Notes on the Intensional Expressive Power of Bounded Calculi

Ugo Dal Lago

February 2, 2007

Until recently, implicit computational complexity has focused on extensional characterizations of complexity classes. These characterizations are interesting from a programming language perspective as well, but with the following *proviso*: extensional correspondence with complexity classes is not enough, we need to capture as many *algorithms* as possible (without losing the correspondence with complexity classes). There have been advances in this direction (for example: non-size-increasing, quasi-interpretations, etc.). It is not easy, however, to measure the improvement that these new systems achieve. What we need are *sharp results* on the intensional expressive power of existing systems.
In these notes, I will clarify my previous statement by a toy example: I introduce a language for primitive recursive definitions and characterize the class of bounded primitive recursive definitions (due to Cobham [1]) as the primitive recursive definitions which are hereditarily polytime. What is interesting is not this result itself but the question it implicitly raises: is it possible to sharply characterize the intensional expressive power of other, more interesting, systems (Bounded Linear Logic, Non-size increasing, Quasi-interpreations, etc.)?

1 Bounded Recursion on Notation

We can give the notion of a *primitive recursive definition (PRD) with a integer arity* as follows, by induction:

- The symbol *nil* is a PRD with arity 0.
- The symbols *cons0* and *cons1* are PRDs with arity 1.
- For every $n \geq 1$ and for every $1 \leq i \leq n$, the symbo π^n_i is a PRD with arity n.
- If s is a PRD with arity $n \geq 0$ and $t_1, \ldots, t_n$ are PRDs with arity $m \geq 0$, then $comp(s, t_1, \ldots, t_n)$ is a PRD with arity m.
- If s is PRD with arity $n \geq 0$ and t_0, t_1 are two PRDs with arity $n+2$, then $rec(s, t_0, t_1)$ is a PRD with arity $n+1$.

$\mathcal{D}$ is the class of all primitive recursive definitions. $\mathcal{B}$ is simply the language $\{0,1\}^*$ of binary, finite, strings. The semantics of a PRD s with arity n is a function $[\![s]\!] : \mathcal{B}^n \to \mathcal{B}$ defined as follows:

$$
\begin{aligned}
[\![nil]\!] &= \varepsilon \\
[\![cons0]\!](u) &= 0 \cdot u \\
[\![cons1]\!](u) &= 1 \cdot u \\
[\![\pi^n_i]\!](u_1, \ldots, u_n) &= u_i \\
[\![comp(s, t_1, \ldots, t_n)]\!](u_1, \ldots, u_m) &= [\![s]\!]([\![t_1]\!](u_1, \ldots, u_m), \ldots, [\![t_n]\!](u_1, \ldots, u_m)) \\
[\![rec(s, t_0, t_1)]\!](\varepsilon, u_1, \ldots, u_n) &= [\![s]\!](u_1, \ldots, u_1) \\
[\![rec(s, t_0, t_1)]\!](0 \cdot u_0, u_1, \ldots, u_n) &= [\![t_0]\!]([\![rec(s, t_0, t_1)]\!](u_0, \ldots, u_n), u_0, \ldots, u_n) \\
[\![rec(s, t_0, t_1)]\!](1 \cdot u_0, u_1, \ldots, u_n) &= [\![t_1]\!]([\![rec(s, t_0, t_1)]\!](u_0, \ldots, u_n), u_0, \ldots, u_n)
\end{aligned}
$$

The semantics $[\![s]\!]$ of a PRD s is a primitive recursive function in the classical sense. Conversely, every primitive recursive function can be expressed as a PRD. As a consequence, $[\![\mathcal{D}]\!] = \mathscr{R}$, where $\mathscr{R}$ is the class of primitive recursive functions.
We will now look at an operational semantics for PRDs. To do that, we need syntactic object expressing (partially evaluated) function calls to PRDs. A *primitive recursive expression (PRE)* is either an element of $\mathcal{B}$ or has the form $s(e_1, \dots, e_n)$, where s is a PRD with arity n and $e_1, \dots, e_n$ are PREs. $\mathcal{E}$ is the class of primitive recursive expressions. The relation $\longrightarrow$ on $\mathcal{E}$ is obtained from the following rules by closing them under any context:

$$
\begin{array}{rcl}
\mathit{nil} & \longrightarrow & \varepsilon \\
\mathit{cons0}(u) & \longrightarrow & 0 \cdot u \\
\mathit{cons1}(u) & \longrightarrow & 1 \cdot u \\
\pi_i^n(u_1, \dots, u_n) & \longrightarrow & u_i \\
\mathit{comp}(s, t_1, \dots, t_n)(u_1, \dots, u_m) & \longrightarrow & s(t_1(u_1, \dots, u_m), \dots, t_n(u_1, \dots, u_m)) \\
\mathit{rec}(s, t_0, t_1)(\varepsilon, u_1, \dots, u_n) & \longrightarrow & s(u_1, \dots, u_n) \\
\mathit{rec}(s, t_0, t_1)(0 \cdot u_0, u_1, \dots, u_n) & \longrightarrow & t_0(\mathit{rec}(s, t_0, t_1)(u_0, \dots, u_n), u_0, \dots, u_n) \\
\mathit{rec}(s, t_0, t_1)(1 \cdot u_0, u_1, \dots, u_n) & \longrightarrow & t_1(\mathit{rec}(s, t_0, t_1)(u_0, \dots, u_n), u_0, \dots, u_n)
\end{array}
$$

Clearly, $t(u_1, \dots, u_n) \longrightarrow^* u$ iff $[\![t]\!](u_1, \dots, u_n) = u$. A string $u \in \mathcal{B}$ *appears inside* the PRE e iff u is a subterm of e. Similarly, a PRD t appears inside e iff t is a subterm of e.

Proposition 1.1 *Let t be a PRD with arity n and let $u_1, \dots, u_n \in \mathcal{B}$. If $t(u_1, \dots, u_n) \longrightarrow^n e$ and u appears inside e, then $|u| \leq n + \max\{|u_1|, \dots, |u_n|\}$. Moreover, if a PRD s appears inside e, then s is a subterm of t.*

Proof We proceed by induction on the structure of t:

- If $t = \varepsilon$, then the hypothesis cannot be realized.
- If $t = 0$, then $t(u_1) \longrightarrow 0 \cdot u_1$ and the thesis is verified. Similarly when $t = 1$ and $t = \pi_i^n$.
- If $t = \mathit{comp}(s, t_1, \dots, t_n)$, then the reduction of $t(u_1, \dots, u_m)$ goes as follows:

$$
\begin{array}{rcl}
t(u_1, \dots, u_m) & \longrightarrow & s(t_1(u_1, \dots, u_m), \dots, t_n(u_1, \dots, u_m)) \\
& \longrightarrow^k & s(v_1, \dots, v_n)) \\
& \longrightarrow^h & v
\end{array}
$$

where $t_1(u_1, \dots, u_m) \longrightarrow^{k_i} v_i$ for every $1 \leq i \leq m$ and $k = \sum_{i=1}^m k_i$. The thesis is satisfied.

- If $t = \mathit{rec}(s, t_0, t_1)$ and $u_0, \dots, u_m \in \mathcal{B}$, it is convenient to go by induction on u_0 as follows:
  - If $u_0 = \varepsilon$, then the reduction of $t(u_0, \dots, u_m)$ goes as follows:

$$
t(u_0, \dots, u_n) \longrightarrow s(u_1, \dots, u_n) \longrightarrow^k v
$$

  The thesis follows.
  - If $u_0 = 0 \cdot u$, then the reduction of $t(u_0, \dots, u_m)$ goes as follows:

$$
\begin{array}{rcl}
t(u_0, \dots, u_n) & \longrightarrow & t_0(t(u, u_0 \dots, u_n), u, u_1 \dots, u_n) \\
& \longrightarrow^k & t_0(v, u, u_1, \dots, u_n) \\
& \longrightarrow^h & w
\end{array}
$$

  where $t(u, u_0 \dots, u_n) \longrightarrow^k v$. The thesis is satisfied.
  - If $u_0 = 1 \cdot u$, then we can proceed similarly.

This concludes the proof. □

As an easy corollary we get the following:

Corollary 1.1 *For every PRD t with arity n, there is a polynomial $p : \mathbb{N}^{n+1} \to \mathbb{N}$ such that whenever $t(u_1, \dots, u_n) \longrightarrow^m e$, it holds that $|e| \leq p(m, |u_1|, \dots, |u_n|)$.*

Proof Let k be a natural number greater than the size of t and let h be a natural number greater than the maximum arity of subterms of t. Then we define:

$$p(x, y_1, \ldots, y_n) = h(x+1)(k + x + y_1 + \ldots + y_n)$$

Clearly, p is a polynomial. Moreover, the size of $t(u_1, \ldots, u_n)$ is clearly majorized by

$$p(0, |u_1|, \ldots, |u_n|) = h(k + |u_1| + \ldots + |u_n|) \leq k + |u_1| + \ldots + |u_n|.$$

Finally, observe that, by proposition 1.1, if $t(u_1, \ldots, u_n) \longrightarrow^m e \longrightarrow f$, then

$$|f| \leq |e| + h(m + |u_1| + \ldots, |u_n|) + k \leq h(k + m + |u_1| + \ldots, |u_n|).$$

However,

$$\begin{aligned} p(x+1, y_1, \ldots, y_n) &= h(x+2)(k + x + 1 + y_1 + \ldots + y_n) \\ &\geq h(x+2)(k + x + y_1 + \ldots + y_n) \\ &= h(x+1)(k + x + y_1 + \ldots + y_n) + h(k + x + y_1 + \ldots + y_n) \\ &= p(x, y_1, \ldots, y_n) + h(k + x + y_1 + \ldots + y_n) \end{aligned}$$

The thesis follows. □

By the previous results, we can define the notion of a polytime PRD t with arity n by just counting the number of reductions steps: t is *polytime* iff there is a polynomial $p : \mathbb{N}^n \to \mathbb{N}$ such that whenever $t(u_1, \ldots, u_n) \longrightarrow^m v$, it holds that $m \leq p(|u_1|, \ldots, |u_n|)$. A PRD t is *hereditarily polytime* iff t and every sub-definition of t are polytime. $\mathcal{PD} \subseteq \mathcal{D}$ is the class of all polytime PRDs. Similarly, $\mathcal{HPD} \subseteq \mathcal{PD}$ is the class of all hereditarily polytime PRDs. These classes are defined from the operational semantics. The class $\mathscr{P}$ of polytime functions from $\mathcal{B}^n$ to $\mathcal{B}$ is usually defined in terms of Turing Machines. From corollary 1.1, it follows that $[\![\mathcal{PD}]\!] = [\![\mathcal{HPD}]\!] = \mathscr{P}$. In other words, the cost model induced by the operational semantics is invariant [2]. Notice there are polytime PRDs which are not hereditarily polytime (for example, $comp(const, exp)$, where $[\![const]\!]$ is a constant function and exp takes exponential time)
. We can now define the class of *bounded primitive recursive definitions (BPRD)* as follows[1]: take the definition of a PRD and modify the last inductive clause as follows: If s is BPRD with arity $n \geq 0$ and t_0, t_1 are two BPRDs with arity $n+2$, then $t = rec(s, t_0, t_1)$ is a BPRD with arity $n+1$ provided there is a polynomial $p : \mathbb{N}^{n+1} \to \mathbb{N}$ such that $|[\![t]\!](u_0, \ldots, u_n)| \leq p(|u_0|, \ldots, |u_n|)$ whenever $u_0, \ldots, u_n \in \mathcal{B}$. $\mathcal{BD}$ is the class of all BPRD. Extensionally, bounded primitive recursive definitions capture polynomial time:

Theorem 1.1 (Cobham) $[\![\mathcal{BD}]\!] = \mathscr{P}$.

What we would like, however, is a result on the *intensional* expressive power of bounded recursion. We can easily get it:

Theorem 1.2 $\mathcal{BD} = \mathcal{HPD}$.

This is not too surprising, since the definition of a BPRD involves the existence of a polynomial, precisely as the definition of a polytime primitive recursive definition. In the definition of a BPRD, however, the operational semantics is not mentioned.

References

[1] Cobham, A. The intrinsic computational difficulty of functions. In *Proceedings of 1964 International Congress for Logic, Methodology and Philosophy of Sciences* (1965), pp. 24–30.

[2] van Emde Boas, P. Machine models and simulations. In *Handbook of Theoretical Computer Science*, vol. A. Elsevier Science Publishers, 1990, pp. 1–66.

[1] this way of defining bounded recursion on notation is slightly different from the original one [1]