A General Framework for Static Profiling of Parametric Resource Usage [*]

published in Theory and Practice of Logic Programming

P. LOPEZ-GARCIA[1,2] M. KLEMEN[1] U. LIQAT[1] M.V. HERMENEGILDO[1,3]

[1] *IMDEA Software Institute*
(*e-mail:* {pedro.lopez,maximiliano.klemen,umer.liqat,manuel.hermenegildo}@imdea.org)
[2] *Spanish Council for Scientific Research (CSIC)*
[3] *Technical University of Madrid (UPM)*

submitted April 30, 2016; revised July 10, 2016; accepted July 22, 2016

Appendix A
Additional Examples

Example 1
Consider the following program to determine whether a list is a sublist of another. A sublist can be specified in terms of prefixes and suffixes: a suffix of a prefix, or a prefix of a suffix. The following program uses the latter to implement the `sublist` predicate.

```
sublist(L, L).
sublist(Sub, List) :- suffix(List, Suf), prefix(Suf, Sub).

suffix(List, Suffix):- append(_, Suffix, List).

prefix(List, Prefix):- append(Prefix, _, List).

append([], L, L).
append([X|Xs], L, [X|Zs]):- append(Xs, L, Zs).
```

Assume we are going to perform the analysis, to infer upper bounds on both the standard and accumulated costs, in terms of resolution steps, for the calling pattern `sublist(list, list)`, i.e., for the case where `sublist` is called with both of its arguments bound to lists. Given such calling pattern, CiaoPP infers the unique calling patterns `suffix(list, var)` and `prefix(var, list)`, for `suffix` and `prefix`, where `var` represents an unbound variable. However, two different calling patterns are inferred for `append`: `append(var,var,list)`, when it is called from `suffix`, and `append(list,var,list)`, when it is called from `prefix`.

Assume that size relations have been inferred for the different arguments in a clause, and that the size metric used is the *list length* of an argument, since all arguments are lists. The size of the output (second) argument of `suffix` is inferred as a function on

* This research has received funding from the EU FP7 agreement no 318337, *ENTRA*, Spanish MINECO TIN2012-39391 *StrongSoft* and TIN2015-67522-C3-1-R *TRACES* projects, and the Madrid M141047003 *N-GREENS* program. Special thanks are due to John Gallagher for many fruitful and inspiring discussions and to the anonymous reviewers for their careful and useful comments.

its input (first) argument, and is represented by $Sz^2_{\mathtt{suffix}}(n)$. Such inference sets up the following size relations:

$$\begin{array}{rll} Sz^2_{\mathtt{suffix}}(n) = & Sz^2_{\mathsf{append}}(n) & \\ Sz^2_{\mathsf{append}}(n) = & 1 & \text{if } n = 1 \\ Sz^2_{\mathsf{append}}(n) = & n & \text{if } n > 1 \\ Sz^2_{\mathsf{append}}(n) = & Sz^2_{\mathsf{append}}(n-1) & \text{if } n > 1 \end{array}$$

and finds the closed form $Sz^2_{\mathtt{suffix}}(n) = n$.

In order to infer the standard cost of this program, the analysis sets up the following cost relations for `sublist`, `suffix`, `prefix` and `append`:

$$\begin{array}{rl} \mathsf{C}_{\mathtt{sublist}}(n,m) = & \mathsf{C}_{\mathtt{suffix}}(n) + Sol_{\mathtt{suffix}}(n) * \mathsf{C}_{\mathtt{prefix}}(Sz^2_{\mathtt{suffix}}(n), m) + 2 \\ \mathsf{C}_{\mathtt{suffix}}(n) = & \mathsf{C}_{\mathsf{append}}(n) + 1 \\ \mathsf{C}_{\mathtt{prefix}}(n,m) = & \mathsf{C}_{\mathsf{append}}(n,m) + 1 \end{array}$$

Note that the size of the input to the call to `prefix` is given by the size of the output of `suffix`, represented by $Sz^2_{\mathtt{suffix}}(n)$.

The cost relations for the two variants of `append` are:

$$\begin{array}{rll} \mathsf{C}_{\mathsf{append}}(n) = & 1 & \text{if } n = 1 \\ \mathsf{C}_{\mathsf{append}}(n) = & \mathsf{C}_{\mathsf{append}}(n-1) + 2 & \text{if } n > 1 \end{array}$$

$$\begin{array}{rll} \mathsf{C}_{\mathsf{append}}(n,m) = & 1 & \text{if } m = 1, n = 1 \\ \mathsf{C}_{\mathsf{append}}(n,m) = & 1 & \text{if } m = 1 \\ \mathsf{C}_{\mathsf{append}}(n,m) = & \mathsf{C}_{\mathsf{append}}(n-1, m-1) + 1 & \text{if } m > 1 \end{array}$$

and the their closed forms are $\mathsf{C}_{\mathsf{append}}(n) = 2\ n - 1$ and $\mathsf{C}_{\mathsf{append}}(n,m) = m$ respectively.

Note that in this program `suffix` produces multiple solutions. For each solution of `suffix` the `prefix` predicate is executed on backtracking.

The cost relations for the inference of the number of solutions are:

$$\begin{array}{rll} Sol_{\mathtt{suffix}}(n) = & Sol_{\mathsf{append}}(n) & \\ Sol_{\mathsf{append}}(n) = & 1 & \text{if } n = 1 \\ Sol_{\mathsf{append}}(n) = & Sol_{\mathsf{append}}(n-1) + 1 & \text{if } n > 1 \end{array}$$

and the closed form is $Sol_{\mathtt{suffix}}(n) = n$.

After composing all the closed forms, the analysis obtains the following function representing an upper bound on the resource usage of all the predicates:

$$\begin{array}{rl} \mathsf{C}_{\mathtt{suffix}}(n) = & 2\ n \\ \mathsf{C}_{\mathtt{prefix}}(n) = & m + 1 \\ \mathsf{C}_{\mathsf{append}}(n) = & 2\ n - 1 \\ \mathsf{C}_{\mathsf{append}}(n,m) = & m \\ \mathsf{C}_{\mathtt{sublist}}(n) = & n\ m + 3\ n + 2 \end{array}$$

Assume now that we declare `sublist` and `append` as cost centers to infer the accumulated costs in them. The cost relations set up for `sublist` are:

$$\begin{array}{rl} \mathsf{C}^{\mathtt{sublist}}_{\mathtt{sublist}}(n,m) = & \mathsf{C}^{\mathtt{sublist}}_{\mathtt{suffix},1}(n) + Sol_{\mathtt{suffix}}(n) * \mathsf{C}^{\mathtt{sublist}}_{\mathtt{prefix},1}(Sz^2_{\mathtt{suffix}}(n), m) + 2 \\ \mathsf{C}^{\mathsf{append}}_{\mathtt{sublist}}(n,m) = & \mathsf{C}^{\mathsf{append}}_{\mathtt{suffix},0}(n) + Sol_{\mathtt{suffix}}(n) * \mathsf{C}^{\mathsf{append}}_{\mathtt{prefix},0}(Sz^2_{\mathtt{suffix}}(n), m) \end{array}$$

Replacing the functions for sizes (Sz) and solutions (Sol) from the previous step we get:

$$\begin{aligned}
\mathtt{C}^{\mathtt{sublist}}_{\mathtt{sublist}}(n,m) = &\quad \mathtt{C}^{\mathtt{sublist}}_{\mathtt{suffix},1}(n) + n * \mathtt{C}^{\mathtt{sublist}}_{\mathtt{prefix},1}(n,m) + 2\\
\mathtt{C}^{\mathtt{append}}_{\mathtt{sublist}}(n,m) = &\quad \mathtt{C}^{\mathtt{append}}_{\mathtt{suffix},0}(n) + n * \mathtt{C}^{\mathtt{append}}_{\mathtt{prefix},0}(n,m)
\end{aligned}$$

Furthermore, the intermediate cost relations are set up as follows:

$$\begin{aligned}
\mathtt{C}^{\mathtt{sublist}}_{\mathtt{suffix},1}(n) = &\quad \mathtt{C}^{\mathtt{sublist}}_{\mathtt{append}}(n) + 1\\
= &\quad 1 && \text{(Lemma 3)}\\
\mathtt{C}^{\mathtt{sublist}}_{\mathtt{prefix},1}(n,m) = &\quad \mathtt{C}^{\mathtt{sublist}}_{\mathtt{append}}(n,m) + 1\\
= &\quad 1 && \text{(Lemma 3)}\\
\mathtt{C}^{\mathtt{append}}_{\mathtt{suffix},0}(n) = &\quad \mathtt{C}^{\mathtt{append}}_{\mathtt{append}}(n)\\
\mathtt{C}^{\mathtt{append}}_{\mathtt{prefix},0}(n,m) = &\quad \mathtt{C}^{\mathtt{append}}_{\mathtt{append}}(n,m)
\end{aligned}$$

$$\begin{aligned}
\mathtt{C}^{\mathtt{append}}_{\mathtt{append}}(n) = &\quad \mathtt{C}^{\mathtt{append}}_{\mathtt{append}}(n-1) + 2 && \text{if } n > 1\\
\mathtt{C}^{\mathtt{append}}_{\mathtt{append}}(n) = &\quad 1 && \text{if } n = 1
\end{aligned}$$

$$\begin{aligned}
\mathtt{C}^{\mathtt{append}}_{\mathtt{append}}(n,m) = &\quad 1 && \text{if } m = 1, n = 1\\
\mathtt{C}^{\mathtt{append}}_{\mathtt{append}}(n,m) = &\quad 1 && \text{if } m = 1\\
\mathtt{C}^{\mathtt{append}}_{\mathtt{append}}(n,m) = &\quad \mathtt{C}^{\mathtt{append}}_{\mathtt{append}}(n-1) + 1 && \text{if } m > 1
\end{aligned}$$

After composing all the intermediate cost relations, the analysis obtains the following functions representing upper bounds on the accumulated resource usage of `sublist` and `append`:

$$\begin{aligned}
\mathtt{C}^{\mathtt{sublist}}_{\mathtt{sublist}}(n) = &\quad n + 3\\
\mathtt{C}^{\mathtt{append}}_{\mathtt{sublist}}(n) = &\quad nm + 2n - 1
\end{aligned}$$

Example 2

Consider the following program $\mathcal{P}$:

```
p(X, Y) :- i1, i2, q(X, Z),  s(Z, Y).

q(0, 0).
q(X, Y) :- X1 is X - 1, q(X1, Z), i1, i3, s(Z, W), Y is W + 1.

s(0, 0):- i1.
s(X, Y) :- i2, i4, X1 is X - 1, s(X1, Z), Y is Z + 1.
```

Assume that `im`, for $\mathtt{m} \in \{1,2,3,4\}$, are builtin/library predicates and that their standard costs are given by means of trust assertions: for simplicity we assume that $\Psi(\mathtt{im}) = \mathtt{C}_{\mathtt{im}} = 1$, for $\mathtt{m} \in \{1,2,3,4\}$, and that the (standard) cost of the $is/2$ arithmetic predicate is given as zero. Assume also that $\varphi(\mathtt{p}) = 0$ for all predicates $\mathtt{p} \in \mathcal{P}$.

Assume that for all the predicates, the first argument is an input argument and the second one is output, and that the type of all arguments is the set of natural numbers. Assume that the following size relationships, expressing the size of the output argument as a function of the size of the input argument, have already been inferred for all of them:

- $Sz^2_{\mathtt{p}}(n) = n$, which means that the size (under the natural value metric) of the second argument of predicate `p` is n, the size of the input argument.
- Similarly, $Sz^2_{\mathtt{q}}(n) = n$, and $Sz^2_{\mathtt{s}}(n) = n$.

Assume that the set of cost centers is $\lozenge = \{\mathtt{p}, \mathtt{q}, \mathtt{s}\}$, and that we want to estimate (upper bounds on) the cost accumulated in all the cost centers for the predicates p, q, and s. Let $\mathtt{C}^{\mathtt{q}}_{\mathtt{p}}(\bar{\mathtt{x}})$ denote an upper bound on the accumulated cost in cost center q corresponding to a call $\mathtt{p}(\bar{\mathtt{x}})$.

Assume we process each strongly-connected component of the call graph of the program in reverse topological order. We start by inferring the costs accumulated in cost center s. The accumulated cost in s corresponding to a call to s is expressed by the following cost relation:

$$\begin{array}{rcl} \mathtt{C}^{\mathtt{s}}_{\mathtt{s}}(0) & = & \mathtt{C}^{\mathtt{s}}_{\mathtt{i1},1} = \mathtt{C}_{\mathtt{i1}} = 1 \\ \mathtt{C}^{\mathtt{s}}_{\mathtt{s}}(n) & = & \mathtt{C}^{\mathtt{s}}_{\mathtt{i2},1} + \mathtt{C}^{\mathtt{s}}_{\mathtt{i4},1} + \mathtt{C}^{\mathtt{s}}_{\mathtt{s}}(n-1) \end{array}$$

which can be written as:

$$\begin{array}{rcl} \mathtt{C}^{\mathtt{s}}_{\mathtt{s}}(0) & = & 1 \\ \mathtt{C}^{\mathtt{s}}_{\mathtt{s}}(n) & = & 2 + \mathtt{C}^{\mathtt{s}}_{\mathtt{s}}(n-1) \end{array}$$

and whose closed-form solution is:

$$\mathtt{C}^{\mathtt{s}}_{\mathtt{s}}(n) \quad = \quad 2\ n + 1.$$

Now, we analyze predicate q. To this end, the accumulated cost in s for a call to q is expressed by:

$$\begin{array}{rcl} \mathtt{C}^{\mathtt{s}}_{\mathtt{q}}(0) & = & 0 \\ \mathtt{C}^{\mathtt{s}}_{\mathtt{q}}(n) & = & \mathtt{C}^{\mathtt{s}}_{\mathtt{q}}(n-1) + \mathtt{C}^{\mathtt{s}}_{\mathtt{i1},0} + \mathtt{C}^{\mathtt{s}}_{\mathtt{i3},0} + \mathtt{C}^{\mathtt{s}}_{\mathtt{s}}(n-1) \end{array}$$

Since there is a trust assertion providing the cost of i1, $\Psi(\mathtt{i1}) = 1$ (as already said), according to Expression 5, we have that $\mathtt{C}^{\mathtt{s}}_{\mathtt{i1},0} = B_{\varphi}(\mathtt{i1}, \mathtt{s}, 0) \times \Psi(\mathtt{i1}) = 0 \times 1 = 0$. Note that $B_{\varphi}(\mathtt{i1}, \mathtt{s}, 0) = 0$ because $\mathtt{i1} \neq \mathtt{s}$ and the environment (third argument of B_{φ}) is 0 (since i1 is called in the scope of cost center q, not in the scope of the cost center where the analysis is accumulating costs in this equation, i.e., s). Thus, the cost of i1 is not taken into account in this equation. The same consideration applies to i3.

Since the cost function for $\mathtt{C}^{\mathtt{s}}_{\mathtt{s}}(n)$ has already been computed, replacing values we have

$$\begin{array}{rcl} \mathtt{C}^{\mathtt{s}}_{\mathtt{q}}(0) & = & 0 \\ \mathtt{C}^{\mathtt{s}}_{\mathtt{q}}(n) & = & \mathtt{C}^{\mathtt{s}}_{\mathtt{q}}(n-1) + 2\ (n-1) + 1 \end{array}$$

and

$$\begin{array}{rcl} \mathtt{C}^{\mathtt{s}}_{\mathtt{q}}(0) & = & 0 \\ \mathtt{C}^{\mathtt{s}}_{\mathtt{q}}(n) & = & \mathtt{C}^{\mathtt{s}}_{\mathtt{q}}(n-1) + 2\ n - 1 \end{array}$$

The solution to the cost relation above is:

$$\mathtt{C}^{\mathtt{s}}_{\mathtt{q}}(n) \quad = \quad n^2.$$

We now analyze predicate p, so that the accumulated cost in s for a call to p is expressed by:

$$\mathtt{C}^{\mathtt{s}}_{\mathtt{p}}(n) \quad = \quad \mathtt{C}^{\mathtt{s}}_{\mathtt{i1},0} + \mathtt{C}^{\mathtt{s}}_{\mathtt{i2},0} + \mathtt{C}^{\mathtt{s}}_{\mathtt{q}}(n) + \mathtt{C}^{\mathtt{s}}_{\mathtt{s}}(n)$$

For the same considerations as before, the costs of i1 and i2 in the body of the clause defining p are not taken into account (i.e., $\mathtt{C}^{\mathtt{s}}_{\mathtt{i1},0} = \mathtt{C}^{\mathtt{s}}_{\mathtt{i2},0} = 0$). Replacing values we have that:

$$\mathtt{C}^{\mathtt{s}}_{\mathtt{p}}(n) \quad = \quad n^2 + 2n + 1.$$

The inference of the accumulated cost in `s` for predicates `p`, `q`, and `s` has finished, and we start now the inference of the accumulated costs in `q`. By Lemma 3, $\mathtt{C}^{\mathtt{q}}_{\mathtt{s}}(n) = 0$, i.e., we do not need to analyze predicate `s`, since it does not call `q`. However, now the costs of `i1` and `i3` in the body of the second clause of `q` do have to be taken into account. To this end, the recurrence equations expressing the accumulated cost in `q` for a call to `q` are:

$$\begin{array}{rcl} \mathtt{C}^{\mathtt{q}}_{\mathtt{q}}(0) & = & 0 \\ \mathtt{C}^{\mathtt{q}}_{\mathtt{q}}(n) & = & \mathtt{C}^{\mathtt{q}}_{\mathtt{q}}(n-1) + \mathtt{C}^{\mathtt{i1}}_{\mathtt{q},1} + \mathtt{C}^{\mathtt{i3}}_{\mathtt{q},1}. \end{array}$$

$$\begin{array}{rcl} \mathtt{C}^{\mathtt{q}}_{\mathtt{q}}(0) & = & 0 \\ \mathtt{C}^{\mathtt{q}}_{\mathtt{q}}(n) & = & \mathtt{C}^{\mathtt{q}}_{\mathtt{q}}(n-1) + 2. \end{array}$$

The solution to the recurrence above is:

$$\mathtt{C}^{\mathtt{q}}_{\mathtt{q}}(n) \quad = \quad 2\ n.$$

Now, the accumulated cost in `q` for a call to `p` is expressed as:

$$\mathtt{C}^{\mathtt{q}}_{\mathtt{p}}(n) \quad = \quad \mathtt{C}^{\mathtt{q}}_{\mathtt{q}}(n) + \mathtt{C}^{\mathtt{q}}_{\mathtt{s}}(n)$$

Replacing values we have that:

$$\mathtt{C}^{\mathtt{q}}_{\mathtt{p}}(n) \quad = \quad 2\ n$$

Let us compute now the accumulated cost in `p`. Since it is not called from `q` nor `s`, we have that $\mathtt{C}^{\mathtt{p}}_{\mathtt{q}}(n) = \mathtt{C}^{\mathtt{p}}_{\mathtt{s}}(n) = 0$. The accumulated cost in `p` for a call to `p` is just the cost of `i1` and `i2`:

$$\mathtt{C}^{\mathtt{p}}_{\mathtt{p}}(n) \quad = \quad \mathtt{C}^{\mathtt{p}}_{\mathtt{i1},1} + \mathtt{C}^{\mathtt{p}}_{\mathtt{i2},1} = \mathtt{C}_{\mathtt{i1}} + \mathtt{C}_{\mathtt{i2}}.$$

Thus:

$$\mathtt{C}^{\mathtt{p}}_{\mathtt{p}}(n) \quad = \quad 2.$$

Note that the *standard cost* of `p` ($\mathtt{C}_{\mathtt{p}}(n)$) can be expressed in terms of the accumulated costs in each of the cost centers:

$$\mathtt{C}_{\mathtt{p}}(n) \quad = \quad \mathtt{C}^{\mathtt{p}}_{\mathtt{p}}(n) + \mathtt{C}^{\mathtt{q}}_{\mathtt{p}}(n) + \mathtt{C}^{\mathtt{s}}_{\mathtt{p}}(n).$$

Example 3
Consider the following program where the predicates p and q are mutually recursive.

```
coupled(X, Y):- f(X, Y).

p(0,[]).
p(N,[a|R]) :- N1 is N-1, q(N1,R).

q(0,[]).
q(N,[a|R]) :-N1 is N-1, p(N1,R).
```

Assuming that we want to infer the standard cost of this program in terms of resolution steps, the analysis sets up the following cost relations for *coupled*, p and q, we have the following cost relations:

$$\begin{array}{lll} \mathtt{C}_{p}(n) = & 1 & \text{if } n = 0 \\ \mathtt{C}_{p}(n) = & 1 + \mathtt{C}_{q}(n-1) & \text{if } n > 0 \end{array}$$

$$\begin{array}{lll} \mathtt{C}_{q}(n) = & 1 & \text{if } n = 0 \\ \mathtt{C}_{q}(n) = & 1 + \mathtt{C}_{p}(n-1) & \text{if } n > 1 \end{array}$$

$$\mathtt{C}_{coupled}(n) \quad = \quad 1 + \mathtt{C}_p(n)$$

After composing the closed forms, the analysis obtains the following function representing an upper bound on the resource usage of *coupled*, p and q:

$$\mathtt{C}_{coupled}(n) = \quad n + 2$$

$$\mathtt{C}_p(n) = \quad n + 1$$

$$\mathtt{C}_q(n) = \quad n + 1$$

Notice that in this program the cost relations for p and q are mutually recursive (i.e., they are defined in terms of each other), and for this reason the cost functions representing the upper bound on the resolution steps in the two are same $(n + 1)$. Hence, the cost of each mutually-recursive predicate subsumes the cost of the other. However, this cost is in fact distributed between the p and q predicates. In order to identify the cost that each of these predicates contributes to this $n+1$ expression and to the overall cost of *coupled* $(n + 2)$, we perform the accumulated cost analysis, declaring all the predicates as cost centers. The instantiation of the equational framework described in Sect. 4 obtains the following accumulated costs for *coupled*, p, and q:

$$\mathtt{C}^{coupled}_{coupled}(n) = \quad 1$$

$$\mathtt{C}^{p}_{coupled}(n) = \quad \frac{n}{2} + \frac{(-1)^n}{4} + \frac{3}{4}$$

$$\mathtt{C}^{q}_{coupled}(n) = \quad \frac{n}{2} - \frac{(-1)^n}{4} + \frac{1}{4}$$

It is now clear how much cost each of *coupled*, p, and q contributes to the standard cost of the whole program $(n + 2)$. Note that the standard cost of the mutually recursive predicates p and q, which is $n + 1$, is now halved among the two as accumulated costs of p and q.

In this example we have shown a hypothetical scenario highlighting that the accumulated cost information is more useful for mutually recursive parts of a program in order to identify how much each of the mutually recursive predicates contributes to the overall cost. This was not possible using only the standard cost information.

Example 4
Consider the following program to determine the parity of a number where predicates *even* and *odd* are mutually recursive.

```
even(0).
even(N):- N > 0, N1 is N - 1, odd(N1).

odd(1).
odd(N):- N > 1, N1 is N - 1, even(N1).
```

Similar to the Example 3, this program contains mutually recursive predicates *even* and *odd*. Since both are defined in terms of each other, the standard analysis obtains a same cost function for them representing an upper bound on the resource usage.

$$\mathtt{C}_{even}(n) = \quad n + 1$$

$$\mathtt{C}_{odd}(n) = \quad n + 1$$

In order to identify the cost that each of these predicates contributes to the overall cost of the program $n+1$, we perform the accumulated cost analysis, declaring both *even* and *odd* as as cost centers. The instantiation of the equational framework described in Sect. 4 obtains the following accumulated costs for *even* and *odd*:

$$\mathsf{C}^{even}_{even}(n) = \frac{n}{2} + \frac{(-1)^n}{4} + \frac{3}{4}$$

$$\mathsf{C}^{odd}_{even}(n) = \frac{n}{2} - \frac{(-1)^n}{4} + \frac{1}{4}$$

Appendix B
Additional Comments on the Relation of the Standard and Accumulated Cost

Assume that predicate `p` is a cost center. As already said, in this case the standard cost of a single call $\mathtt{p}(\bar{\mathtt{x}})$ is the sum of its accumulated costs in all the cost centers in the program. This is formalized by Theorem 1 in (Haemmerlé et al. 2016), which holds under the assumption that `p` is a cost center. Intuitively, predicate `c` is "reachable" from predicate `p` if $\mathtt{c} = \mathtt{p}$ or `c` can be invoked (either directly or indirectly) by `p`. If `p` is a cost center, Theorem 1 also holds if we restrict to the set of cost centers that are reachable from `p`, or to the set of cost centers that are descendants (in the call stack) of `p`. The reason is that if `p` is a cost center, and another cost center `c` (different from `p`) is not reachable from `p`, then no part of the cost of a call to `p` is attributed to `c`. This is stated in Lemma 3.

Assume that `p` is the main (entry) predicate in a program, and that we are interested in knowing how its total (standard) cost is *distributed* over the (user-defined) cost centers. In this case, `p` should be declared as a cost center. This is because if `p` is a cost center, the residual cost (as defined in Section 4) of the call to the main predicate will be assigned to `p`. Otherwise the residual cost will be left unassigned to any cost center.

If `p` is not a cost center, the standard cost of a single call $\mathtt{p}(\bar{\mathtt{x}})$ is the sum of its accumulated costs in all the cost centers that are descendants (in the call stack) of `p`, plus the residual cost of that call.

References

Haemmerlé, R., Lopez-Garcia, P., Liqat, U., Klemen, M., Gallagher, J. P., and Hermenegildo, M. V. 2016. A Transformational Approach to Parametric Accumulated-cost Static Profiling. In *FLOPS'16*. LNCS, vol. 9613. Springer, 163–180.