Theoretical Informatics and Applications Will be set by the publisher
Informatique Théorique et Applications

A COMPLETE CHARACTERIZATION OF PRIMITIVE RECURSIVE INTENSIONAL BEHAVIOURS

P. Valarcher[1]

Abstract. We give a complete characterization of the class of functions that are the intensional behaviours of Primitive Recursive algorithms. This class is the set of primitive recursive functions that have a null basic case of recursion. This result is obtained using the property of ultimate unarity and a geometrical approach of sequential functions on N the set of positive integers.

AMS Subject Classification. — Give AMS classification codes —.

1. Introduction

In [4,5], L. Colson studies the *behaviour* of PR algorithms and proves that there is no algorithm (in PR) that computes the minimum of two (unary) integers in time $O(inf)$. He proves this result by studying the interpretation of PR algorithms on the domain of *lazy* (or *partial*) integers [12]: this domain captures the way an algorithm writes on its output as a function of what it has *already* read on its inputs, this function is called *intensional behaviour*. He notices that all algorithms look ultimately at at most one of their inputs so that ultimate interpretations are unaries functions (this propety is called *ultimate obstinacy*). Later, T. Coquand [7] gives a constructive proof of this property and as side effect he gives a definition of the class of functions that are ultimatly the intensional behaviours of PR algorithms: successor function, constant functions, projections, predecessor, closed by composition and an iteration schema.

The author has shown some new results on intensional behaviour (now called structural complexity) of other primitive recursive schemas (in [17]). In the same framework, L. Colson and D. Fredholm (see [6, 13]) show that call-by-value strategy (with primitive recursion over lists of integers and with primitive recursion in

Keywords and phrases: Intensional behaviour, semantics, primitive recursion

[1] LACL, Université Paris Est

higher types, called system T of Goedel) does not allow to compute the good algorithm of the *min* function. Similar questions have been studied by S. Brookes and D. Dancanet in [3] and [9] with non-determinism and CDS languages. Recently, in [15], Y. Moschovakis has established a linear lower bound for the complexity of non-trivial primitive recursive program from piecewise linear given functions. His main result is that logtime programs for the greatest common divisor from such givens (such as Steins) cannot matched in efficiency by primitive recursive programs from the same given functions. He ended by an open problem relative to the classical Euclidean algorithm (L. Van Den Dries gives a partial answer in [11]). And lastly, T. Crolard, S. Lacas and the author extend the result of incompletness of algorithms to imperative language in [8].

In this paper, we give a complete characterization of the intensional behaviour of PR algorithms: these are functions that verify the property of *sequentiality* [2] and with ultimate behaviours definable by primitive recursive schema with null basic case.

The paper is organized as follow: in section 2 we present the language of PR terms as term rewriting systems, we present some computations (reductions) of terms that contain free variables. In section 3 we give the interpretation of PR terms on the domain of lazy integers, this domain captures the central notion of this paper: *intensional behaviours*. This notion is defined in section 4 with a geometrical approach of sequentiality. Section 5 is devoted to a new class of functions *prn* (primitive recursion with null basic case) and a new language PRN that extensionaly computes *prn* functions. We show that with PRN terms intensionality coincide with extensionality, so intensional behaviours of PRN terms are exactly *prn* functions. In section 6 we define the property of *ultimate unarity* and give a proof that all PR terms have this property. Finally, we prove that *prn* functions are exactly the class of function that are intensional behaviours of PR terms using previous property and the geometrical approach of sequentiality.

2. PR ALGORITHMS

2.1. Terms

We consider an alphabet with the following symbols:

- a countable infinite set of variables x,y,...,
- for each integer $n \geq 1$ a countable infinite set of function symbol of arity n,
- the constant symbol 0,
- the function symbol S of arity 1.

From this alphabet we define the set Term of terms inductively:

- variables are in Term,
- 0 is in Term
- if t is in Term then S(t) is in Term,

- if $\mathtt{f}$ is a function symbol of arity n and if $\mathtt{t_1}, ..., \mathtt{t_n}$ are in $\mathtt{Term}$ then $\mathtt{f(t_1}, ..., \mathtt{t_n})$ is in $\mathtt{Term}$

Terms without variables are said to be *closed*. Otherwise a term is open.

2.2. PR rewriting systems

We define the notion of PR system of rewriting rules inductively:

- The empty set is a PR system of rules.
- If Σ is a PR system of rules, if $\mathtt{t}$ is a term with all function symbols occuring in Σ, with variables among $\mathtt{x_1}, ..., \mathtt{x_n}$, and if $\mathtt{f}$ is a symbol of function of arity n not in Σ, then $\Sigma \cup \{\mathtt{Exp_{f,t}}\}$ is a PR system of rules where $\mathtt{Exp_{f,t}}$ is defined by the following rule:

$$\mathtt{f(x_1}, ..., \mathtt{x_n}) \quad = \quad \mathtt{t}$$

  We say that $\mathtt{f}$ is defined by explicit definition (or rule).
- If Σ is a PR system of rules, if $\mathtt{g}, \mathtt{h}$ are functions symbols occuring in Σ with arity n and $n+2$ respectively, and if $\mathtt{f}$ is a function symbol of arity $n+1$ not occuring in Σ, then $\Sigma \cup \{\mathtt{Rec_{f,g,h}}\}$ is a PR system of rules where $\mathtt{Rec_{f,g,h}}$ is the pair of rules:

$$\begin{array}{lcl} \mathtt{f(0, x_1}, ..., \mathtt{x_n}) & = & \mathtt{g(x_1}, ..., \mathtt{x_n}) \\ \mathtt{f(S(x), x_1}, ..., \mathtt{x_n}) & = & \mathtt{h(x, f(x, x_1}, ..., \mathtt{x_n}), \mathtt{x_1}, ..., \mathtt{x_n}) \end{array}$$

  We say that $\mathtt{f}$ is defined by recursive definition (or rule) from $\mathtt{g}$ and $\mathtt{h}$.

2.3. Examples of PR systems

$\mathtt{add_1}$ system. This system computes the addition of two integers, and is defined by:

$$\begin{array}{lcl} \mathtt{Id(x)} & = & \mathtt{x} \\ \mathtt{pi_3s(x, y, z)} & = & \mathtt{S(y)} \\ \mathtt{add_1(0, y)} & = & \mathtt{Id(y)} \\ \mathtt{add_1(S(x), y)} & = & \mathtt{pi_3s(x, add1(x, y), y)} \end{array}$$

Remark 2.1. For sake of simplicity we write such a system as: $\mathtt{add_1(0, y)} = \mathtt{y}$ and $\mathtt{add_1(S(x), y)} = \mathtt{S(add_1(x, y))}$

$\mathtt{add_2}$ system.

$$\mathtt{add_2(x, y)} \quad = \quad \mathtt{add_1(y, x)}$$

Following 2.1, instead of this explicit definition, we may use the following simpler system: $\mathtt{add_2(x, 0)} = \mathtt{x}; \mathtt{add_2(x, S(y))} = \mathtt{S(add_2(x, y)}$.
pred system. Using the last simplification: $\mathtt{pred(0)} = \mathtt{0}; \mathtt{pred(S(x))} = \mathtt{x}$

2.4. Computation

Let Σ be a PR system of rules. We say that a *redex* is a term that has one of the following forms:

(1) $\mathtt{f}(\mathtt{t}_1, ..., \mathtt{t}_n)$, where $\mathtt{f}$ is defined by explicit rule.
(2) $\mathtt{f}(\mathtt{0}, \mathtt{t}_1, ..., \mathtt{t}_n)$, where $\mathtt{f}$ is defined by recursive rule.
(3) $\mathtt{f}(\mathtt{S}(\mathtt{t}), \mathtt{t}_1, ..., \mathtt{t}_n)$, where $\mathtt{f}$ is defined by recursive rule.

We denote by $\mathtt{f}[\mathtt{v}_1/\mathtt{x}_1, \cdots, \mathtt{v}_n/\mathtt{x}_n)$ the usual capture-avoiding substitution. We define the *one-step reduction* relation $\to_1$ as the smallest relation between terms that verifies:

(1) If $\mathtt{f}(\mathtt{x}_1, ..., \mathtt{x}_n) = \mathtt{t}$ is in Σ, $\mathtt{f}(\mathtt{t}_1, ..., \mathtt{t}_n) \to_1 \mathtt{t}[\mathtt{t}_1/\mathtt{x}_1, ..., \mathtt{t}_n/\mathtt{x}_n]$.
(2) If $\mathtt{f}$ is defined by recursive rule :
   - $\mathtt{f}(\mathtt{0}, \mathtt{t}_1, ..., \mathtt{t}_n) \to_1 \mathtt{g}(\mathtt{t}_1, ..., \mathtt{t}_n)$
   - $\mathtt{f}(\mathtt{S}(\mathtt{t}), \mathtt{t}_1, ..., \mathtt{t}_n) \to_1 \mathtt{h}(\mathtt{t}, \mathtt{f}(\mathtt{t}, \mathtt{t}_1, ..., \mathtt{t}_n), \mathtt{t}_1, ..., \mathtt{t}_n)$

and is compatible with contexts: if $\mathtt{t} \to_1 \mathtt{u}$ then

- $\mathtt{S}(\mathtt{t}) \to_1 \mathtt{S}(\mathtt{u})$
- $\mathtt{j}(\mathtt{u}_1, ..., \mathtt{t}, ..., \mathtt{u}_n) \to_1 \mathtt{j}(\mathtt{u}_1, ..., \mathtt{u}, ..., \mathtt{u}_n)$

We denote the reflexive and transitive closure of $\to_1$ with $\to$. A term $\mathtt{t}$ is *irreducible* (or in *normal form*) if there is no $\mathtt{u}$ such that $\mathtt{t} \to \mathtt{u}$.

Remark 2.2.
- the term $\mathtt{f}(\mathtt{x}, \mathtt{t}_1, ..., \mathtt{t}_n)$ where $\mathtt{f}$ is defined by recursion rule is not a redex ($\mathtt{x}$ is a variable),
- a irreducible term may be an open term: from the $\mathtt{add}_1$ system we compute:

$$\begin{array}{rcl} \mathtt{add}_1(\mathtt{S}(\mathtt{S}(\mathtt{x})), \mathtt{add}_1(\mathtt{0}, \mathtt{S}(\mathtt{0}))) & \to_1 & \mathtt{S}(\mathtt{add}_1(\mathtt{S}(\mathtt{x}), \mathtt{add}_1(\mathtt{0}, \mathtt{S}(\mathtt{0})))) \\ & \to_1 & \mathtt{S}(\mathtt{add}_1(\mathtt{S}(\mathtt{x}), \mathtt{S}(\mathtt{0})) \\ & \to_1 & \mathtt{S}(\mathtt{S}(\mathtt{add}_1(\mathtt{x}, \mathtt{S}(\mathtt{0})) \end{array}$$

2.5. Classical results

We recall some main results concerning PR systems: they compute primitive recursive functions and satisfy the strong normalisation property and the Church-Rosser property.

Definition 2.3. The class of primitive recursive functions is defined as the smallest set of functions $f : N^n \to N$ containing 0, *Succ*, projections that is closed by composition and by the following schema: from $g : N^n \to N$ and $h : N^{n+2} \to N$ we define $f : N^{n+1} \to N$ by

$$\begin{array}{rcl} f(0, x_1, ..., x_n) & = & g(x_1, ..., x_n) \\ f(Succ(x), x_1, ..., x_n) & = & h(x, f(x, x_1, ..., x_n), x_1, ...x_n) \end{array}$$

The two following theorems say that all primitive recursive functions are computable by a PR system and that all PR systems defines a *p.r.* function.

Theorem 2.4. *Let f be a p.r. function. There exists a* PR *system containing the function symbol* f *such that, for all integers $k_1, ..., k_n$*

$$\mathtt{f}(\mathtt{S}^{k_1}(\mathtt{0}), ..., \mathtt{S}^{k_n}(\mathtt{0})) \quad =_{\mathtt{PR}} \quad \mu(f(Succ^{k_1}(0), ..., Succ^{k_n}(0))$$

where μ is defined inductively by $\mu(0) = \mathtt{0}$ and $\mu(Succ(t)) = \mathtt{S}(\mu(t))$. And $=_{\mathtt{PR}}$ is the reflexive, symmetric and transitive closure of $\rightarrow_1$.

Theorem 2.5. *Let Σ be a* PR *system and let* f *be a function symbol over Σ. Then* f *computes a p.r. function.*

Theorem 2.6. **Strong Normalization:** *Let* t *be a term of a* PR *system. All sequences of reduction starting by* t *are finite.*

Church-Rosser: *Let* t *be a term of a* PR *system such that* $\mathtt{t} \rightarrow \mathtt{u}$ *and* $\mathtt{t} \rightarrow \mathtt{v}$ *then there exists a term* w *such that* $\mathtt{u} \rightarrow \mathtt{w}$ *and* $\mathtt{v} \rightarrow \mathtt{w}$.

Remark 2.7. An irreducible closed term is of the form $\mathtt{S}^k(\mathtt{0})$ for a non-negative integer k.

2.6. Reduction of open terms

As we have seen PR systems compute *p.r.* functions when inputs are of the form $\mathtt{S}^k(\mathtt{0})$. We are interested in the class of functions that are computed by PR systems when all inputs are of the form $\mathtt{S}^k(\mathtt{x})$ where x is a variable. Let T be the following PR system:

$$\begin{array}{lcl} \mathtt{add}_1(\mathtt{0},\mathtt{y}) & = & \mathtt{y} \\ \mathtt{add}_1(\mathtt{S}(\mathtt{x}),\mathtt{y}) & = & \mathtt{S}(\mathtt{add}_1(\mathtt{x},\mathtt{y})) \\ \mathtt{t}(\mathtt{x}) & = & \mathtt{add}_1(\mathtt{S}(\mathtt{0}),\mathtt{add}_1(\mathtt{S}(\mathtt{S}(\mathtt{x})),\mathtt{add}_1(\mathtt{0},\mathtt{S}(\mathtt{0})))) \end{array}$$

Look at a computation of t(0) and t(y) (of course we may compute more than one redex at each step-reduction):

$$\begin{array}{lcl} \mathtt{t}(\mathtt{0}) & \rightarrow_1 & \mathtt{S}(\mathtt{add}_1(\mathtt{0},\mathtt{add}_1(\mathtt{S}(\mathtt{S}(\mathtt{0})),\mathtt{add}_1(\mathtt{0},\mathtt{S}(\mathtt{0}))))) \\ & \rightarrow_1 & \mathtt{S}(\mathtt{add}_1(\mathtt{S}(\mathtt{S}(\mathtt{0})),\mathtt{add}_1(\mathtt{0},\mathtt{S}(\mathtt{0})))) \\ & \rightarrow_1 & \mathtt{S}(\mathtt{S}(\mathtt{add}_1(\mathtt{S}(\mathtt{0}),\mathtt{add}_1(\mathtt{0},\mathtt{S}(\mathtt{0})))))) \\ & \rightarrow_1 & \mathtt{S}(\mathtt{S}(\mathtt{S}(\mathtt{add}_1(\mathtt{0},\mathtt{add}_1(\mathtt{0},\mathtt{S}(\mathtt{0}))))))) \\ & \rightarrow_1 & \mathtt{S}(\mathtt{S}(\mathtt{S}(\mathtt{add}_1(\mathtt{0},\mathtt{S}(\mathtt{0})))))) \\ & \rightarrow_1 & \mathtt{S}(\mathtt{S}(\mathtt{S}(\mathtt{S}(\mathtt{0}))))) \end{array}$$

and

$$\begin{array}{lcl} \mathtt{t}(\mathtt{y}) & \rightarrow_1 & \mathtt{S}(\mathtt{add}_1(\mathtt{0},\mathtt{add}_1(\mathtt{S}(\mathtt{S}(\mathtt{y})),\mathtt{add}_1(\mathtt{0},\mathtt{S}(\mathtt{0}))))) \\ & \rightarrow_1 & \mathtt{S}(\mathtt{add}_1(\mathtt{S}(\mathtt{S}(\mathtt{y})),\mathtt{add}_1(\mathtt{0},\mathtt{S}(\mathtt{0})))) \\ & \rightarrow_1 & \mathtt{S}(\mathtt{S}(\mathtt{add}_1(\mathtt{S}(\mathtt{y}),\mathtt{add}_1(\mathtt{0},\mathtt{S}(\mathtt{0})))))) \\ & \rightarrow_1 & \mathtt{S}(\mathtt{S}(\mathtt{S}(\mathtt{add}_1(\mathtt{y},\mathtt{add}_1(\mathtt{0},\mathtt{S}(\mathtt{0}))))))) \\ & \rightarrow_1 & \mathtt{S}(\mathtt{S}(\mathtt{S}(\mathtt{add}_1(\mathtt{y},\mathtt{S}(\mathtt{0})))))) \end{array}$$

The two computations do not lead to the same normal form, moreover they do not lead to the same number of S in front of their normal form. More generally,

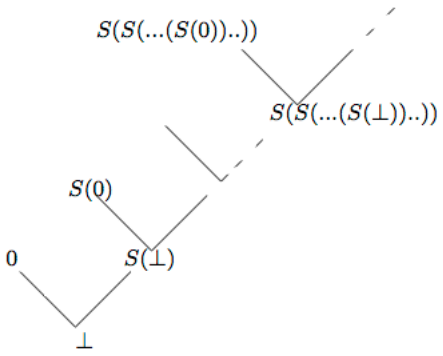

FIGURE 1. The domain D of lazy integers

if we give to the term $\mathtt{t}$ an input of the form $\mathtt{S}^k(\mathtt{0})$ we obtain $\mathtt{S}^{k+4}(\mathtt{0})$ (as normal form) and with input $\mathtt{S}^k(\mathtt{y})$ we obtain $\mathtt{S}^{k+3}(\mathtt{u})$, where $\mathtt{u}$ is irreducible.

To capture these behaviours, we use denotational semantics with the specific domain of lazy natural numbers (following [4, 5]).

3. Finite denotational semantics

The domain of lazy integers D captures the above intuition.

3.1. The domain

Elements of D have the following form: $D_0 = \{S^k(0)\}_{k\in N}$ or $D_\perp = \{S^k(\perp)\}_{k\in N}$. They are ordered by the $<_D$ relation (see fig. 1):

- $\perp <_D x$, for all $x \neq \perp$,
- if $u <_D v$ then $S(u) <_D S(v)$

Remark 3.1. $\perp$ does not have the standard denotational semantic meaning of *a non terminating computation*. Here, it means *a computation that is not yet finished*.

3.2. Interpretation

Since terms of PR systems contain variables, the interpretation of a term needs a context which allow us to assign a value of D to each variables.

Definition 3.2. An environment ρ for the variables $\mathtt{x}_1, ..., \mathtt{x}_n$ is a function from $\{\mathtt{x}_1, \cdots, \mathtt{x}_n\}$ to D that assigns to each variables a value $\rho(\mathtt{x}_i)$ of D.

From an environment ρ, a value $u \in D$ and a variable $\mathtt{y}$, we construct the environment $\rho' = \rho[\mathtt{y} \leftarrow u]$ such that $\rho'(\mathtt{x}) = \rho(\mathtt{x})$, if $\mathtt{x} \neq \mathtt{y}$ and $\rho'(\mathtt{y}) = u$ otherwise.

We now define by simultaneous induction the interpretation $Sem(\mathtt{t})_\rho$ of terms over PR systems relatively to an environment ρ and $sem(\mathtt{f})$ of function symbols over PR systems:

- interpretation of terms:
  - $Sem(\mathtt{0})_\rho = 0$,
  - $Sem(\mathtt{x_i})_\rho = \rho(\mathtt{x_i})$,
  - $Sem(\mathtt{S(u)})_\rho = S(Sem(\mathtt{u})_\rho)$,
  - $Sem(\mathtt{f(t_1,...,t_n)})_\rho = sem(\mathtt{f})(Sem(\mathtt{t_1})_\rho, ..., Sem(\mathtt{t_n})_\rho)$.
- interpretation of function symbols:
  - if $\mathtt{f}$ is defined by explicit rule $(\mathtt{f(x_1,...,x_n)} = \mathtt{t})$, then

$$sem(\mathtt{f})(d_1, ..., d_n) \quad = \quad Sem(\mathtt{t})_{[\mathtt{x_1} \leftarrow d_1, ..., \mathtt{x_n} \leftarrow d_n]}$$

  - if $\mathtt{f}$ is defined by recursive rules from $\mathtt{g}$ and $\mathtt{h}$, then

$$sem(\mathtt{f})(d, d_1, ..., d_n) \quad = \quad (\lambda x_1 ... x_n.\theta(d))d_1 \, ... \, d_n$$

with θ defined by:

$$\begin{array}{lcl} \theta(\bot) & = & \bot \\ \theta(0) & = & sem(\mathtt{g})(x_1, ...x_n) \\ \theta(S(m)) & = & sem(\mathtt{h})(m, \theta(m), x_1, ...x_n) \end{array}$$

For the sake of simplicity, if $\mathtt{t}$ is a term over a PR system we write $sem(\mathtt{t})(d_1, ..., d_n)$ instead of $Sem(\mathtt{t})_{[\mathtt{x_1} \leftarrow d_1, ..., \mathtt{x_n} \leftarrow d_n]}$.

4. INTENSIONAL BEHAVIOUR

We now define the central notion of the paper:

Definition 4.1. The intensional behaviour of a term $\mathtt{t}$ over a PR system is the function $Int(\mathtt{t}) : N^n \to N$ defined by:

$$Int(\mathtt{t})(k_1, ...k_n) = n \Leftrightarrow sem(\mathtt{t})(S^{k_1}(\bot), ..., S^{k_n}(\bot)) = S^n(\iota), \ \iota = 0 \text{ or } \bot$$

The*intensionality* of $\mathtt{t}$ is the function $sem(\mathtt{t})$ defined on $D_\bot$ (elements of the form $S^k(\bot)$ for $k \geq 0$).

Remark 4.2. The **extensional function associated to the term** $\mathtt{t}$, named $Ext(\mathtt{t})$, is defined by:

$$Ext(\mathtt{t})(k_1, ...k_n) = n \Leftrightarrow sem(\mathtt{t})(S^{k_1}(0), ..., S^{k_n}(0)) = S^n(0)$$

4.1. Examples

- $Int(\texttt{add}_1) = \pi_1^2$, $Ext(\texttt{add}_1) = +$,
- $Int(\texttt{add}_2) = \pi_2^2$, $Ext(\texttt{add}_2) = +$
- $Int(\texttt{pred}) = pred$, $Ext(\texttt{pred}) = pred$.

We may now state the central question of this paper:

> "What can we say about the class of functions that are intensional behaviours of PR systems ?"

We first recall a well-known result from Theory of Domains that restricts the class of intensional behaviours (it is not a restriction of $mathttPR$ systems but is due to the sequentiality of the language [1, 2]). We give a geometrical approach to the class of functions that may be intensional behaviours.

4.2. Geometrical approach of intensional behaviour

From results in denotational semantics, we know that the semantics of PR systems have the property of *sequentiality*. Intuitively a function f is sequential in $(x_1, \ldots, x_n)$ if either the function is constant or there exists i such that the incompleteness of f "comes from" that of x_i. More formally (from [1]):

Definition 4.3.
- f is *monotonic* if X and Y are two uplets of terms and $X \leq Y \Rightarrow f(X) \leq f(Y)$.
- f is *continuous* if f is monotonic and verifies

$$f(max\{X, Y\}) = max\{f(X), f(Y)\}$$

- $f : D^m \to D$ is *sequential* iff f is monotonic, continuous and satisfies the following condition: for all $X \in D^m$, if $f(X) = S^k(\bot)$ then
  - either for all $X <_D Y$, we have $f(Y) = S^k(\bot)$,
  - or there exists an integer $i \in \{1, \ldots, m\}$ and $X(i) = S^p(\bot)$ for some p, and for all $Y \in D^n$ if $S^k(\bot) <_D f(Y)$ then $S^p(\bot) <_D Y(i)$

  Note: The order is that of D !

The integer i is called a *sequentiality index* for f at X.

Remark 4.4. Using the informational interpretation of the order on D, this means that for $f(Y)$ to be more informative than $f(X)$, it is necessary that for some i, $Y(i)$ is more informative than $X(i)$.

Theorem 4.5. *Every denotation of an algorithm of* PR *verifies the property of sequentiality.*

As intensional behaviour is defined over N, we translate the definition of sequentiality to this domain.

Definition 4.6. We say that a function $\rho : N^m \to N$ is **step-stair** iff ρ is monotone, continuous and satisfies the following condition: for all $X \in N^m$, if $f(X) = k$ then

- either for all $Y \geq X$, we have $f(Y) = k$,
- or there exists an integer $i \in 1, \ldots, m$ and $X(i) = p$, and for all $Y \in N^n$ if $f(Y) > k$ then $Y(i) > p$

Note: The order is the usual order on N.

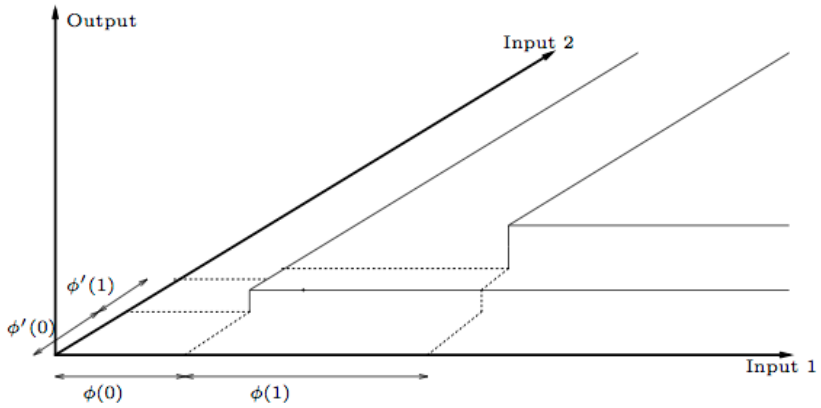

FIGURE 2. A binary step-stair function

In fact, on N, the notion of sequentiality and stability coincide (f is *stable* if f is monotonic and verify for all $X, Y \in N^n$, $f(X \wedge Y) = f(X) \wedge f(Y)$). Then using stability instead of sequentiality we can prove that

Proposition 4.7.
- *ρ is a step-stair function iff ρ is stable.*
- *and then if* `f` *is a term over a* `PR` *system then $Int($*`f`*$)$ is a step-stair function.*

A more suggestive presentation of step-stair functions uses the following notion:

Definition 4.8. Given A, B in N^m, with $A \leq B$, we call **Corner** the set:

$$Co(A, B) \quad = \quad \{M \in N^m | M \geq A \text{ and } M \not\geq B\}$$

We also consider the case when B is missing.

A is called the *origin* point and B is called the *extrem* point of the corner $Co(A, B)$. The following proposition is easy to prove:

Proposition 4.9. *f is step-stair iff f is increasing and $f^{-1}(\alpha)$ is a corner for all α in range(f).*

Proof. The proof is straightforward by the stability of f. □

Remark 4.10.
- If α and β are consecutive elements of *range*(f) then the origin point of $f^{-1}(\beta)$ is obtained by incrementing some components of the extrem point of $f^{-1}(\alpha)$ (namely the x component in case (1), the y in case (2), both in case (3) of Fig. 3).

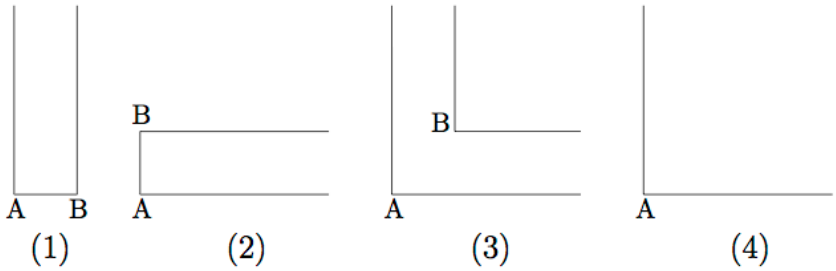

FIGURE 3. The four kinds of corners in N^2

- $range(f)$ is finite iff $f^{-1}(max(range(f)))$ has an infinite extrem point.

We may now define the origin sequence of a function f.

Definition 4.11. Let $range(f) = \{\alpha_0, \alpha_1, \cdots\}$ with $\alpha_0 < \alpha_1 < \cdots$ and A_i be the origin point of $f^{-1}(\alpha_i)$. The sequence $(A_0, A_1, \cdots)$ is called the **origin sequence** of f.

Following [14] the origin sequence may be called *trace* of the function.

5. *p.r.n.* FUNCTIONS AND PRN SYSTEMS

We introduce in this section a language that has a class of extensional functions equals to its class of intensional behaviours.

Definition 5.1. The class of *p.r.n* functions (recursive primitive with basic case null) is the smallest set of functions from $N^n \to N$ containing 0, Succ, projections and which is closed by composition and schema prn: from $h : N^{n+1} \to N$ we construct f by:

$$\begin{array}{lcl} f(0, Y) & = & 0 \\ f(Succ(x), Y) & = & h(x, f(x, Y), Y) \end{array}$$

We define PRN systems as PR systems in which all recursive rules are of the form $\mathtt{Rec}_{\mathtt{f},0,\mathtt{h}}$. We now prove that the intensional behaviour of a term $\mathtt{t}$ over a PRN system ($Int(\mathtt{t})$) is the extensionnal function associated to $\mathtt{t}$ ($Ext(\mathtt{t})$) (this function is in *p.r.n*).

Theorem 5.2. *Let* $\mathtt{t}$ *be a term over a* PRN *system with variables* $\mathtt{x}_1, ..., \mathtt{x}_\mathtt{p}, \mathtt{y}_1, ..., \mathtt{y}_\mathtt{n}$. *Let* ρ *and* θ *be two environment such that* $\rho(\mathtt{y}_\mathtt{j}) = \theta(\mathtt{y}_\mathtt{j}) = S^{l_j}(0)$, $\rho(\mathtt{x}_\mathtt{i}) = S^{k_i}(0)$ *and* $\theta(\mathtt{x}_\mathtt{i}) = S^{k_i}(\bot)$. *Then* $Sem(\mathtt{t})_\rho = S^l(0)$ *and* $Sem(\mathtt{t})_\theta = S^l(\iota)$ ($\iota = 0$ *or* $\bot$).

Proof. For the sake of simplicity, we consider only the case $p = 1$. From the inductive definition of Sem, we reduce to the case when $\mathtt{f}$ is defined by recursion over $\mathtt{x}_1$ and $k_1 = 0$ so that $Sem(\mathtt{t})_\theta = \bot$. But $\mathtt{t}$ is in PRN so $Sem(\mathtt{t})_\rho = 0$. □

Corollary 5.3. $Int(\texttt{PRN}) = Ext(\texttt{PRN}) = p.r.n.$

Proof. $Ext(\texttt{PRN}) = p.r.n$ by definition of Ext and $Int(\texttt{PRN}) = Ext(\texttt{PRN})$ by theorem 5.2. □

6. Ultimate unarity

We give now a new proof of the well known result [4, 5, 7] about the intensional behaviours of terms over PR systems, *i.e.* they obstinate on only one of its arguments.

Definition 6.1. We say that a step-stair function $f : N^n \to N$ is **ultimately unary of index** i, $1 \leq i \leq n$, **from the point** $A \in N^n$ if $\forall X \geq A$, i is a sequentiality index for f in X, *i.e.*

$$\forall X, Y \geq A, (X(i) = Y(i)) \to (f(X) = f(Y))$$

In particular, for $X \geq A$, $f(X)$ depends only on $X(i)$.

Notations: If f is ultimately unary then:

- we denote by A_f the smallest point of N^n such that f is ultimately unary from A_f and such that $\forall X \not\geq A_f$ then $f(X) < f(A_f)$. It's the first origin point from which the index of sequentiality is constant afterward.
- we denote by $Ult_f : N \to N$ the unary function defined by

$$\begin{array}{lcl} Ult_f(z) & = & 0 \text{ if } z \leq A_f(i) \\ Ult_f(z) & = & f(A_f(1), \cdots, z, \cdots, A_f(n)) \end{array}$$

The proof of the previous theorem gives the ultimate class of intensional behaviours as in [7] except that it's not constructive. Let $\texttt{t}$ be a term of a PR system then:

Theorem 6.2. *The intensional behaviour of* $\texttt{t}$ *is ultimately unary.*

Proof. The proof follows the one of T. Coquand in [7]. It can be found in [16]. □

7. Full characterization

Corollary 7.1. Coquand [7] *The ultimate intensional behaviours of PR systems are among the asymptotic behaviours of the smallest set containing constant functions, identity functions beyond a certain rank, and which is closed by composition and the following schemas:* $n \mapsto \phi(n-1)$ *and* $n \mapsto \phi^n(N_0)$ *if* $\phi(N_0) \geq N_0$, N_0 *an integer. We denote this class by* $Ult(PR)$.

Remark 7.2. In fact, this class shows that we do not have sub-linear and polynomial ultimate intensional behaviours with PR systems.

7.1. Ultimate intensional behaviours are *prn*

We show that the class $Ult(PR)$ is in *prn*.

Proposition 7.3. *The function pred is in prn and if ϕ is an increasing function in prn such that $\phi(N_0) \geq N_0$ then $n \mapsto \phi^n(N_0)$ is in prn.*

Proof.

- the *pred* function is defined by:

$$pred(0) = 0 \quad ; \quad pred(S(p)) = p$$

- for the iteration $n \mapsto \phi^n(N_0)$ with N_0 an integer and $\phi(N_0) \geq N_0$, we define $h(n) = Succ^{N_0}(h'(n))$ as followed:

$$h'(0) = 0 \quad ; \quad h'(S(n)) = pred^{N_0}(\phi(S^{N_0}(h'(n))))$$

□

7.2. Outside the ultimate intensional behaviours

Using the geometrical approach of the trace of a step-stair *pr* functions, we characterize the set of points that are not concerned with the ultimate unarity (the beginning of the trace of the function).

We introduce some basic functions:

$$\begin{array}{l} if_0(0,y) = 0; if_0(S(x),y) = y \\ if_{a+1}(0,y) = 0; if_{a+1}(S(x),y) = if_a(x,y) \\ distr_{(i,p,a)}(X,z) = if_a(X(i),z) \\ distr_{(i,p,a)@l}(X,z) = distr_{(i,p,a)}(X, S^p(distr_l(X,z))) \end{array}$$

The function if_a is defined such that if the first input is greater than a then the value is the second input else 0 (if_a is in *prn*).

The function *distr* is a composition of if_a. It is indexed by a list of triplets (i,p,a) where i is a place in the list of inputs. Using if_a, it tests if the input on the i^{th} argument is greater than S^a and produces p S's, else it stops (given a list l, $distr_l$ is a macro definable in *prn*).

Theorem 7.4. *The class of functions that are intensional behaviours of PR systems is exactly the set of prn functions.*

Moreover, this intensional behaviour may be written with the canonical following form using distr function: $distr_s(X, f(X(i)))$ where f is a unary function of $Ult(PR)$ and s the beginning of the trace.

Proof. Let `t` be a term of a PR system. We construct the list s from the trace $T_0 = 0, T_1, \cdots, T_k = A_t$ of the intensional behaviour of `t`: $s = (i_1, p_1, a_1), \cdots, (i_k, p_k, a_k)$ where i_j and a_j are such that T_j is obtained from T_{j-1} by incrementing by a_j the i_jth argument and p_j is $t(T_j) - t(T_{j-1})$. We let $f = pred^{p_1+\cdots+p_k}$ *o* Ult_t and $i = i_t$. □

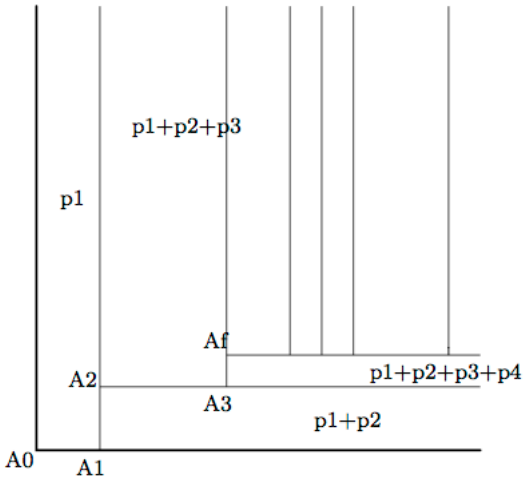

FIGURE 4. The graph of an ultimately unary function.

The following term matches with Fig. 4:

$$\mathtt{S}^{\mathtt{p1}}(\mathtt{if1}(\mathtt{x},\mathtt{S}^{\mathtt{p2}}(\mathtt{if1}(\mathtt{y},\mathtt{S}^{\mathtt{p3}}(\mathtt{if5}(\mathtt{x},\mathtt{S}^{\mathtt{p4}}(\mathtt{if2}(\mathtt{y},\mathtt{g}(\mathtt{x}))))))))))$$

where $\mathtt{g}$ is an unary function. We may use the *distr* function as follows:

$$distr_{(1,p1,1),(2,p2,1),(1,p3,5),(2,p4,2)}(x,y,g(x))$$

8. Conclusion

From denotational semantics theory, it is well know that PR systems may compute only sequential function on $D_\perp$ domain. Moreover PR systems verify the stronger property called *ultimately unary* or *ultimate obstinacy* [4,5,7]. We give in this paper a geometrical approach of sequentiality (and stability) that allows us to reason with sequential function. A new (and short) proof of ultimate unarity is sketched allowing to remark that ultimate intensional behaviours are definable with *prn* functions. Geometrical approach and remarks on *prn* functions give a complete (but non constructive) characterization of PR intensional behaviours.

Depending on the ultimate property this may not be extended to some other languages such as Mutual Primitive Recursion (PRM) and Alternate PRimitive Recursion (PRA) where a similar result may occur, *i.e.* the class of functions that are intensional behaviours of PRM (resp. PRA) systems is exactly the set of functions definable with *prmn* (resp. *pran*) (PRM with null basic case of recursion).In a forthcoming paper, R. David and the author prove that the framework of trace

(introduced by R. David in [10]) allows to extend the result of this paper for PRM and PRA.

I am grateful to Pr. Serge Grigorieff for his help and his permanent support.

References

[1] Roberto Amadio and Pierre-Louis Curien. *Domains and Lambda-Calculi*. Number 46 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1998.
[2] G. Berry. Séquentialité de l'évaluation formelle des lambda-expressions. *3eme Colloque International sur la Programmation*, Paris, 1978.
[3] Stephen Brookes and Denis Dancanet. Sequential algorithms, deterministic parallelism, and intensional expressiveness. In *22nd Annual Symposium on POPL*, 1995.
[4] L. Colson. About primitive recursive algorithms. *Theoretical Computer Science*, 372, 1989.
[5] L. Colson. About primitive recursive algorithms. *Lecture Notes in Computer Science*, 83, pp57-69, 1991.
[6] L. Colson and D. Fredholm. System t, call-by-value and the minimum problem. *Theoretical Computer Science*, 206, 1998.
[7] T. Coquand. Une preuve directe du théclvorème d'ultime obstination. *Compte Rendus de l'Académie des Sc.*, 314, Serie I, 1992.
[8] T. Crolard, S. Lacas, and P. Valarcher. On the expressive power of loop language. *Nordic Journal of Computing*, 13:46–57, 2006.
[9] D. Dancanet and S. Brookes. Programming language expressiveness and circuit complexity. In *Internat. Conf. on the Mathematical Foundations of Programming Semantics*, 1996.
[10] R. David. On the asymptotic behaviour of primitive recursive algorithms. *Theor. Comput. Sci.*, 266(1-2):159–193, 2001.
[11] L. Van Den Dries. Generating the greatest common divisor, and limitations of primitive recursive algorithms. *to appear in Foundations of Computational Mathematics*, 2003.
[12] Martin Hotzel Escardo. On lazy natural numbers with applications. *SIGACT News*, 24(1), 1993.
[13] D. Fredholm. Computing minimum with primitive recursion over lists. *Theoretical Computer Science*, 163, 1996.
[14] P. Taylor J.Y. Girard, Y. Lafont. *Proofs and Types*, volume 7. Cambridge Tracts in Theorical Comp. Sci., 1989.
[15] Yiannis N. Moschovakis. On primitive recursive algorithms and the greatest common divisor function. *Theor. Comput. Sci.*, 301(1-3):1–30, 2003.
[16] P. Valarcher. Contribution à l'etude du comportement intentionel des algorithmes: le cas de la récursion primitive. *Thèse de doctorat, Université P 7*, 1996.
[17] P. Valarcher. Intensionality vs extensionality and primitive recursion. *ASIAN Computing Science Conference - LNCS*, 1179, 1996.

Communicated by (The editor will be set by the publisher).
(The dates will be set by the publisher).