

ANDRÉ ARNOLD

Sémantique algébrique de l'appel par valeur

RAIRO. Informatique théorique, tome 12, n° 2 (1978), p. 69-82

http://www.numdam.org/item?id=ITA_1978__12_2_69_0

© AFCET, 1978, tous droits réservés.

L'accès aux archives de la revue « RAIRO. Informatique théorique » implique l'accord avec les conditions générales d'utilisation (<http://www.numdam.org/conditions>). Toute utilisation commerciale ou impression systématique est constitutive d'une infraction pénale. Toute copie ou impression de ce fichier doit contenir la présente mention de copyright.

NUMDAM

Article numérisé dans le cadre du programme
Numérisation de documents anciens mathématiques

<http://www.numdam.org/>

SÉMANTIQUE ALGÈBRIQUE DE L'APPEL PAR VALEUR (*)

par André ARNOLD ⁽¹⁾

Communiqué par M. NIVAT

Résumé. — A tout schéma de programme récursif S dont certains paramètres sont appelés par valeur, on associe un schéma S' tel que pour toute interprétation I , la fonction calculée par S est l'image de l'arbre infini $T(S')$ engendré par la grammaire schématique associée à S' . La transformation qui fait passer de S à S' , analogue à celle d'Astesiano et Costa, peut être généralisée afin de caractériser de la même façon les fonctions calculées par les schémas génériques de B. Lang.

INTRODUCTION

Si on affecte des constantes aux variables d'entrée d'un programme P et qu'on exécute ce programme sur une machine M , on obtient un résultat, ou on n'obtient rien. On peut donc associer au couple (P, M) une fonction partielle qui est par définition la fonction calculée par le programme P sur la machine M . Ceci a donné naissance, par abstraction, à la notion de *sémantique opérationnelle* : un programme P est un couple (S, I) formé d'un schéma de programme éventuellement récursif et d'une interprétation des symboles de base du schéma relativement à un domaine D représentant les constantes-machine; la machine M devient alors un ensemble de règles R indiquant comment le programme doit être utilisé pour transformer les données en résultat. On peut définir alors la fonction notée $\text{Valop}_R(S, I)$ calculée par ce programme [4, 16]. En pratique on peut considérer que tout "interpréteur" d'un langage de programmation est un tel ensemble de règles.

De façon encore plus abstraite on peut associer à un programme (S, I) une application fonctionnelle \hat{S}_I de l'ensemble des fonctions partielles dans lui-même. Cette fonctionnelle admet un plus petit point fixe Valden (S, I) . Cette façon de définir la fonction calculée par un programme est appelée *sémantique dénotationnelle* [12, 15]. Enfin on peut considérer les définitions récursives de procédures figurant dans un schéma S comme l'ensemble des

(*) Reçu septembre 1977, révisé janvier 1978.

(1) Laboratoire d'Informatique; Université de Lille I, Villeneuve d'Ascq.

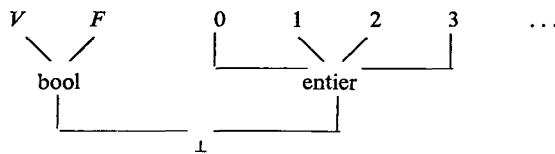
règles d'une grammaire d'arbres qui engendre un arbre infini $T(S)$; pour chaque interprétation I on note $\text{Valalg}(S, I)$ l'image de $T(S)$ par I . Cette façon d'associer une fonction au programme (S, I) est appelée *sémantique algébrique* [13].

Si la sémantique algébrique et la sémantique dénotationnelle coïncident en ce sens que pour toute interprétation « raisonnable » I , $\text{Valalg}(S, I)$ est égal à $\text{Valden}(S, I)$ [13], il n'en est pas de même pour la sémantique opérationnelle : on sait seulement que $\text{Valop}_R(S, I)$ est « moins définie » que $\text{Valden}(S, I)$ [4]. Mais comme $\text{Valop}_R(S, I)$ dépend de R il se peut que pour certains R on ait l'égalité. C'est le cas en particulier quand R est défini de façon à formaliser l'appel par nom [4, 16]. Autrement dit les sémantiques dénotationnelle et algébrique permettent bien de définir la fonction calculée par un programme récursif lorsque la règle de passage des paramètres est l'appel par nom.

Mais dans la pratique courante de la programmation, c'est le plus souvent la règle de passage des paramètres par appel par valeur qui est utilisée. Aussi la sémantique de l'appel par valeur a-t-elle fait l'objet de nombreuses études [1, 2, 3, 4, 6, 8, 9, 10, 12, 14].

Alors que dans le cas de l'appel par nom le paramètre effectif est simplement substitué au paramètre formel, dans le cas de l'appel par valeur, ce paramètre effectif est d'abord calculé et c'est sa valeur qui est substituée au paramètre formel. La différence entre appel par valeur et appel par nom apparaît donc seulement quand le calcul de l'argument appelé par valeur ne se termine pas. Boudol [3] et de Bakker [2] ont modifié légèrement la définition de la fonctionnelle \hat{S}_I associée à un programme (S, I) de façon à tenir compte de cette différence entre appel par nom et appel par valeur et ont pu définir une sémantique dénotationnelle de ces programmes lorsque l'interprétation I est discrète, c'est-à-dire que son domaine est un ensemble auquel on a ajouté un élément noté \perp qui est la valeur conventionnellement attribuée à toute expression dont le calcul est infini, ou donne un résultat indéfini. On peut donc considérer que lorsqu'un argument est appelé par valeur, on teste d'abord si sa valeur est ou non l'élément \perp . C'est le point de vue adopté par Manna et Vuillemin [12] et également par Astesiano et Costa [1]. Ces derniers ont étendu la définition de la fonctionnelle \hat{S}_I associée au programme (S, I) avec appel par valeur au cas où I est une interprétation dans un domaine non discret, c'est-à-dire un domaine dont les éléments sont des valeurs qui peuvent être plus ou moins définies.

Dans le cas des domaines non discrets la condition « $x \neq \perp$ » peut être remplacée par « $x \geq d$ » où d est un élément fixé du domaine d'interprétation, ce qui permettrait de formaliser une notion d'appel par valeur où l'argument appelé par valeur ne serait pas nécessairement entièrement calculé mais calculé seulement jusqu'à ce qu'on obtienne une certaine information sur cet argument. Plus généralement encore on peut considérer que l'évaluation préalable des arguments d'une procédure détermine le choix du corps de la procédure. Ainsi sur le domaine



la procédure $ADD(x, y)$ peut exécuter l'addition entière ou booléenne selon que les arguments sont tous deux des entiers ou des booléens. Ce sont de telles procédures que Lang [9] a définies et étudiées sous le nom de procédures génériques. Il est clair que les procédures dont certains paramètres sont appelés par valeur sont des cas particuliers de procédures génériques. Lang a montré l'équivalence de la sémantique dénotationnelle et de la sémantique opérationnelle pour les procédures génériques, ce qui généralise les résultats déjà cités de Boudol et de de Bakker.

La question que nous résolvons dans cet article est celle de savoir s'il existe une sémantique algébrique pour l'appel par valeur et, plus généralement, pour les procédures génériques, ou, en d'autres termes, si la fonction calculée par des schémas récursifs avec appel par valeur ou par des schémas génériques sous une certaine interprétation peut être définie comme l'image par cette interprétation d'un arbre infini engendré par une grammaire d'arbres. Nous répondons positivement à cette question en modifiant les schémas considérés par une transformation syntaxique qui consiste à recopier les arguments des procédures de façon à pouvoir les calculer séparément. Dans le cas de l'appel par valeur on retrouve la transformation mise en évidence et utilisée par Astesiano et Costa [1] pour obtenir un résultat analogue dans un cadre formel différent, celui des algèbres dérivées d'algèbres hétérogènes [6, 11].

Il découle du résultat déjà cité de B. Lang que cette sémantique algébrique des schémas génériques coïncide avec leur sémantique opérationnelle telle qu'elle a été définie par Lang.

Dans le cas particulier de l'appel par valeur Astesiano et Costa [1] ont montré que leur sémantique algébrique coïncidait avec la sémantique opérationnelle de la « full value substitution rule » dans le cas des domaines discrets et ont conjecturé que c'était encore vrai dans le cas des domaines non discrets. Or la sémantique opérationnelle de Lang est, dans le cas de l'appel par valeur, exactement celle de la « full value substitution rule »; le résultat de Lang prouve donc cette conjecture [5].

Donnons un exemple simple de la transformation que nous utilisons : soit le schéma $\varphi(x, y) = t(x, y)$ où y est appelé par valeur; nous le transformons en $\varphi(x, y) = \alpha(y, t(x, y))$ où l'interprétation de $\alpha(y, z)$ sera toujours : *si* $y = \perp$ *alors* \perp *sinon* z . C'est donc un test au sens de I. Guessarian [7]. De manière générale, les transformations syntaxiques que nous utilisons consistent à introduire dans le schéma de nouveaux symboles fonctionnels qui seront toujours interprétés comme des tests.

DÉFINITION : On appelle *schéma de programme récursif avec appel par valeur* la donnée d'un schéma

$$S = \left\{ \begin{array}{l} \varphi_i(x_1, \dots, x_{n_i}) = u_i, \\ i = 1, \dots, k \end{array} \right.$$

et pour chaque $i \leq k$ d'une partie $W_i = \{ \sigma_i(1), \dots, \sigma_i(m_i) \}$ de $\{ 1, \dots, n \}$ qui est l'ensemble des indices des variables de φ_i appelées par valeur.

A chaque i nous associons un nouveau symbole α_i d'arité $m_i + 1$ et nous formons le nouveau schéma

$$S^W = \left\{ \begin{array}{l} \varphi_i(x_1, \dots, x_{n_i}) = \alpha_i(x_{\sigma_i(1)}, \dots, x_{\sigma_i(m_i)}, u_i), \\ i = 1, \dots, k, \end{array} \right.$$

L'interprétation du symbole α_i de arité $m_i + 1$ sera toujours la fonction $(\alpha_i)_I$ à $m_i + 1$ arguments définie par

$$(\alpha_i)_I(x_1, \dots, x_{m_i}, y) = \begin{cases} \perp & \text{si } \exists k \leq m_i \text{ tel que } x_k = \perp, \\ y & \text{sinon.} \end{cases}$$

Cette interprétation donnera aux équations du schéma transformé S^W la signification intuitive suivante : $\varphi_i(x_1, \dots, x_{n_i})$ a la même valeur que u_i si tous les arguments appelés par valeur sont différents de \perp et vaut \perp sinon.

REMARQUE : Dans le cas où tous les paramètres sont appelés par nom le schéma

$$S = \left\{ \begin{array}{l} \varphi_i(x_1, \dots, x_{n_i}) = u_i, \\ i = 1, \dots, k, \end{array} \right.$$

est transformé en

$$S^W = \begin{cases} \varphi_i(x_1, \dots, x_n) = \alpha(u_i), \\ i = 1, \dots, k, \end{cases}$$

où l'interprétation de α est l'identité. On a donc $\hat{S}_I^W = \hat{S}_I$ et les deux schémas calculent la même fonction pour toute interprétation.

Dans la mesure où la sémantique algébrique reflète la sémantique opérationnelle, cette transformation a aussi une signification opérationnelle : tout argument appelé par valeur sera recopié et calculé séparément. Dans son étude de la sémantique algébrique de l'instruction d'affectation, L. Kott [8] a également mis en évidence la nécessité de calculer séparément de tels arguments, qu'il appelle variables locales. Mais, alors que la méthode de Kott consiste à évaluer séparément les variables locales puis à réintroduire les valeurs ainsi obtenues dans le « calcul principal », la nôtre consiste à calculer deux fois les variables locales : une fois dans « le calcul principal » comme si ces arguments étaient appelés par nom et une autre fois « à côté ». Nous verrons qu'il n'est pas nécessaire d'obtenir la valeur exacte de l'argument dans ce calcul « à côté » mais qu'il suffit de s'assurer qu'il peut prendre une valeur différente de \perp .

Exemple : Il n'est pas possible de parler d'appel par valeur sans citer le célèbre exemple [12] :

$$\varphi(x, y) = \text{si } x = 0 \text{ alors } 0 \text{ sinon } \varphi(x-1, \varphi(x, y)); \text{ valeur } y.$$

Nous associons à ce programme le schéma

$$\varphi(x, y) = \alpha(y, h(x, \varphi(a(x), \varphi(x, y))))$$

et l'interprétation I définie sur $N \cup \{\perp\}$ par

$$\alpha_I(x, y) = \begin{cases} \perp & \text{si } x = \perp, \\ y & \text{sinon;} \end{cases}$$

$$h_I(x, y) = \begin{cases} \perp & \text{si } x = \perp, \\ 0 & \text{si } x = 0, \\ y & \text{sinon;} \end{cases}$$

$$a_I(x) = \begin{cases} \perp & \text{si } x = \perp, \\ \perp & \text{si } x = 0, \\ x-1 & \text{sinon.} \end{cases}$$

Selon la sémantique dénotationnelle, la fonction calculée par ce programme est la limite de la suite $(f_i)_i$ où

$$f_0(x) = \perp \text{ pour tout } x,$$

$$f_{i+1} = \hat{S}_I(f_i).$$

Par définition $\hat{S}_I(f) = \alpha_I(y, h_I(x, f(a_I(x)), f(x, y))) = \text{si } y = \perp \text{ alors } \perp \text{ sinon si } x = 0 \text{ alors } 0 \text{ sinon } f(a_I(x), f(x, y))$.

On a donc $f_1(x, y) = \text{si } y = \perp \text{ alors } \perp \text{ sinon si } x = 0 \text{ alors } 0 \text{ sinon } \perp = \text{si } y = \perp \text{ ou } x \neq 0 \text{ alors } \perp \text{ sinon } 0$; $f_2(x, y) = \text{si } y = \perp \text{ alors } \perp \text{ sinon si } x = 0 \text{ alors } 0 \text{ sinon } f_1(a_I(x), f_1(x, y))$.

Dans le cas où $y \neq \perp$ et $x \neq 0$ on a $f_1(x, y) = \perp$ et donc

$$f_1(a_I(x), f_1(x, y)) = f_1(a_I(x), \perp) = \perp$$

d'où $f_2(x, y) = \text{si } y = \perp \text{ alors } \perp \text{ sinon si } x = 0 \text{ alors } 0 \text{ sinon } \perp = \text{si } y = \perp \text{ ou } x \neq 0 \text{ alors } \perp \text{ sinon } 0 = f_1(x, y)$.

Cette suite $(f_i)_i$ est donc stationnaire et sa limite est *si* $y = \perp$ ou $x \neq 0$ *alors* \perp *sinon* 0 qui est bien la fonction calculée par le programme considéré.

Comme Astesiano et Costa [1] nous considérons que la définition de \hat{S}^W est la bonne généralisation de la définition de la fonctionnelle associée à un programme récursif avec appel par valeur donnée par Boudol [3] et de Bakker [2] dans le cas des interprétations discrètes. Nous en concluons donc qu'il existe une sémantique algébrique de l'appel par valeur.

Tentons de préciser l'intuition sur laquelle s'appuie une telle affirmation. Si le calcul de $\alpha_i(y_1, \dots, y_m, u_i)$ donne \perp parce que l'un des arguments appelé par valeur n'a pu être calculé et que cette valeur \perp est substituée à y dans $f(x, y)$, alors $f(x, \perp)$ doit nécessairement prendre la valeur \perp puisque l'on n'a pas réussi à calculer le deuxième argument de f , à moins qu'il ne soit pas nécessaire d'avoir calculé ce second argument pour calculer la valeur de $f(x, \perp)$. C'est ce qui se passe par exemple pour le test $h(x, y, z) = \text{si } x = 0 \text{ alors } y \text{ sinon } z$, où, lorsqu'on écrit $h(1, \perp, z) = z$, c'est bien parce que dans ce cas l'argument substitué à y n'a pas besoin d'être calculé. Par contre il se pourrait qu'une fonction dont l'un des arguments est \perp prenne une valeur définie parce que l'on considère que le résultat est indépendant de la valeur affectée à la variable correspondante; c'est le cas si on pose $h(\perp, y, y) = y$, arguant du fait que quelle que soit la branche du test choisie le résultat sera le même. Mais donc ce dernier cas si l'argument substitué

à x dans $h(x, y, y)$ ne peut être calculé, le résultat doit être \perp et non y . Donnons un autre exemple de cette situation : soit la fonction f qui à tout entier n associe la limite de la suite convergente $x_0 = n$, $x_{i+1} = (1/2)(x_i + 2/x_i)$; on aura donc $f(n) = \sqrt{2}$ pour tout n , ce qui n'entraîne pas pour autant $f(\perp) = \sqrt{2}$. Pour éclairer cette situation il faut considérer que l'élément « indéfini » peut avoir deux significations : d'une part il peut s'interpréter comme « n'importe quelle valeur indifféremment », d'autre part comme « l'ensemble vide de valeurs ». C'est uniquement en donnant à \perp cette deuxième signification que notre affirmation prend son sens. Remarquons que les conditions imposées aux interprétations par Boudol et de Bakker dans le cas des domaines plats sont bien des conséquences de cette deuxième signification de \perp .

La première partie de cet article contient les préliminaires et quelques rappels concernant la sémantique algébrique. Dans la deuxième partie nous définissons la sémantique algébrique annoncée de l'appel par valeur. Cette transformation est généralisée dans la troisième partie pour donner une sémantique algébrique aux procédures génériques de Lang. Dans la quatrième partie nous mettons en évidence quelques propriétés des tests généralisés.

1. PRÉLIMINAIRES

Soit D un ensemble ordonné par \leq . On dira qu'une partie E de D est dirigée si $\forall d, d' \in E, \exists d'' \in E$ tel que $d \leq d''$ et $d' \leq d''$. On dira qu'un ensemble ordonné D est inductif (ou complet) si il contient un plus petit élément, noté \perp_D (ou \perp), et si toute partie dirigée E de D admet une borne supérieure, notée $\sqcup E$. Si D est un ensemble ordonné inductif, D^n est aussi un ensemble ordonné inductif pour l'ordre produit \leq^n . On notera parfois \vec{d} un élément de D^n . Une application f de D^n dans D est continue si pour toute partie dirigée E de D^n , l'ensemble $\{f(\vec{d})/\vec{d} \in E\}$ admet une borne supérieure égale à $f(\sqcup E)$; ceci implique que f est croissante et donc que $\{f(\vec{d})/\vec{d} \in E\}$ est un ensemble dirigé.

Soit Σ un alphabet gradué, c'est-à-dire tel qu'à tout symbole f de Σ est associée un entier $d(f)$. Soit $V_n = \{x_1, \dots, x_n\}$ et $V = \bigcup_{n>0} V_n$ des ensembles de variables. Nous noterons $T_\Sigma(E)$ le Σ -magma libre engendré par E , où E est un ensemble quelconque. Soit Ω un symbole 0-aire n'appartenant pas à Σ . On définit sur $T_\Sigma(\{\Omega\} \cup V)$ l'ordre syntaxique $<$ qui est

le plus petit ordre compatible avec la structure de Σ -magma tel que $\Omega \prec t$ pour tout $t \in T_\Sigma(\{\Omega\} \cup V)$.

Soit $\Phi = \{\varphi_1, \dots, \varphi_k\}$ un ensemble de symboles fonctionnels disjoint de $\Sigma \cup \{\Omega\}$, chaque φ_i étant d'arité n_i . Un schéma de programme récursif S est alors un ensemble d'équations

$$\begin{cases} \varphi_i(x_1, \dots, x_{n_i}) = u_i, \\ i = 1, \dots, k, \end{cases}$$

où $u_i \in T_{\Sigma \cup \Phi}(V_{n_i})$.

A ce schéma on associe la grammaire d'arbres \bar{S} dont les règles sont

$$\begin{cases} \varphi_i(x_1, \dots, x_{n_i}) = u_i + \Omega, \\ i = 1, \dots, k, \end{cases}$$

et on note \xRightarrow{S} la dérivation dans \bar{S} . Si $t \in T_{\Sigma \cup \Phi}(\{\Omega\} \cup V)$ on note $L(S, t) = \{t' \in T_\Sigma(\{\Omega\} \cup V) / t \xRightarrow{*}_S t'\}$ et $L_i(S) = L(S, \varphi_i(x_1, \dots, x_{n_i}))$.

THÉORÈME 1 (Nivat [13]) : *Pour tout $t \in T_{\Sigma \cup \Phi}(\{\Omega\} \cup V)$, $L(S, t)$ est un ensemble dirigé.*

Une *interprétation* I d'un schéma S sur un ensemble ordonné inductif D est une application qui à tout f de Σ associe une application continue f_I de $D^{d(f)}$ dans D . A tout terme t de $T_\Sigma(\{\Omega\} \cup V_n)$ et à toute application γ de V_n dans D on associe l'élément $(I, \gamma)t$ de D défini par récurrence par

$$(I, \gamma)\Omega = \perp,$$

$$(I, \gamma)x_i = \gamma(x_i),$$

$$(I, \gamma)f(t_1, \dots, t_n) = f_I((I, \gamma)t_1, \dots, (I, \gamma)t_n).$$

A tout terme t de $T_{\Sigma \cup \Phi}(\{\Omega\} \cup V_n)$, à toute application γ de V_n dans D et à tout k -uple $\vec{g} = \langle g_1, \dots, g_k \rangle$, où g_i est une application continue de D^{n_i} dans D on associe l'élément $(I, \gamma, \vec{g})t$ de D défini par

$$(I, \gamma, \vec{g})\Omega = \perp,$$

$$(I, \gamma, \vec{g})x_i = \gamma(x_i),$$

$$(I, \gamma, \vec{g})f(t_1, \dots, t_n) = f_I((I, \gamma, \vec{g})t_1, \dots, (I, \gamma, \vec{g})t_n),$$

$$(I, \gamma, \vec{g})\varphi_i(t_1, \dots, t_{n_i}) = g_i((I, \gamma, \vec{g})t_1, \dots, (I, \gamma, \vec{g})t_{n_i}).$$

On pose alors $\hat{S}_I(\vec{g}) = \langle g'_1, \dots, g'_k \rangle$ où g'_i est la fonction continue de D^m dans D telle que

$$\forall \vec{d} = \langle d_1, \dots, d_{n_i} \rangle \in D^m, \quad g'_i(\vec{d}) = (I, \gamma_{\vec{d}}, \vec{g})u_i,$$

où $\gamma_{\vec{d}}$ est l'application qui associe d_i à x_i .

THÉORÈME 2 (Nivat [13]) : *Pour toute interprétation I , la fonctionnelle \hat{S}_I admet un plus petit point fixe $\vec{g} = \langle g_1, \dots, g_k \rangle$ qui vérifie*

$$\forall i \leq k, \quad \forall \vec{d} \in D^m, \quad g_i(\vec{d}) = \sqcup \{ (I, \gamma_{\vec{d}})t / t \in L_i(S) \}.$$

2. SÉMANTIQUE DE L'APPEL PAR VALEUR

En reprenant les notations données dans l'introduction (S est un schéma avec appel par valeur, S^W est le schéma transformé) toute interprétation I de S sur un ensemble ordonné inductif D est étendue en une interprétation \bar{I} de S^W en posant $f_{\bar{I}} = f_I$ et

$$(\alpha_i)_{\bar{I}}(d_1, \dots, d_m, d) = \begin{cases} d & \text{si } \forall j \leq m_i, \quad d_j \neq \perp, \\ \perp & \text{sinon.} \end{cases}$$

La validité de notre approche se fonde sur le lemme :

LEMME 3 : *Pour tout $i \leq k$, $(\alpha_i)_{\bar{I}}$ est une application continue de D^{m_i} dans D .*

Démonstration : Pour simplifier l'écriture posons $m_i = m$ et $(\alpha_i)_{\bar{I}} = \bar{\alpha}$.

Soit E une partie dirigée de D^{m+1} et $\vec{d} = \langle d_1, \dots, d_m, d \rangle = \sqcup E$.

S'il existe $j \leq m$ tel que $d_j = \perp$ alors d'une part $\bar{\alpha}(\vec{d}) = \perp$ et d'autre part $\forall \vec{d}' = \langle d'_1, \dots, d'_m, d' \rangle$ appartenant à E , $d'_j = \perp$ d'où $\forall \vec{d}' \in E$, $\bar{\alpha}(\vec{d}') = \perp$.

Si, pour tout $j \leq m$, $d_j \neq \perp$, on a $\bar{\alpha}(\vec{d}) = d$ et il existe $\vec{d}' = \langle d'_1, \dots, d'_m, d' \rangle$ dans E tel que $\forall j \leq m$, $d'_j \neq \perp$. Soit $\vec{d}'' = \langle d''_1, \dots, d''_m, d'' \rangle$ dans E ; on a toujours $\bar{\alpha}(\vec{d}'') = \perp$ ou $\bar{\alpha}(\vec{d}'') = d''$, d'où $\bar{\alpha}(\vec{d}'') \leq d$.

Réciproquement soit d_0 tel que $\forall \vec{d}'' \in E$, $d_0 \geq \bar{\alpha}(\vec{d}'')$ et montrons que $d_0 \geq d$. Soit $E' = \{ \vec{d}'' \in E / \bar{\alpha}(\vec{d}'') \geq \vec{d}' \}$. Il est clair que E' est une partie dirigée et que $\sqcup E = \sqcup E'$. Mais si $\vec{d}'' \geq \vec{d}'$ on a $d''_i \neq \perp$ pour tout $i \leq m$ et donc $\bar{\alpha}(\vec{d}'') = d'' \leq d_0$ d'où

$$\vec{d}'' \leq \langle d''_1, \dots, d''_m, d_0 \rangle$$

et donc $\sqcup E = \sqcup E' \leq \langle d_1, \dots, d_m, d_0 \rangle$ ce qui entraîne bien $d \leq d_0$.

C. Q. F. D.

Nous sommes donc dans les conditions d'application du théorème 2, c'est-à-dire que le plus petit point fixe de la fonctionnelle \hat{S}_I^W est $\langle g_1, \dots, g_k \rangle$ où g_i est définie par

$$g_i(\vec{d}) = \sqcup \{ (\bar{I}, \gamma_{\vec{d}}) t / t \in L_i(S^W) \}.$$

Remarquons que d'après la définition de \bar{I} et de \hat{S}_I^W nous avons

$$(\vec{I}, \gamma, \vec{g}) \alpha_i(x_{\sigma_i(1)}, \dots, x_{\sigma_i(m_i)}, u_i) = \begin{cases} (\vec{I}, \gamma, \vec{g}) u_i & \text{si } \forall l \leq m_i, \gamma(x_{\sigma_i(l)}) \neq \perp, \\ \perp & \text{sinon} \end{cases}$$

et comme pour $u_i \in T_{\Sigma \cup \Phi}(V_{n_i})$, $(\bar{I}, \gamma, \vec{g}) u_i = (I, \gamma, \vec{g}) u_i$ nous obtenons donc : si $\langle g'_1, \dots, g'_k \rangle = \hat{S}_I^W(\vec{g})$ alors :

$$g'_i(\vec{d}) = \begin{cases} g''_i(\vec{d}) & \text{si } \forall l \in W_i, d_l \neq \perp, \\ \perp & \text{sinon,} \end{cases}$$

où $\langle g''_1, \dots, g''_k \rangle = \hat{S}_I(\vec{g})$.

Il en résulte que les deux k -uples de fonctions $\hat{S}_I(\vec{g})$ et $\hat{S}_I^W(\vec{g})$ coïncident sur $D - \{ \perp \}$.

3. SCHÉMAS GÉNÉRIQUES

DÉFINITIONS : Un schéma générique (cf. [9]) est un ensemble :

$$S = \left\{ \begin{array}{l} \varphi_i(x_1, \dots, x_{n_i}) = \begin{cases} q_{i,1} \rightarrow u_{i,1}, \\ \vdots \\ q_{i,m_i} \rightarrow u_{i,m_i}, \end{cases} \\ i = 1, \dots, k \end{array} \right.$$

tel que $u_{i,j} \in T_{\Sigma \cup \Phi}(V_{n_i})$ et $q_{i,j} \in T_{\Gamma}$ où Γ est un alphabet gradué disjoint de Σ

Une interprétation de S est un couple (I, J) où I est défini comme précédemment et J associée à $q_{i,j}$ un ensemble de points isolés de D^{n_i} tel que

$$\left. \begin{array}{l} \text{si } j \neq j', \quad \forall \vec{d} \in J(q_{i,j}), \quad \forall \vec{d}' \in J(q_{i,j'}), \\ \vec{d}' \text{ et } \vec{d} \text{ n'ont aucun majorant commun dans } D^{n_i}. \end{array} \right\} \quad (\star)$$

(Rappelons qu'un point d d'un ensemble ordonné inductif D est *isolé* si pour toute partie dirigée E de D telle que $d \leq \bigcup E$, il existe un élément d' de E tel que $d \leq d'$.)

La propriété (★) permet de définir l'application partielle $\text{seuil}_i : D^n \rightarrow \{1, \dots, m_i\}$ par : $\text{seuil}_i(\vec{d})$ est l'unique élément j , s'il existe, tel qu'il existe \vec{d}' appartenant à $J(q_{i,j})$ tel que $\vec{d}' \leq \vec{d}$. Considérée comme une application de D^n dans $\{1, \dots, m_i\} \cup \{\perp\}$, seuil_i est continue (cf. [9]).

Les $q_{i,j}$ sont en quelques sorte des « conditions d'applicabilité » en ce sens que $\varphi_i(\vec{d})$ va se dériver en $u_{i,j}(\vec{d})$ à condition que $\text{seuil}_i(\vec{d})$ soit défini et égal à j . Il est clair que ces schémas généralisent bien la notion d'appel par valeur : il suffit de remplacer

$$\varphi_i(x_1, \dots, x_{n_i}) = u_i; W_i$$

par

$$\varphi_i(x_1, \dots, x_{n_i}) = q_i \rightarrow u_i$$

et de poser $J(q_i)$ égal à l'ensemble des points isolés $\langle d_1, \dots, d_{n_i} \rangle$ de D^n tels que si $j \in W_i$ alors $d_j \neq \perp$. On aura donc $\text{seuil}_i(\vec{d}) = 1$ ssi $\forall j \in W_i, d_j \neq \perp$.

Soit donc un schéma générique S . Pour chaque i , on considère un nouveau symbole fonctionnel d'arité $n_i + m_i$ et on considère le schéma

$$S^W = \begin{cases} \varphi_i(x_1, \dots, x_{n_i}) = \alpha_i(x_1, \dots, x_{n_i}, u_{i,1}, \dots, u_{i,m_i}), \\ i = 1, \dots, k. \end{cases}$$

L'interprétation (I, J) de S est étendue en une interprétation \bar{I} de S^W en posant $f_{\bar{I}} = f_I$ pour tout symbole fonctionnel f figurant dans S et

$$(\alpha_i)_{\bar{I}}(d_1, \dots, d_{n_i}, d'_1, \dots, d'_{m_i}) = \begin{cases} d'_j & \text{si } \text{seuil}_i(d_1, \dots, d_{n_i}) = j, \\ \perp & \text{sinon.} \end{cases}$$

Comme seuil_i est continue à valeur dans un domaine discret, il est facile de montrer que :

LEMME 4 : $(\alpha_i)_{\bar{I}}$ est une application continue de $D^{n_i+m_i}$ dans D .

On peut alors appliquer le théorème 2 : l'application fonctionnelle \hat{S}_I^W admet un plus petit point fixe qui peut être aussi défini au moyen de l'interprétation \bar{I} à partir du langage d'arbres engendré par la grammaire associée à S^W .

Or d'après la définition de \bar{I} et de \hat{S}_I^W , on a : si $\langle g'_1, \dots, g'_k \rangle = \hat{S}_I^W(g)$ alors $g'_i(d) = \text{si } \text{seuil}_i(\vec{d}) = \perp \text{ alors } \perp \text{ sinon } (I, \gamma_d, g) u_{i, \text{seuil}_i}(\vec{d})$.

Cette fonctionnelle \hat{S}_I^W est donc bien identique à celle que Lang associe au schéma générique S . Il en résulte que pour les schémas génériques *il est encore possible de définir une sémantique algébrique* qui coïncide avec la sémantique dénotationnelle.

4. TESTS GÉNÉRALISÉS

On a pu définir une sémantique algébrique de l'appel par valeur, et plus généralement une sémantique algébrique des schémas génériques de Lang en introduisant dans la syntaxe des nouveaux symboles fonctionnels auxquels on donne une interprétation canonique. Étant donné la nature de cette interprétation, on peut considérer ces nouveaux symboles comme des tests généralisés.

De façon plus précise une application continue g de D^{n+m} dans D est un *test généralisé de type (n, m)* s'il existe une partition E_0, E_1, \dots, E_n de D^n telle que

$$g(\vec{d}, d'_1, \dots, d'_m) = \begin{cases} \perp & \text{si } \vec{d} \in E_0, \\ d'_i & \text{si } \vec{d} \in E_i. \end{cases}$$

Il est clair que le test habituel (*cf.* [7]) est un test généralisé de type $(1, 2)$ et que tous les symboles α que nous avons introduits s'interprètent comme des tests généralisés. Nous pensons que ces tests généralisés sont susceptibles d'une caractérisation syntaxique analogue à celle que I. Guessarian a étudié pour les tests habituels [7].

De même qu'à tout test habituel on peut associer un prédicat, à tout test généralisé g de type (n, m) on peut associer le « prédicat généralisé » Pg défini par

$$\text{Pg}(d) = \begin{cases} \perp & \text{si } \vec{d} \in E_0, \\ i & \text{si } \vec{d} \in E_i, \quad i \neq 0, \end{cases}$$

on retrouve la fonction seuil utilisée précédemment qui est donc une application continue de D^n dans $\{\perp, 1, \dots, m\}$. Comme Pg est continue à valeurs dans un ordonné discret, pour toute suite croissante $\vec{d}_1, \dots, \vec{d}_i, \dots$ d'éléments de D^n , il existe un entier i_0 tel que $\forall i \geq i_0, \text{Pg}(\vec{d}_i) = \text{Pg}(\bigsqcup \{\vec{d}_i / i \geq 1\})$.

Comme l'a fait remarquer B. Lang cette propriété est nécessaire pour montrer l'équivalence entre la sémantique opérationnelle et la sémantique dénotationnelle. Elle joue même un rôle intuitivement fondamental car elle assure que si un prédicat généralisé prend une valeur définie i pour une valeur \vec{d} de D^n qui est obtenue par un calcul infini (i. e. \vec{d} est la limite de la suite des approximants finis obtenus par des calculs finis) alors le prédicat prend la valeur i pour un approximant fini de \vec{d} . Cette valeur i peut donc être obtenue par un calcul fini.

Ainsi que nous l'avons déjà fait remarquer, la transformation syntaxique que nous utilisons revient à faire calculer deux fois les arguments d'une procédure. Le calcul « à côté » a pour seul objet d'évaluer ces arguments de façon à déterminer la valeur du prédicat Pg. Si Pg prend une valeur définie, le calcul « à côté » sera donc fini et dans ce cas on n'a rien changé aux propriétés de terminaison du programme.

Je remercie Jean-François Perrot de l'aide qu'il m'a apportée dans la mise en forme de ce texte.

BIBLIOGRAPHIE

1. E. ASTESIANO et G. COSTA, *On Algebraic Semantics of Polyadic Recursive Schemes in Les Arbres en Algèbre et en Programmation*, 2nd Colloque, Lille, 1977.
2. J. W. DE BAKKER, *Least Fixed Points Revisited*, Theor. Computer Science, vol. 2, 1976, p. 155-181.
3. G. BOUDOL, *Langages polyadiques algébriques. Théorie des schémas de programme : sémantique de l'appel par valeur*, Thèse 3^e cycle, Paris, 1975.
4. J. M. CADIOU, *Recursive Definitions of Partial Functions and their Computation*, Ph. D. Thesis Stanford University, 1972.
5. G. COSTA, Communication personnelle.
6. J. ENGELFRIET et E. M. SCHMIDT, *IO and OI*, Daimi Report PB 47, University of Aarhus, 1975.
7. I. GUESSARIAN, *Les tests et leur caractérisation syntaxique*, R.A.I.R.O. série Informatique Théorique, vol. 11, 1977, p. 133-156.
8. L. KOTT, *Sémantique algébrique d'un langage de programmation type Algol*, R.A.I.R.O. série Informatique Théorique, vol. 11, 1977, p. 237-263.
9. B. LANG, *Threshold Evaluation and the Semantics of Call by Value Assignment and Generic Procedures*, 4th Ann. A.C.M. Symp. on Principles of Programming Languages, 1977.
10. J. MCCARTHY, *A Basis for a Mathematical Theory of Computation in Computer Programming and Formal Systems*, BRAFFORT et HIRSCHBERG, éd., North Holland, Amsterdam, 1963.

11. T. S. E. MAIBAUM, *A Generalized Approach to Formal Languages*, J. Comput. System Sc., vol. 8, 1974, p. 402-432.
12. Z. MANNA et J. VUILLEMIN, *Fixpoint Approach to the Theory of Computation*, Comm of the Assoc. Comput. Mach., vol. 15, 1972, p. 528-536.
13. M. NIVAT, *On the Interpretation of Recursive Program Schemes*, Symposia Mathematica, vol. 15, 1975, p. 255-281.
14. G. D. PLOTKIN, *Call-by-Name, Call-by-Value and the ω -Calculus*, Theor. Computer Science, vol. 1, 1975, p. 125-159.
15. D. SCOTT, *Outline of a Mathematical Theory of Computation*, Oxford University, 1970.
16. J. VUILLEMIN, *Syntaxe, sémantique et axiomatique d'un langage de programmation simple*, Thèse d'État, Paris, 1974.