R.A.I.R.O. Informatique Théorique/Theoretical Computer Science
(vol. 11, n° 1, 1977, p. 3 à 16)

NOTE ON THE SUPPRESSION OF LABELS IN PROGRAMMING

English translation

by F. H. RAYMOND*

Communicated by M. Nivat

Translator's note. This is an English translation of F. H. Raymond, "Note sur la suppression des étiquettes en programmation," *RAIRO – Informatique théorique*, tome 11, n° 1 (1977), p. 3–16, published by AFCET and archived by Numdam (`http://www.numdam.org/item?id=ITA_1977__11_1_3_0`). This is the direct sequel to Raymond's 1975 note "Note sur l'algèbre des fonctions" [ref. 7 below], of which an English translation was separately prepared. It is the second of the two Raymond papers cited by John Backus in "The Algebra of Functional Programs" (LNCS 107, 1981) as [Raym77].

The source scan's two schematic branch/bifurcation diagrams in § 2.4 were not legibly recoverable at the level of individual arrows and brace positions; they are reproduced here as illustrative reconstructions of the concept described in the surrounding text (nested packaging with forward exits and backward branches), clearly marked as such, rather than as guaranteed letter-for-letter transcriptions of the original figures. The two worked examples in § 3.3 ("Example 1°" and "2°") were also affected by heavy OCR corruption in the source; the formulas given there are a best-effort reconstruction consistent with the surrounding pattern of the text, flagged accordingly.

The definition and properties of a formal algebra called the algebra of functions were the subject of an earlier note, ref. [7]. The present note gives an extension of it by the introduction of a packaging operator*, together with functions allowing one to exit from it and functions allowing its execution to be repeated. By restricting the use of packaging to loop-functions (expressing the semantics of program loops), the proposed approach joins that of J. Arsac and expresses the semantics of compound instructions containing EXIT (i) instructions, i an integer.*

1. Introduction

This note follows on[1] from [7] by introducing into the algebra of functions $\mathcal{A}$ a new operator, called the *packaging operator*, and into the set of functions, those that express within $\mathcal{A}$ the concept of the EXIT (i) instruction on the one hand, and that of a backward branch or loop on the other.

Let us recall the definition of $\mathcal{A}$:

Definition: $\mathcal{A}$ is an algebra with operators whose terms form the set F (the set of functions) and whose operators form the set S:

$$S = \{\mathrm{COND}, \mathrm{REC}, S^0, S^1, S^2, \dots\},$$

COND, REC, S^0, S^1, ... are atomic symbols. To $\mathcal{A}$ is associated the formal algebra denoted $\hat{\mathcal{A}}$, as is classical. If $f_1, \dots, f_n \in \mathcal{A}$ then:

$$f_1 \dots f_n S^n \in \hat{\mathcal{A}}, \qquad f_1 f_2 f_3 \mathrm{COND} \in \hat{\mathcal{A}}, \qquad \chi\varphi(\chi)\mathrm{REC} \in \hat{\mathcal{A}}.$$

*Conservatoire National des Arts et Métiers.

[1] The reader is kindly asked to correct the statement of Property 1 in § 6 of [7] by deleting "and only if."

In $\chi\varphi(\chi)$REC, $\varphi(\chi)$ denotes a functional scheme of $\hat{\mathcal{A}}$ having at least one occurrence of some arbitrary functional scheme χ.

Definition: Let F_0 be a nonempty subset of F, and $\hat{\mathcal{A}}_{F_0}$ the functional subalgebra generated by F_0; we call a *function* any member of the functional closure of F_0.

Remark: We agree to express by $f \in \mathcal{A}$ the membership of a function f in $\mathcal{A}$, or the membership of a functional scheme in $\hat{\mathcal{A}}$.

The product of functions, usually denoted $f_1 \circ f_2$ and composed by the operator S^2 according to $f_1 f_2 S^2$ in the algebra of functions, can be regarded in two ways.

On the first view, $f_1 \circ f_2$ and $f_1 f_2 S^2$ are objects of the same nature as f_1 and f_2 taken separately, and are apprehended as such by an executor. Thus if the symbol S^2 is attached to f_2 as a suffix, $f_2 S^2$ in isolation has no meaning for any executor, but acquires a meaning once $f_2 S^2$ is associated with another function f_1 with which we form a product: the executor understands that f_2 is to be applied to the object (or objects) produced by f_1.

This "passing" of objects is inseparable from the sequentiality of a control structure, but sequentiality can be the primary principle if the executor "understands" each of the symbols in $f_1 \circ f_2$ as follows: f_1 and f_2 designate actions or instructions to it, and $\circ$ instructs it, after executing f_1, to continue by executing f_2. This is the second way of regarding $f_1 \circ f_2$ and $f_1 f_2 S^2$: the concept of a product of functions (leading to that of a "product of actions") is thus a consequence of sequentiality. The two views coincide if the environment in which the executor's actions take place is defined without ambiguity, and if there are no side effects between the actions arising from the understanding of f_1 and those arising from f_2.

The function $f_1 \dots f_n S^n$ leads to an analogous analysis, but moreover it brings out the fact that sequentiality does not impose itself — either as a primary principle or as a consequence — since distinct executors can take charge of the actions to be associated with each of the functions $f_1 \dots f_{n-1}$. A similar, or comparable, remark applies to the conditional function. If we consider a simple recursive function $\chi\varphi(\chi)$REC, as defined in [7], the executor apprehends it as a whole, and the first directive it receives is to execute $\varphi(\chi)$: we are brought back to the preceding cases for as long as it does not have to execute the function χ contained in $\varphi(\chi)$. When it must execute χ, it is instructed to substitute for it the function $\chi\varphi(\chi)$REC itself, and consequently to continue execution as we have just explained.

χ thus plays the role of a marker within the body $\varphi(\chi)$, as soon as the function submitted to the executor is the function $\chi\varphi(\chi)$REC — a role made explicit by χ occupying the leftmost position in every function formed with the binary operator REC.

It is clear that the concept of a control structure does not impose itself as a first principle, but we are free to act otherwise, and it is in this way that the parallel development of computers and programming languages took place. We are thus led to think that progress might perhaps be made if the choice of a control structure does not precede all conceptual analysis. This is the approach we followed in [7], and it characterizes the present note as well.

Section 2 defines an extension of the algebra of functions; Section 3 applies it to loop-functions and, following J. Arsac [1], restricts the so-called packaging operator introduced in Section 2 to these functions alone. Section 4 offers a brief conclusion.

2. Extension of the algebra of functions

2.1.

In this note $\mathcal{A}$ denotes the formal algebra of functions whose, according to the definition (§ 2.1 of [7]), set of operators is:

$$S = \{\text{EMP}, \text{COND}, \text{REC}, S^0, S^1, \dots\}$$

it contains a new operator EMP, defined below, and whose set F_0 has as subsets

$$\{f_i^* \mid \forall i \in N\} \qquad \text{and} \qquad \{f_i^{**} \mid \forall i \in N\},$$

defined in § 2.3 further below.

2.2. The packaging operator

It is denoted by the proper name (symbol) EMP.

Definition: Let $f \in \mathcal{A}$; then $f\text{EMP} \in \hat{\mathcal{A}}$.

fEMP is called a *packaged function.*

f is the *body* of fEMP.

From a computing standpoint it is more natural to represent the function fEMP by $\{f\}$, it being agreed that in $\mathcal{A}$ braces will be reserved for this use.

Examples: let $g, f \in \mathcal{A}$; $\{f\} \in \mathcal{A}$; $\underbrace{\{\ldots\{f\}\ldots\}}_{i \text{ times, } i \text{ times}} \in \mathcal{A}$; $\{f\}\{g\}S^2 \in \mathcal{A}$.

The interpretation of packaging is tied to the interpretation of the functions f_i^* and f_i^{**}, introduced in § 2.3 below, and reciprocally.

Definitions: f is at depth 0 in the functions $f_1 \ldots f \ldots f_n S^n$, $f_1 \ldots f_{n-1}, \ldots, fS^n$, $ff_1f_2\text{COND}$, and $f_1f_2f\text{COND}$.

An occurrence of a function is at depth 1 in $\{\varphi\}$ if it is at depth 0 in the function φ, and in particular f is at depth 1 in $\{f\}$.

An occurrence of a function is at depth $i+1$ in $\{\varphi\}$ if it is at depth i in φ.

Examples: f_1, f_2, f_3 are at depth 0 in the functions $f_1f_2f_3S^3$ and $f_1f_2f_3\text{COND}$, and at depth 2 in $\{\{f_1f_2f_3S^3\}\}$.

The function $\{f_2\}$ is at depth 0 in $f_1\{f_2\}f_3S^3$ and at depth 1 in $\{f_1\{f_2\}f_3S^3\}$, whereas f_2 itself is at depth 2 there.

Property: *An occurrence of a function at depth i within a function that is itself at depth j within a function φ is at depth $i+j$ in φ.*

Proof: If the occurrence of f under consideration is at depth i in g, then by definition of depth, this occurrence of f lies within a function formed by applying the packaging operator i times.

Since g is at depth j in φ, for the same reason all its occurrences lie within a function φ formed by applying the operator EMP j times, among other operators of S; consequently f has all its occurrences in the function φ formed with $i+j$ applications of the operator EMP, from which the property follows.

Example: f_2 is at depth 2 in $\{f_1\{f_2\}S^2\}$, which is itself at depth 1 in $\{\{f_1\{f_2\}S^2\}f_3f_0S^3\}$; hence f_2 is at depth 3 in this last function.

2.3. Functions f_i^* and f_i^{**}, indexed by i

As is natural in $\mathcal{A}$, the functions of F_0 are defined by their interpretation; the same therefore holds for the particular functions f_i^*, f_i^{**}.

Definitions: 1° Every occurrence of the function f_i^*, or of the function f_i^{**}, at depth j within a function φ is *free* there if $i \leq j$, and *bound* if $i > j$;

2° a function is *well-formed* if and only if all occurrences of the functions f_i^*, f_i^{**} that it may contain are free.

Examples: f_1^* is bound in $pf_1^*f\text{COND}$ $(i=1, j=0)$;

f_1^* is at depth 1 in $\{f_1^*\}$, and this in turn is at depth 0 in $p\{f_1^*\}\text{COND}$, so f_1^* is at depth 1 in the latter; $i=1$, $j=1$, so f_1^* is free there;

f_2^{**} is bound in $\{pf_2^{**}f\text{COND}\}$ and free in $\{\{pf_2^{**}f\text{COND}\}f'S^2\}$ $(i=2, j=2)$;

f_1^* is free in $\{p\{f_1^*\}ff_1^*S^2\text{COND}f'S^2\}$; this function is well-formed.

Interpretation: In every interpretation of the atoms of $\mathcal{A}$, every free occurrence of f_i^*, $i \in N$, is understood by the executor as instructing it to consider as *complete* the interpretation of the packaged function in which the occurrence of f_i^* is at depth i.

Every free occurrence of f_i^{**}, $i \in N$, is understood by the executor as instructing it to *substitute for* f_i^{**} the packaged function in which the occurrence of f_i^{**} is at depth i.

Examples: According to the interpretation defined above, we have the equivalence

$$\{pf_1ff_1^*S^2\mathrm{COND}f'S^2\} \equiv pf_1f'S^2f\mathrm{COND}.$$

Let $\{f_1p\pi_0f_2f_1^{**}S^2\mathrm{COND}S^2\}$, or, using the alternative way of expressing COND (see § 2.8 of [7]):

$$\{f_1(p \to \pi_0, f_2f_1^{**}S^2)S^2\},$$

the interpretation of f_1^{**} leads to the equivalence:

$$\{f_1(p \to \pi_0, f_2f_1^{**}S^2)S^2\} \equiv \chi f_1(p \to \pi_0, f_2\chi S^2)S^2\mathrm{REC}.$$

Remark: In what follows, $pf_1f_2\mathrm{COND}$ will always be represented by $(p \to f_1, f_2)$.

Important observation: The interpretation of f_i^* gives no meaning to a packaged function in which f_i^* is contained via a function of the form $f_1 \dots f^* \dots f_nfS^{n+1}$, in which f_i^* occupies rank j. Such a function acquires meaning — were it useful to do so — (or becomes interpretable) only if the executor is of the sequential type, the most natural instance of which explores the function under consideration from left to right.

In what follows, a function will be well-formed if and only if it satisfies Definition 2 above and has no occurrence of f_i^* in the body of a function formed with S^n except in terminal position (in which case[2] $f_1 \dots f_nf_i^*S^{n+1} \equiv f_1 \dots f_n\pi_0S^{n+1}f_i^*S^2$ by a property of $\mathcal{A}$).

We supplement the definitions of initial and terminal functions in § 3 of [7] with:

Definition: Every function initial (terminal) in f is initial (terminal) in $\{f\}$.

Consequence: f is terminal in $\{f_1f_2\{\varphi(f)\}S^3\}$ and in $(p \to f_1, \{\varphi(f)\})$ if f is terminal in $\varphi(f)$.

2.4. Interpretation of packaging

The proposed interpretation encompasses that of the functions f_i^* and f_i^{**}. x denotes an object of B or B^p, $p \in N$.

The function ν used in points 2 and 3 below is the valuation function of [7]:

1° $[x]\{f\} = [x]f$ if all occurrences of the functions f_i^* and f_i^{**} in f are at a depth at least equal to $i+1$, $(i = 1, 2, \dots)$.

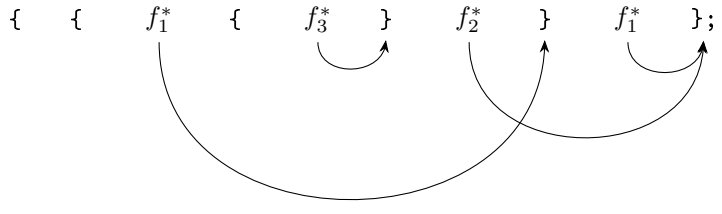
2° $[x]\{f\} = [x]f$ if $[[x]f]\nu$ does not depend on any function f_i^{**} at depth i in f; otherwise

$$[x]\{f\} = [x]fS^{f_i^{**}}_{\{f\}}$$

3° $[x]\{f\} = [x]f$ if $[[x]f]\nu$ does not depend on any function f_i^* at depth i in f; otherwise the application of ν is halted in accordance with the interpretation of f_i^*.

4° The above rules are applied as many times as necessary within the body f of $\{f\}$ if f contains packaged functions having free occurrences of f_i^* and f_i^{**}.

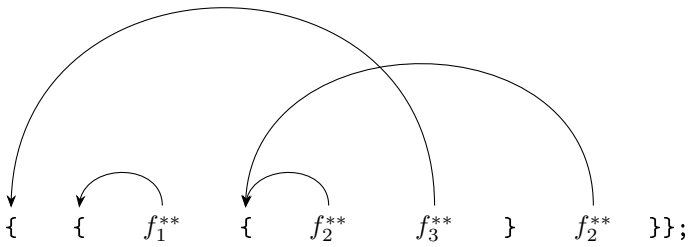
Remark: Let a well-formed function be represented schematically as below, and suppose the executor to be of the sequential type; the interpretation of f_i^* is then represented by the branches indicated by the arrows (illustrative reconstruction of the concept — see translator's note):



[2] Recall that π_0 is the identity function, $\pi_0 \in \mathcal{A}$.

These branches (which it is preferable to call *bifurcations*) are expressed by EXIT instructions, as was proposed among others by Bochmann [3]; J. Arsac [1] gives them the names $\omega[1]$, $\omega[2]$, ... The set of functions f_i^* generalizes the function f^* introduced earlier by J. Kott [6] and by ourselves.

In the same way, the sequential interpretation of f_i^{**} leads to the diagram below when the functions f_i^{**} are all terminal (again an illustrative reconstruction):



If each opening brace is equipped with a label I (as in § 4.2 of [7]), the label is then that of the corresponding packaged function; the function f_i^{**} at depth i within $I\{\dots\}$ is the semantics, within $\mathcal{A}$, of the instruction GOTO I.

2.5. Axioms of $\mathcal{A}$

To the axioms proposed in [7] we must add those concerning the interpretation of EMP and of the functions f_i^* and f_i^{**} (they will be referenced $a_1, a_2, \dots$). The preceding discussion leads naturally to:

a_1, $\{f\} \equiv f$ if and only if all occurrences of the functions f_i^{**} are at a depth at least $i+1$ in $\{f\}$ and all occurrences of f_i^* are free, or if f has no occurrence of these functions;

a_2, for every function f and $\forall i \in N$, $f_i^* f S^2 \equiv f_i^*$;

a_3, every terminal occurrence of π_0 (respectively f_i^*) at depth i within a packaged function may be replaced by f_i^* (respectively π_0) (an equivalent function is obtained);

a_4, if $f_1 \preccurlyeq f_2$ then $\{f_1\} \preccurlyeq \{f_2\}$.

Remark: The calculation rules and properties of $\mathcal{A}$ apply inside $\{\dots\}$; we shall have to extend them whenever the packaging operator is involved.

2° Since the second part of point 2 of the interpretation of packaging is close to the interpretation of the simple recursion function[3] $\chi\varphi(\chi)$REC (§ 4.1 of [7]), we immediately obtain[4]

$$\chi\varphi(\chi)\text{REC} \equiv \{\varphi(\chi) S^{\chi}_{f_1^{**}}\}. \tag{1}$$

The general case will lead us to axiom a_5; but first let us recall:

Definition: 1° Every simple recursion function is a (general) recursion function;

2° let a function be represented by $\varphi(\chi, \xi_1, \dots, \xi_r)$ in which the functions $\xi_1, \dots, \xi_r$ are recursion functions, all or some of which possess occurrences of χ; then

$$\chi\varphi(\chi, \xi_1, \dots, \xi_r)\text{REC}$$

is a recursion function;

3° there are no other rules for the definition of recursion functions.

Axiom A_{14} of [7] reads:

[3] $\chi\varphi(\chi)$REC is a simple recursion function if $\varphi(\chi)$ has no occurrence of REC.

[4] $S^f_{f'}$ is the substitution operator; let $g \in \mathcal{A}$; $gS^f_{f'} \in \mathcal{A}$, $gS^f_{f'}$ is obtained from g by substituting the function f' for every occurrence of f in g.

Axiom A_{14}:

$$\chi\varphi(\chi,\xi_1,\ldots,\xi_r)\mathrm{REC} \equiv \varphi(\chi,\xi_1,\ldots,\xi_r)S^{\chi}_{\chi\varphi(\chi)\mathrm{REC}}$$

where all or some of $\xi_1,\ldots,\xi_r$ may possess occurrences of χ; one proceeds likewise for the functions $\xi_1,\ldots,\xi_r$.

Definition: f is at depth 1 in $\chi\varphi(\chi)$REC if it is at depth 0 in φ; f is at depth j there if it is at depth $j-1$ in φ, that is, at depth $j-1$ in one of the functions $\xi_1,\ldots,\xi_r$.

Definition: Every function initial (terminal) in φ is initial (terminal) in $\chi\varphi$REC.

The semantics of relation (1) is axiom a_5:

Axiom a_5: *let there be a recursion function with body $\varphi(\chi,\xi_1,\ldots,\xi_r)$, this notation recalling that $\xi_1,\ldots,\xi_r$ are occurrences of recursion functions; then:*

$$\chi\varphi(\chi)\mathrm{REC} \equiv \{\varphi'\},$$

where φ' is obtained from φ by replacing every occurrence of χ at depth $i-1$ in φ by f_i^{**}, and proceeding in the same way for the functions $\xi_1,\ldots,\xi_r$, until no REC *operator remains in φ'.*

2.6. Factorization of a function

Property 1: *Let φ be a function whose terminal functions are $f,f_1,\ldots,f_n$ (the order is not significant), among which we choose f; then:*

$$\varphi \equiv \{\varphi' f S^2\},$$

where φ' is obtained from φ by substituting:

— *the function π_0 for all terminal occurrences of f (and simplifying the function so obtained),*

— *the functions $f_1f_1^*S^2,\ldots,f_nf_1^*S^2$ for the terminal occurrences of $f_1,\ldots,f_n$ respectively.*

Note: $f,f_1,\ldots,f_n$ may of course be packaged functions.

Proof: By Property 4 (§ 3.2 of [7]):

$$\varphi' f S^2 \equiv \varphi' S^{\pi_0}_{\pi_0 f S^2} S^{f_1^*}_{f_1^* f S^2} \equiv \varphi' S^{\pi_0}_f \qquad \text{(axioms } A_{12} \text{ and } a_2\text{)}.$$

Now, by Property 1 [of § 3.1 of [7]]:

$$\varphi' \equiv \varphi(f,f_1,\ldots,f_n)S^f_{\pi_0}S^{f_1}_{f_1f_1^*S^2}\cdots S^{f_n}_{f_nf_1^*S^2},$$

$$\varphi' S^{\pi_0}_f \equiv \varphi(f,f_1f_1^*S^2,\ldots,f_nf_1^*S^2),$$

whence

$$\{\varphi' f S^2\} \equiv \{\varphi(f,f_1f_1^*S^2,\ldots)\}$$

$$\equiv \{\varphi(f,f_1,\ldots,f_n)\} \qquad \text{(axioms } a_3 \text{ and } A_{12}\text{)}$$

$$\equiv \varphi(f,f_1,\ldots,f_n) \qquad \text{(axiom } a_1\text{)}.$$

Example: Let $\varphi \equiv (p \to f_1f_2fS^3, g_1g_2S^2)$; its terminal functions are f and g_2.

Factorizing with f according to the property above, we obtain:

$$(p \to f_1f_2fS^3, g_1g_2S^2) \equiv \{(p \to f_1f_2\pi_0S^3, g_1g_2S^2f_1^*S^2)fS^2\}.$$

Property 2: *Let φ be a function all of whose occurrences of f_i^{**} are at depth at least $i+1$; then:*

$$f\{\varphi\}S^2 \equiv \{\varphi'\}$$

where φ' is obtained from φ by replacing its initial functions with their product having f as left factor.

The proof follows directly from a_1 and Property 2 (§ 3.1 of [7]).

Property 3: $\{\varphi\}fS^2 \equiv \{\varphi' fS^2\}$ *where φ' is obtained from φ by substituting, for every free occurrence of f_i^* at depth i, the product $ff_i^*S^2$.*

Proof: Let $f_1, \ldots, f_n$ be the terminal functions of φ; by construction of φ', these are also the terminal functions of φ' (if f_i^* is terminal, axiom a_3 is applied). Those of $\varphi' fS^2$ are therefore, by Property 4 (§ 3.2 of [7]), $f_1 fS^2, \ldots, f_n fS^2$. Now, every occurrence of f_i^* at depth i in φ causes the valuation of φ to halt in order to continue with that of f as a right factor; the valuation of $\{\varphi' fS^2\}$ ends with $ff_i^*S^2$ if f_i^* intervenes according to the interpretation of this function — which completes the proof.

Example:

$$\{(p \to f_1^*, g_1)(q \to f_1, f_2)S^2\}fS^2 \equiv \{(p \to ff_1^*S^2, g_1)(q \to f_1, f_2)S^2 fS^2\}.$$

3. Loop-functions

3.1.

Loop-functions form a subset of recursion functions; they are defined as follows:

Definition 1: 1° A *simple loop-function* is a simple recursion function with body $\varphi(\chi)$ in which all occurrences of χ are terminal;

2° the recursion function with body $\varphi(\chi, \xi_1, \ldots, \xi_r)$, where $\xi_1, \ldots, \xi_r$ are loop-functions, is a *loop-function* if the occurrences of χ are terminal in φ.

It follows that the occurrences of χ in $\xi_1, \ldots, \xi_r$ are terminal functions, and that these functions are terminal in φ.

Consequence: Axiom a_5 leads to:

$$\chi\varphi(\chi, \xi_1, \ldots, \xi_r)\text{REC} \equiv \{\varphi'\}$$

φ' being the function derived from φ according to axiom a_5, with all occurrences of f_i^{**} in φ' terminal; hence:

Property 1: *Every loop-function is equivalent to a packaged function in which all the functions f_i^{**} are terminal.*

Example:

$$\chi(p_1 \to f_1\chi S^2, \chi_1(p_2 \to f_2\chi S^2, (p_3 \to f_3\chi_1 S^2, f_4))\text{REC})\text{REC}$$

is a loop-function. By a_5 it is equivalent to:

$$\{(p_1 \to f_1 f_1^{**}S^2, \{(p_2 \to f_2 f_2^{**}S^2, (p_3 \to f_3 f_1^{**}S^2, f_4))\})\}.$$

We note that it is derived directly from the labeled form proposed earlier (§ 4.2 of [7]).

Let us now restrict, following J. Arsac [1, 2], the use of packaging to the construction of loop-functions; consequently the same restriction applies to the functions f_i^* and f_i^{**}.

It follows from Property 1 above that all occurrences of packaged functions within a packaged function are terminal.

Let us begin with simple loop-functions.

3.2. Simple loop-functions

Let $g_1\chi S^2, \ldots, g_k\chi S^2, f_1, \ldots, f_n$ be the terminal functions of the body denoted $\varphi(g_1\chi S^2, \ldots, g_k\chi S^2, \ldots, f_n)$ of a simple loop-function, according to the definition recalled at the beginning of § 3.1. By Property 1 (§ 2.6) we have

$$\varphi \equiv \{\varphi'(g_1, \ldots, g_k, f_1 f_1^* S^2, \ldots, f_n f_1^* S^2)\chi S^2\}.$$

Whence

$$\chi\varphi(\chi)\mathrm{REC} \equiv \chi\{\varphi'\chi S^2\}\mathrm{REC}$$

and, by axiom a_5:

$$\chi\varphi(\chi)\mathrm{REC} \equiv \{\varphi' f_1^{**} S^2\}. \tag{1}$$

Using axiom a_5 first instead, giving

$$\chi\varphi(\chi)\mathrm{REC} \equiv \{\varphi'(g_1 f_1^{**} S^2, \ldots, g_k f_1^{**} S^2, f_1 f_1^{*} S^2, \ldots, f_n f_1^{*} S^2)\} \tag{2}$$

we deduce from this that Property 5 (§ 3.2 of [7]) is valid, taking axiom a_2 into account.

Relation (2) shows that either of the following rules may be adopted.

Rule 1: f_1^{**} is implicit; then, within the body of a packaged function, the terminal functions $g_1, \ldots, g_k$ are interpreted as $g_1 f_1^{**} S^2, \ldots, g_k f_1^{**} S^2$.

Rule 2: f_1^{*} is implicit; then, within the body of a packaged function, the terminal functions $f_1, \ldots, f_n$ are interpreted as $f_1 f_1^{*} S^2, \ldots, f_n f_1^{*} S^2$.

It should be noted that the second rule expresses axiom a_3 (§ 2.4). The rule applied to (1) leads to the same result as when applied to (2), whereas it makes no sense to apply Rule 2 to (1), since the property used to establish (1) expresses a property of f_1^{*}.

In a programming language where the symbols BEGIN and END are not reserved for constructing packaged instruction sequences, a symbol — REPEAT, for example — can announce that BEGIN and END play the roles of { and } respectively, so that the instruction

REPEAT BEGIN S END

has as its semantics the packaged function $\{f_S\}$, f_S being the function describing the semantics of S. The instruction

LOOP S REPEAT

would be equivalent, as would any other analogous instruction, and $\{S\}$, which is simpler, as proposed by J. Arsac.

Examples: 1° The primitive loop-function with body $(p \to f\chi S^2, \pi_0)$ (§ 5.1 of [7]):

$$\chi(p \to f\chi S^2, \pi_0)\mathrm{REC} \equiv \{(p \to f, f_1^{*})\} \qquad \text{(Rule 1)}$$

$$\equiv \{(p \to f f_1^{**} S^2, \pi_0)\} \qquad \text{(Rule 2);}$$

2° the loop-function with body $f_1(p \to \pi_0, f_2\chi S^2)S^2$, which is the semantics of Dijkstra's $n + (1/2)$ program schema:

$$\chi f_1(p \to \pi_0, f_2\chi S^2)S^2\mathrm{REC} \equiv \{f_1(p \to f_1^{*}, f_2)S^2\} \qquad \text{(Rule 1)}$$

$$\equiv \{f_1(p \to \pi_0, f_2 f_1^{**} S^2)S^2\} \qquad \text{(Rule 2).}$$

We have

$$\{f_1(p \to f_1^{*}, f_2)S^2\} \equiv \{f_1(p \to f_1^{*}, \pi_0)S^2 f_2 S^2\},$$

this last function describing the semantics of the instruction proposed by N. Wirth [9]:

REPEAT BEGIN S_{f_1}; WHEN p : EXIT; S_{f_2} END

and of the instruction proposed by O. Dahl [4]:

LOOP S_{f_1}; WHILE $\neg p$: S_{f_2}; REPEAT

the instruction sequences S_{f_1}, S_{f_2} having f_1, f_2 as their respective semantics.

Note that the first could be expressed as

REPEAT BEGIN S_{f_1}; IF p THEN EXIT ELSE S_{f_2} END

and the second as

LOOP S_{f_1}; IF $\neg p$ THEN EXIT; S_{f_2}; REPEAT.

3.3. General loop-functions

By axiom a_5, all the functions f_i^{**} in the body of the packaged function $\{\varphi'\}$ equivalent to a loop-function are terminal; and the same holds for the loop-functions $\xi_1, \ldots, \xi_r$ referred to in axiom a_5, so that all packaged functions within φ' are terminal. The terminal functions other than f_i^{**} in $\{f\}$ at depth j in φ' are therefore terminal in φ'. Consequently, applying axiom a_3, we may substitute for them their product with f_{j+1}^{*} as a terminal factor.

The function $\{f'\}$ thus obtained has, as its only terminal functions, f_i^{*} and f_i^{**}, $i = 1, 2, \ldots$.

Consider such a function $\{f'\}$ at depth 0 in φ, hence at depth 1 in $\{\varphi\}$. Let $\{f''\}$ be the function obtained by substituting, for every occurrence of f_i^{**} at depth $i-1$ in f', the function f_{i-1}^{*}.

It is immediate that:

$$\{\varphi(\{f''\}f_1^{**}S^2)\} \equiv \{\varphi\}. \tag{1}$$

We proceed in the same way at the level of every occurrence of a packaged function in φ, until the functions f_i^{**} disappear from the body of each of them — they cease to be terminal; only the function f_1^{**} occurring in relation (1) remains terminal.

We thus have:

Property 4: *Every loop-function is equivalent to a packaged function containing occurrences of packaged functions as a left factor with the single function f_1^{**} alone; the terminal functions of the body of each of them are the functions f_i^{*}.*

Example:[5] 1° Let the loop-function be:

$$(2)\quad \{(q_1 \to \{(p_1 \to f_1 f_2^{**}S^2, (p_2 \to f_2 f_1^{**}S^2, f_3))\}, (q_2 \to h_1 f_1^{**}S^2, h_2))\}$$

we obtain, first (construction of the functions $\{f'\}$ following the exposition leading to Property 4):

$$\{(q_1 \to \{(p_1 \to f_1 f_2^{**}S^2, (p_2 \to f_2 f_1^{**}S^2, f_3 f_2^{*}S^2))\}, (q_2 \to h_1 f_1^{**}S^2, h_2 f_1^{*}S^2))\}$$

and second [by (1) above]:

$$\{(q_1 \to \{(p_1 \to f_1 f_1^{*}S^2, (p_2 \to f_2 f_2^{**}S^2, f_3 f_2^{*}S^2))\}f_1^{**}S^2, (q_2 \to h_1 f_1^{**}S^2, h_2 f_1^{*}S^2))\}.$$

Rule 1 is applicable, whence

$$\{(q_1 \to \{(p_1 \to f_1 f_1^{*}S^2, (p_2 \to f_2, f_3 f_2^{*}S^2))\}, (q_2 \to h_1, h_2 f_1^{*}S^2))\}$$

whereas Rule 2 is not.

2° Let us replace, in the function (2) above, the function f_3 by

$$\{(p_3 \to f_3 f_3^{**}S^2, (p_4 \to f_4 f_2^{**}S^2, f_5))\},$$

it is at depth 2 in (2). It is replaced by

$$\{(p_3 \to f_3 f_3^{**}S^2, (p_4 \to f_4 f_2^{**}S^2, f_5 f_3^{*}S^2))\}$$

then by

$$\{(p_3 \to f_3 f_2^{*}S^2, (p_4 \to f_4 f_1^{*}S^2, f_5 f_3^{*}S^2))\}f_1^{**}S^2,$$

whence the function

$$\{(q_1 \to \{(p_1 \to f_1 f_1^{*}S^2, (p_2 \to f_2 f_1^{**}S^2,$$
$$\{(p_3 \to f_3 f_2^{*}S^2, (p_4 \to f_4 f_1^{*}S^2, f_5 f_3^{*}S^2))\}f_1^{**}S^2))\}f_1^{**}S^2,$$

[5] The two worked examples that follow (1° and 2°) correspond to passages of the source scan that were heavily corrupted by OCR errors. The formulas given here are a best-effort reconstruction, internally consistent with the surrounding pattern of indices and rules, but their exact subscripts could not be independently verified letter-for-letter against the original typesetting.

$$(q_2 \to h_1 f_1^{**} S^2, h_2 f_1^* S^2))\}.$$

Rule 1 leads to the removal of every occurrence of f_1^{**}. Property 4 has an immediate corollary.

Property 5: *If f_1^{**} is implicit (Rule 1 of § 3.2), the packaging operator together with the set of functions (for exiting packaging) f_i^* suffice to construct the set of loop-functions.*

Axiom a_3, recalling that every packaged function is terminal within every loop-function at every depth, leads for its part to:

Property: *If f_i^* is implicit (Rule 2 of § 3.2), the packaging operator together with the set of functions (for backward branches) f_i^{**} suffice to construct the set of loop-functions.*

Observe that in this case recourse to the exit functions is not only implicit but unnecessary, on account of axiom a_3.

4. Conclusion

To the extent that the proposed definition of recursion functions might be regarded as introducing a "sort of label," attached to χ in $\chi\varphi(\chi)$REC, and by restricting ourselves to loop-functions, we have proposed an extension of the algebra of functions demonstrating that the label-free packaging of a function is possible by introducing functions expressing the semantics of an instruction for exiting a packaging, either (Rule 1) downstream, or (Rule 2) upstream (under the hypothesis of a sequential executor). The choice is arbitrary, but it is not obvious that the suppression of labels always makes programs clearer and more understandable, or that it necessarily facilitates their systematic construction.

The first choice (Rule 1) corresponds to the path proposed by G. Bochmann [3] and J. Arsac [1], studied by J. Arsac *et al.* [2], S. R. Kosaraju [5], Ruggiu [8], among others.

The second choice (Rule 2) appears to us equally natural, in particular if we imagine the realization of the concept of a sequential executor at the microprogramming level; but this is another subject, and it is not the purpose of this note to address it.

References

[1] J. ARSAC, *Les langages sans étiquettes*, Publication 73/13, Institut de Programmation, Université de Paris VI.

[2] J. ARSAC, L. NOLIN, G. RUGGIU et J.P. VASSEUR, *Le système de programmation structurée EXEL*. Revue Technique THOMSON CSF, vol. 6, n° 3, septembre 1974, p. 715 à 736.

[3] G. V. BOCHMANN, *Multiple Exits From a loop without the GOTO*, Comm. A.C.M., vol. 16, n° 7, juillet 1973, p. 443-444.

[4] O. DAHL, E. DIJKSTRA et C. HOARE, *Structured programming*, Academic Press, London, 1972. Voir aussi D. KNUTH, *Structured programming with "goto" Statements*, Computing Surveys, Vol. 6, n° 4, décembre 1974.

[5] S. R. KOSARAJU, *Analysis of structures Programs*, Journal of computer and systems Sciences, vol. 9, n° 3, décembre 1974.

[6] J. KOTT, *Remarques sur la structure des schémas de programme*, in Théorie des automates, des langages et de la programmation, Colloque I.R.I.A./1972, p. 191-194.

[7] F. H. RAYMOND, *Note sur l'algèbre des fonctions*, R.A.I.R.O., n° R-3, 1975, p. 25-49.

[8] G. RUGGIU, *De l'organigramme à la formule*, Thèse d'État, Université Paris VI, 1974.

[9] N. WIRTH, *On certain basis concepts of programming languages*, Stanford Computer Sciences, Report CS 65, Stanford, Calif., May 1967.