Exercises on COSTA and PUBS

Reliability of Software Systems

November 15, 2010

COSTA web interface: `http://costa.ls.fi.upm.es/~costa/costa/costa.php`
PUBS web interface: `http://costa.ls.fi.upm.es/~costa/pubs/pubs.php`

1 Resolution of CRSs

Obtain a closed form for the following cost relation systems by using PUBS:

(a)
$$\begin{array}{lcll} T_1(n,m) & = & 0 & \{n \leq 0\} \\ T_1(n,m) & = & m + T_1(n-1,m) & \{n > 0\} \end{array}$$

(b)
$$\begin{array}{lcll} T_2(n) & = & 0 & \{n \leq 0\} \\ T_2(n) & = & 1 + T_2(n') & \{n > 0, n' < \frac{n}{2}\} \end{array}$$

(c)
$$\begin{array}{lcll} T_3(n) & = & 1 & \{n \leq 0\} \\ T_3(n) & = & 1 + 3 * T_3(n') & \{n > 0, n' < \frac{n}{3}\} \end{array}$$

(d)
$$\begin{array}{lcll} T_4(n,m) & = & 1 & \{n \leq 0\} \\ T_4(n,m) & = & m^2 + 2 * T_4(n-1, m-1) & \{n > 0\} \end{array}$$

In (b) and (d), explain how PUBS infers, from the ranking function, the number of recursive nodes of the evaluation tree.

2 Linked lists

The file `List.java` contains an implementation of several methods that work on linked lists. By using COSTA, compute for each method below: (1) an upper bound to the number of instructions and (2) an upper bound to the memory consumption (without garbage collection).

(a). `public int length()`
It returns the number of elements of the list.

(b). `public void appendTo(List other)`
It connects the `this` list to the list given as parameter.

(c). `public List merge(List other)`
Assuming that the `this` list and the list given as parameter are sorted, it merges both in order to get another sorted list.

(d). `public List reverse()`
It returns a list with the same elements as `this`, but in reverse order.

Interpret the results given by COSTA as a function of the size of the `this` object and of the given parameters.

3 Binary search trees

For each method below, compute an upper-bound to the number of executed instructions. All these methods can be found in the `Tree.java` file.

(a). `public int getSize()`
Access to the number of nodes of the tree.

(b). `public void insert(int newValue)`
Insertion in a binary search tree.

(c). `public List inorderGen(List l)`
Inorder traversal of a tree: elements are successively added to the beginning of the list given as parameter.

(d). `public List inorder()`
Inorder traversal of a tree, equivalent to `inorderGen(null).reverse()`.

Why does COSTA fail on the last method? *Hint:* Trying to apply the memory consumption analysis to `inorderGen` may give you an idea of the size of its result. Is this size linear on the input's size?

4 Nonlinear ranking functions

PUBS uses the Podelski and Rybalchenko's approach for computing ranking functions. This method is restricted to infer *linear* ranking functions.

(a). Find a CRS whose evaluation results in a tree with an height not depending linearly on the input sizes.

(b). Write a Java method whose analysis (number of instructions) leads to a CRS similar to (a). That is, a method whose number of steps does not depend linearly on the input sizes.

Try both results in PUBS and COSTA, respectively. *Hint:* Consider two nested `for` loops, in which their respective variables range from 1 to n, being n a parameter. Try to do the same program with a single `while` loop, and you will get (b).

5 Mergesort

Consider the mergesort implementation in `Mergesort.java`. The actual cost of this function (number of instructions) belongs to $\mathcal{O}(n \log n)$, being n the length of the input array.

(a). Analyse the method `mergesort` in COSTA (number of instructions):

`void mergesort(int[] values, int begin, int end)`

If it takes too long to compute, you can disable the following option, which can be reached through the *Manual* Interface: `Consider implicit exceptions (thrown by the Virtual Machine)`.

Assume that `begin` takes the value 0 and `end` takes the value `values.length`. You do not have to give the whole cost expression, but just its complexity order (linear, quadratic, logarithmic, etc.)

(b). *(Optional)* The time complexity of a typical mergesort implementation can be represented by this CRS:

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

Why does PUBS return a quadratic upper bound, instead of returning a function in $\mathcal{O}(n \log n)$?