ and l is a linear expression: $a_0 + a_1 x_1 + \ldots + a_n x_n$

Syntax of PUBS declarations

$$C(\overline{x}) = \texttt{exp} + \sum_{i=1}^{n} C(\overline{y_i}) \qquad \varphi$$

```
eq(Head, BasicExp, [RecCalls], [GuardConditions]).
```

Syntax of PUBS declarations

$$C(\overline{x}) = \texttt{exp} + \sum_{i=1}^{n} C(\overline{y_i}) \qquad \varphi$$

```
eq(Head, BasicExp, [RecCalls], [GuardConditions]).
```

- Head: `c(X,Y,...,Z)`
- Basic expressions take the following syntax:

```
BasicExp ::= PosInt | nat(LinearExpression) | BasicExp * BasicExp
           |  BasicExp + BasicExp  | pow(PosInt,BasicExp)
           |  log(PosInt,BasicExp) | max([BasicExp1,...])
           |  BasicExp / PosInt    | BasicExp - PosInt
```

- Examples of linear expressions: `X+Y`, `2*X`, `X-3*Z`

Syntax of PUBS declarations

$$C(\overline{x}) = \texttt{exp} + \sum_{i=1}^{n} C(\overline{y_i}) \qquad \varphi$$

```
eq(Head, BasicExp, [RecCalls], [GuardConditions]).
```

- Each element of `RecCalls` has the same syntax as `Head`.
- `t(2*X)` can be translated into `t(Y)`, if we add the constraint `Y=2*X`.
- In some cases, inline expressions are allowed in the `RecCall` (just try!)

Syntax of PUBS declarations

$$C(\overline{x}) = \texttt{exp} + \sum_{i=1}^{n} C(\overline{y_i}) \qquad \varphi$$

```
eq(Head, BasicExp, [RecCalls], [GuardConditions]).
```

- `GuardConditions` is a comma-separated list of linear conditions:

```
LinearExpression >= LinearExpression
LinearExpression = LinearExpression
```

- `X>Y` can be translated into `X>=Y+1`
- $\frac{1}{2}X = Y$ can be translated into `X=2*Y`

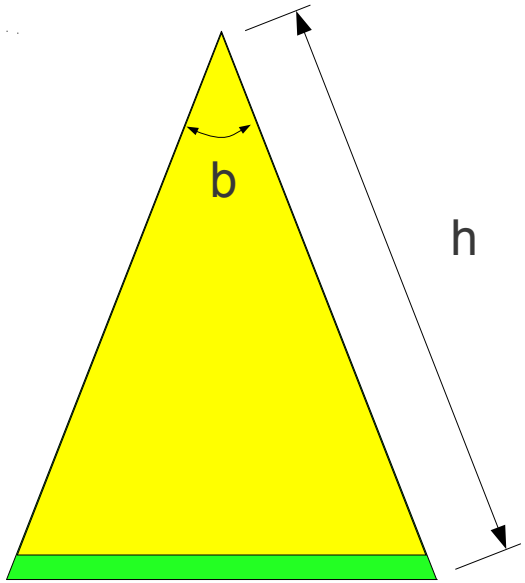
PUBS Example (I)

Example

$$\begin{array}{lcll} T(x,y) & = & 0 & \{x \geq y\} \\ T(x,y) & = & x^2 + T(x+1,y) & \{x < y\} \end{array}$$

```
eq(t(X,Y),0,[],[X>=Y]).
eq(t(X,Y),nat(X)*nat(X),[t(X+1,Y)],[Y>=X+1]).
```

Computing a closed form



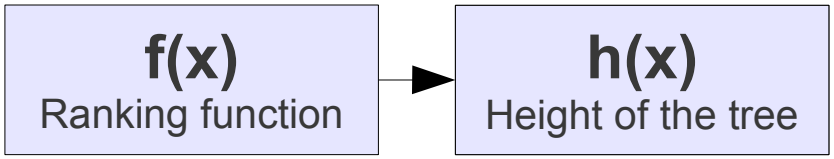
$$C^+(\overline{x}) = nr(\overline{x}) * cr(\overline{x}) + nb(\overline{x}) * cb(\overline{x})$$

Computing a closed form

f(x)
Ranking function

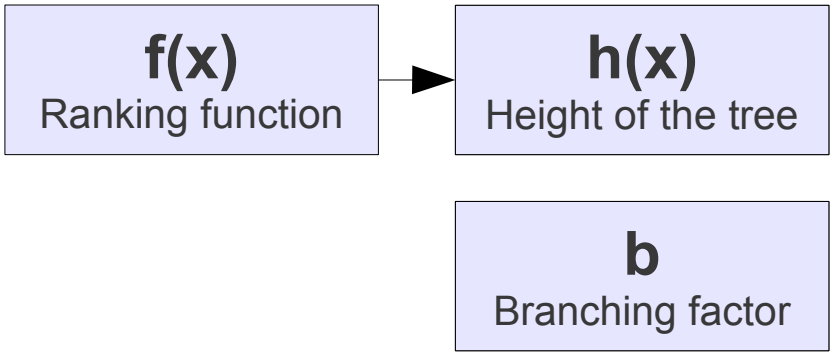
- Compute a ranking function $f(\overline{x})$
  - Podelsky and Rybalchenko, 2004. (linear!)

Computing a closed form



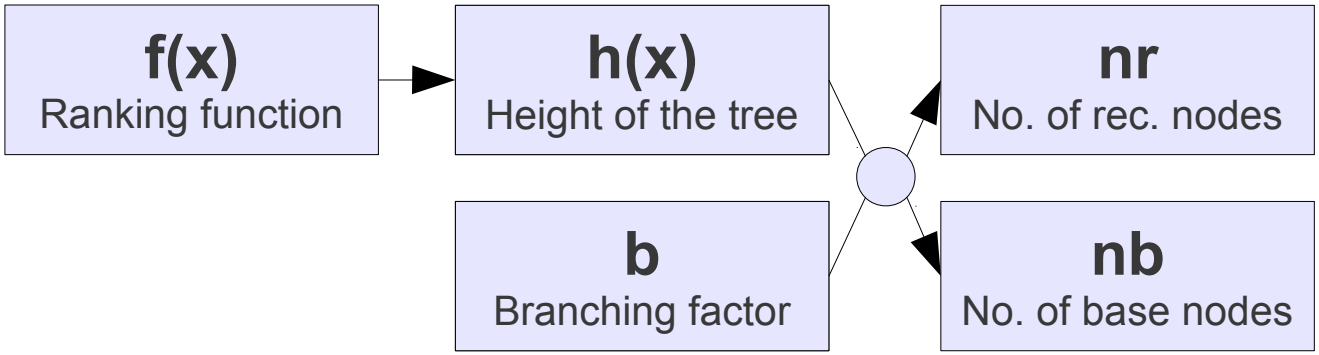
- Compute an upper-bound to the height of the call tree:
  - In the general case: $h(\overline{x}) = f(\overline{x})$
  - Sometimes this bound can be further refined: $\frac{f(\overline{x})}{\delta}$, $\log_{\delta}(\overline{x})$

Computing a closed form



- Determine the branching factor b

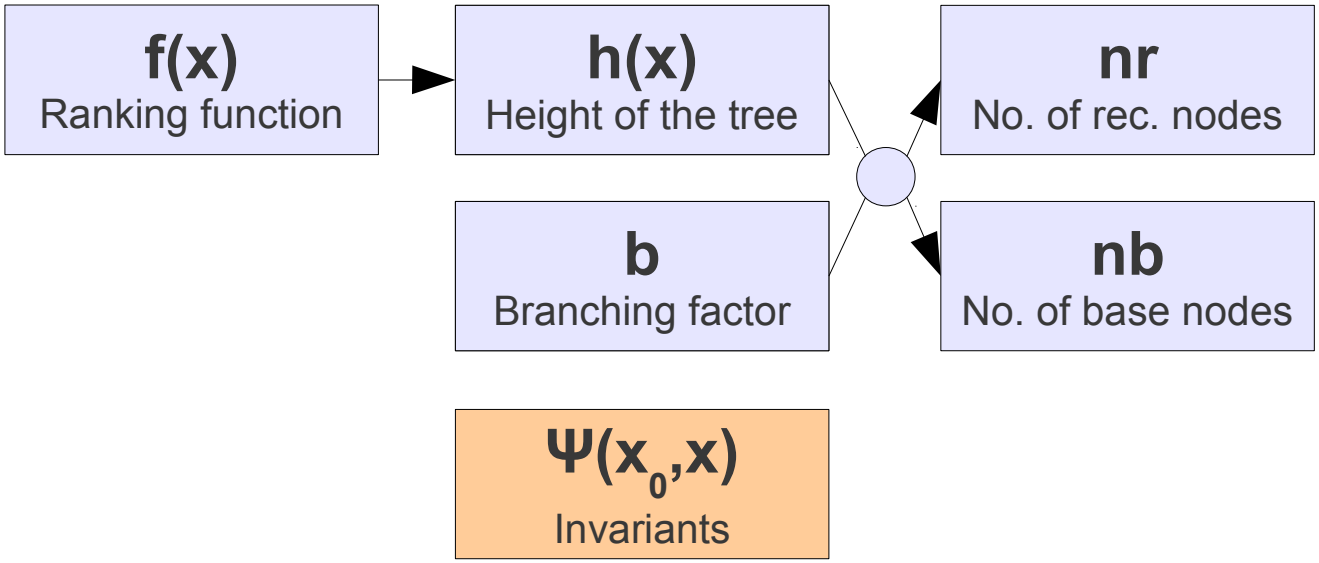
Computing a closed form



- Compute an upper-bound to the number of base and recursive nodes $nr(\overline{x})$, $nb(\overline{x})$

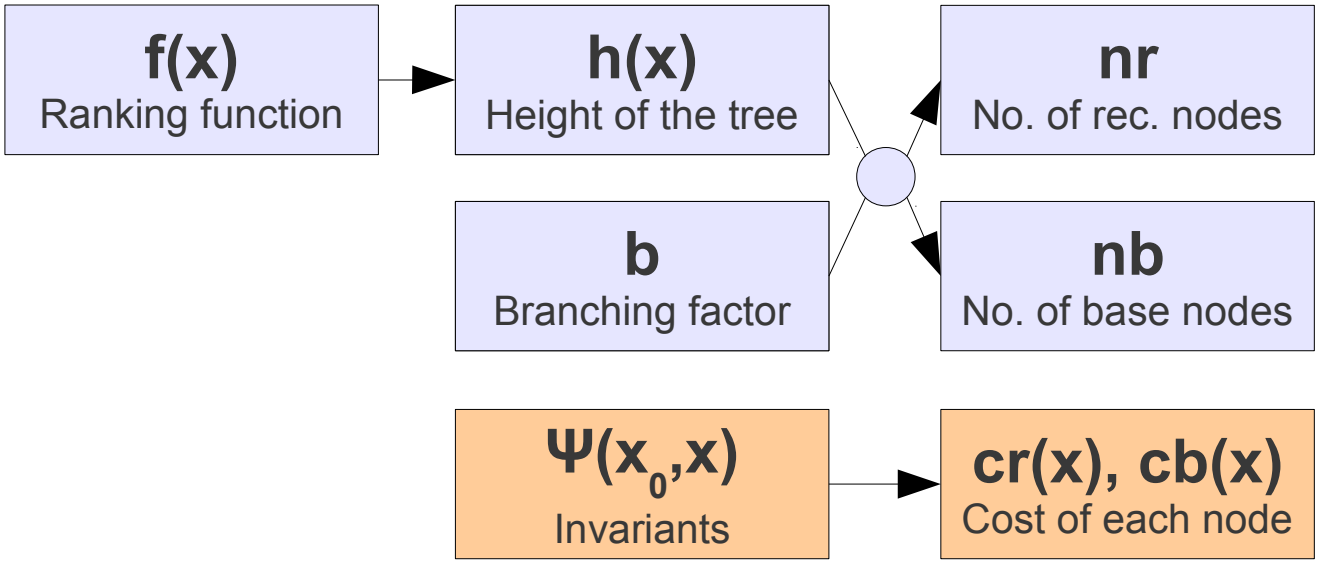
$$nb(\overline{x}) = b^{h(\overline{x})} \qquad nr(\overline{x}) = \begin{cases} h(\overline{x}) & \text{if } b = 1 \\ \frac{b^{h(\overline{x})}-1}{b-1} & \text{if } b \geq 2 \end{cases}$$

Computing a closed form



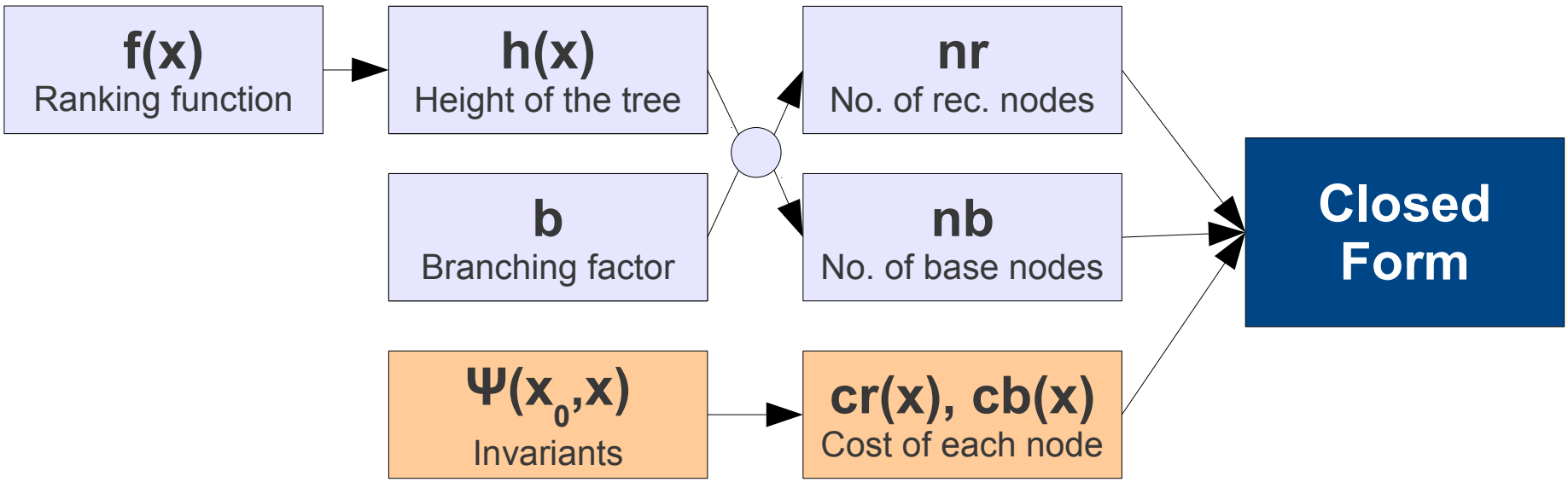
- Compute the relation Ψ between the root call and every child call.

Computing a closed form



- Maximize each $nat(l)$ component, by using the corresponding invariants.

Computing a closed form



$$C^+(\overline{x}) = nr(\overline{x}) * cr(\overline{x}) + nb(\overline{x}) * cb(\overline{x})$$

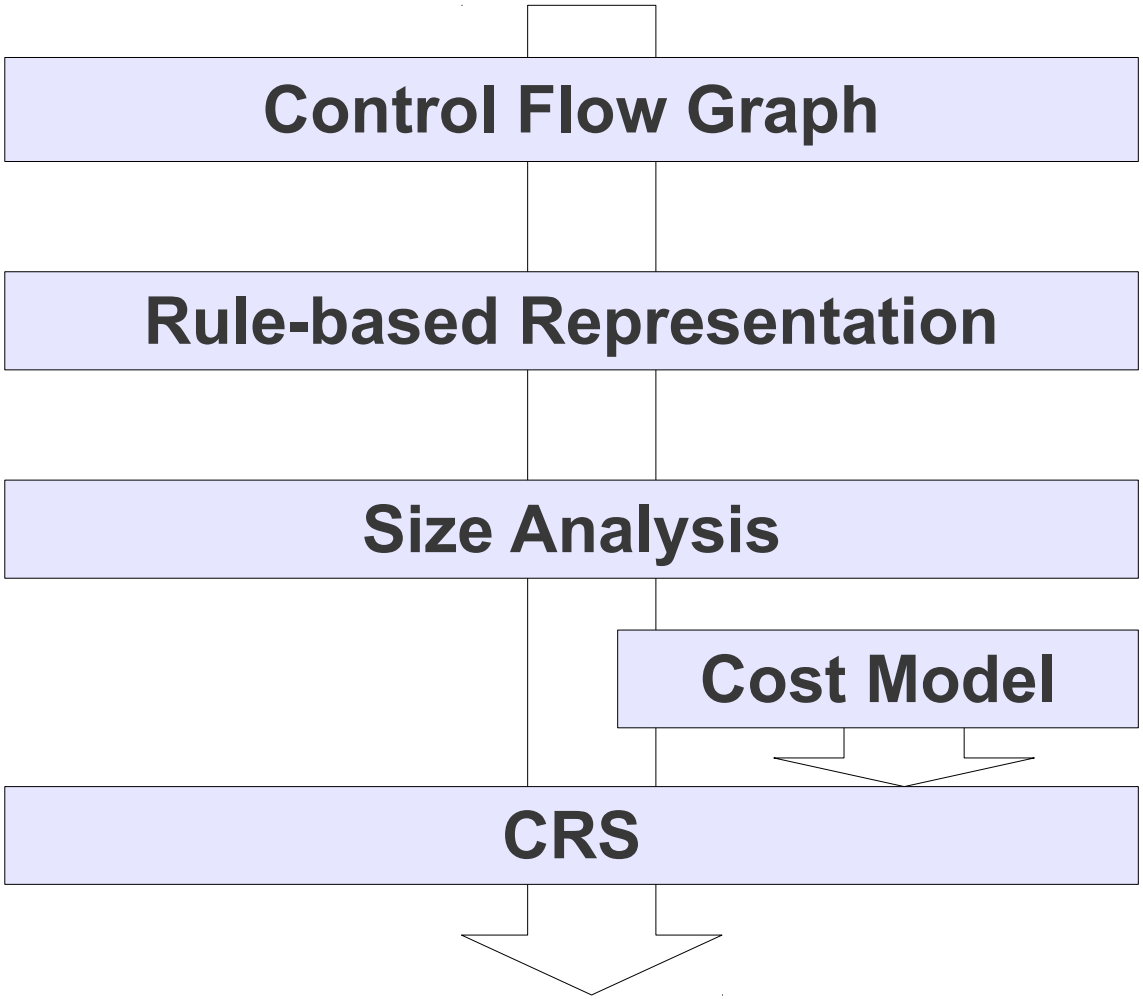
PUBS Example (II)

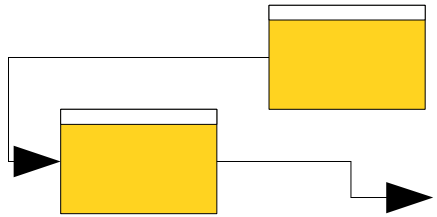
Example

$$\begin{array}{lcll} T(x) & = & 0 & \{x \leq 0\} \\ T(x) & = & 1 + T(x') + T(x'') & \{x > 0, x > 2 * x', x > 2 * x''\} \end{array}$$

```
eq(t(X),0,[],[0>=X]).
eq(t(X),1,[t(X1),t(X2)],[X>=1,X>=2*X1+1,X>=2*X2]).
```

COSTA Overview





$p(x) \leftarrow ...q(y)...$

$y = 2*x$

COSTA Example (I)

```
public void method1(int n, int m) {
  int[] array;
  for (int i = 0; i < n; i++) {
    array = new int[m];
  }
}
```

COSTA Example (II)

```
public static final int ARRAY_SIZE = 20;

public void method2(int n) {
  int[] array;
  for (int i = 0; i < n; i++) {
    array = new int[ARRAY_SIZE];
  }
}
```

COSTA Example (III)

```
public void method3(int n, int m) {
   int numIterations = n * m;
   for (int i = 0; i < numIterations; i++) {
      System.out.println("Iteration");
   }
}
```

COSTA Example (III)

```
public void method3(int n, int m) {
   int numIterations = n * m;
   for (int i = 0; i < numIterations; i++) {
      System.out.println("Iteration");
   }
}
```

- The size of `numIterations` does not depend linearly on `n` and `m`.

Web interface

- The COSTA System:
  `https://costa.ls.fi.upm.es/`
- PUBS:
  `http://costa.ls.fi.upm.es/~costa/pubs/pubs.php`
- COSTA:
  `http://costa.ls.fi.upm.es/~costa/costa/costa.php`

E. Albert, P. Arenas, S. Genaim, M. Gómez-Zamalloa, G. Puebla, D.Ramírez, G. Román, D.Zanardini
Termination and Cost Analysis with COSTA and its User Interfaces.
Spanish Conference on Computer Languages and Programming (PROLE 2009), ENTCS, Elsevier.