

PATRICE NAUDIN

Comparaison et équivalence de sémantiques pour les schémas de programmes non déterministes

Informatique théorique et applications, tome 21, n° 1 (1987), p. 59-91.

http://www.numdam.org/item?id=ITA_1987__21_1_59_0

© AFCET, 1987, tous droits réservés.

L'accès aux archives de la revue « Informatique théorique et applications » implique l'accord avec les conditions générales d'utilisation (<http://www.numdam.org/legal.php>). 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/>

COMPARAISON ET ÉQUIVALENCE DE SÉMANTIQUES POUR LES SCHÉMAS DE PROGRAMMES NON DÉTERMINISTES (*)

par Patrice NAUDIN (¹)

Communiqué par A. ARNOLD

Résumé. – Nous proposons trois définitions différentes de la sémantique d'un schéma de programme récursif non déterministe dans l'interprétation de Herbrand :

- la **sémantique opérationnelle**, basée sur la notion de calcul terminé introduite par Boudol [7, 9];
- la **sémantique dénotationnelle**, construite classiquement par une méthode de plus grand point fixe (Arnold, Nivat [3, 4]);
- la **sémantique « algébrique »** obtenue en ignorant le non déterminisme du schéma et en faisant une semi-interprétation du résultat ainsi obtenu.

Nous étudions les trois ensembles d'arbres ainsi obtenus, montrons qu'ils sont naturellement ordonnés par inclusion, dans le cas général, et égaux, dans le cas des schémas de Greibach.

En remarquant qu'un ensemble de fonctions univoques peut s'interpréter canoniquement comme une fonction multivoque, nous définissons une équivalence sur les ensembles d'arbres et montrons que, modulo cette équivalence, nos trois sémantiques sont égales.

Abstract. – We discuss three different definitions of the semantics of a non-deterministic recursive program scheme in Herbrand's universe:

- the **operational semantics** based on the terminating computations (first introduced by Boudol [7, 9]);
- the **denotational semantics** defined by a greatest fixed point method (Arnold, Nivat [3, 4]);
- the so-called **algebraic semantics** which is obtained in two steps:
 - first we consider the scheme as being a deterministic one, by hiding all non-deterministic choice symbols (**or**) in a tree, and computing all rewriting sequences of that tree,
 - second, we interpret in the resulting finite or infinite trees the **or** symbols as union of sets of trees.

We study those three sets of trees; we show that in the general case they are naturally embedded, and equal in the case of Greibach schemes.

Making notice of the fact that a set of mappings can be canonically interpreted as a multivalued-mapping, we define an equivalence relation on sets of trees, and we show that modulo this relation, the three semantics are indistinguishable.

(*) Reçu décembre 1985, révisé en avril 1986.

(¹) Laboratoire d'informatique, Poitiers, Unité n° 226, associée au C.N.R.S., Bordeaux.

INTRODUCTION

Depuis plusieurs années, l'étude de la sémantique des programmes récur­sifs est l'objet de nombreux travaux en informatique théorique.

Si la sémantique des programmes déterministes peut être considérée comme bien connue, en revanche le cas des programmes non déterministes pose encore des problèmes non résolus. De plus, beaucoup de travaux sur la programmation parallèle font référence, plus ou moins explicitement au non déterminisme [15, 1].

Dans cet article, nous nous proposons d'étudier les liens entre deux théories de la sémantique des programmes non déterministes :

- **les calculs réussis**, introduits par Arnold et Nivat [6, 8];
- **les calculs terminés** de Boudol [10, 12].

Notons que ces deux théories, bien que différentes, constituent deux extensions, la première d'un point de vue sémantique, la seconde d'un point de vue syntaxique, de la théorie des calculs finis.

Pour cette étude, nous nous plaçons dans un contexte bien particulier; nous ne nous intéressons pas à des programmes opérant sur un domaine quelconque, mais à des schémas de programmes (que l'on peut considérer comme des programmes opérant sur le domaine universel de Herbrand) qui calculent des arbres formels.

Dans leurs travaux, Arnold et Nivat [6, 8] appliquent la notion de calcul réussi (très informellement, c'est un calcul qui définit un arbre unique) à des domaines métriques et définissent de cette façon une sémantique opérationnelle pour un programme non déterministe.

Considérant la structure des calculs, Boudol [10, 12] définit un préordre dans l'ensemble des calculs et définit la sémantique opérationnelle d'un programme à l'aide des calculs maximaux pour ce préordre (qu'il nomme calculs terminés); il applique cette notion à des programmes opérant sur des « cpo » (ensemble ordonné complet) et non pas sur des domaines métriques.

On retrouve cette idée de calcul terminé dans Poigné [22], mais avec une connotation différente de celle de Boudol. Alors que la notion de terminaison d'un calcul est opérationnelle chez Boudol (un calcul étant, naturellement, la suite d'opérations effectuées sur un terme), elle est plutôt observationnelle chez Poigné pour qui le résultat obtenu prime sur la façon dont est mené le calcul. Encore une fois, ces deux approches sont issues d'une même théorie : celle des schémas de programmes déterministes. Dans ce cas, en effet, la propriété de Church Rosser permet de passer de l'une à l'autre notion.

Dans notre travail, nous avons choisi de définir la sémantique opérationnelle d'un schéma de programme par les calculs terminés à la Boudol; la notion de calcul réussi, utilisée par Arnold et Nivat, est beaucoup trop forte et restrictive. Quoi qu'il en soit, nous montrons que, dans le cas des schémas de Greibach, les deux définitions de la sémantique opérationnelle coïncident.

Ceci justifie pleinement notre choix; en effet si la sémantique opérationnelle est définie avec les calculs réussis, on n'obtient des propriétés intéressantes de cette sémantique que dans le cas des schémas de Greibach (voir les travaux d'Arnold et Nivat [6 à 9]).

Le non déterminisme des programmes, que nous étudions ici, est borné, et la stratégie de passage de paramètres utilisée pour les procédures, est l'appel par nom, sans restriction aucune. Dans Hennessy [14], et dans Arnold [2, 3], on trouve d'autres stratégies qui amènent à des résultats différents (appel par valeur [2, 14], appel synchrone [3] ou « Call-Time choice » [14]); les auteurs appliquent alors un formalisme différent selon la stratégie utilisée (magmoïde additif [3, 4] pour l'appel synchrone, contraintes [14] pour le « Call-Time choice »).

Il nous reste enfin à choisir la structure du domaine sur lequel opéreront nos schémas de programmes; devons-nous considérer le domaine de Herbrand comme un espace métrique, ou devons-nous lui adjoindre un élément Ω pour en faire un espace ordonné?

Nous avons choisi de définir, sur le domaine de Herbrand, une distance qui en fait un espace ultramétrique; cette façon de procéder nous permet de définir des sémantiques qui sont additives relativement au choix non déterministe, i. e. :

Si Sem est une sémantique, et si $\underline{\text{or}}$ est le symbole de choix non déterministe, on a

$$\text{Sem}(\underline{\text{or}}(t_1, t_2)) = \text{Sem}(t_1) \cup \text{Sem}(t_2).$$

Dans le cas où le domaine d'interprétation D est muni d'un ordre, on est amené à définir sur l'ensemble des parties un préordre qui prolonge l'ordre sur les domaines (powerdomains de Plotkin [21] et Smyth [24], voir aussi Hennessy [14]); la propriété d'additivité est alors conservée si l'on considère l'ensemble quotient $P(D)/\leq$, ou ce qui revient souvent au même si l'on ne considère que les parties convexes du domaine (une partie A de D est dite convexe si

$$\forall a, b \in A \text{ tels que } a \leq b, \text{ alors } a \leq c \leq b \Rightarrow c \in A).$$

Dans notre étude, nous n'introduisons pas d'ordre artificiel sur l'ensemble des parties du domaine, mais le fait d'utiliser une distance introduit malgré tout une confusion entre parties du domaine : en effet on ne peut pas distinguer, à l'aide de la distance, une partie du domaine de sa clôture topologique.

Ces choix faits, le cadre de notre étude est fixé, et nous pouvons alors définir, à partir des calculs terminés, une première notion de sémantique : la **sémantique opérationnelle**.

La seconde approche que nous utilisons, pour obtenir une sémantique d'un schéma de programme récursif non déterministe, est basée sur une méthode de point fixe (ces méthodes aux points fixes sont utilisées assez largement dans d'autres domaines; voir [20] par exemple). La **sémantique dénotationnelle** est définie ici comme plus grand point fixe d'une fonctionnelle [6, 8]. Remarquons que le fait de travailler sur un domaine métrique facilite la définition de cette sémantique.

Nous présentons enfin une troisième définition de la sémantique. Considérons tout d'abord, que le schéma de programme est déterministe, le symbole du choix non déterministe jouant le rôle d'un symbole terminal; nous pouvons alors déterminer la sémantique de ce schéma, puis interpréter les symboles de choix, dans cette sémantique, comme des unions d'ensembles. Nous obtenons la **sémantique dite algébrique** (du fait de la factorisation de son calcul). Cette méthode a été utilisée par Arnold et Nivat [9] dans le cas des domaines métriques, et par Boudol [10, 11] dans le cas des domaines ordonnés. Dans Abramsky [1], nous retrouvons un principe tout à fait analogue, pour définir la sémantique dénotationnelle, dans le cadre de la théorie des catégories. Quant à nous, nous utiliserons ici le formalisme et les définitions d'Arnold et Nivat [9].

En résumé, nous avons trois sémantiques d'un schéma de programme, opérationnelle (Val_{op}), dénotationnelle (Val_{den}) et algébrique (Val_{alg}) qui sont définies par des ensembles d'arbres.

L'étude détaillée de ces sémantiques et de leurs liens figure dans un travail antérieur [16].

Nous montrons ici essentiellement trois résultats :

1. Nos trois sémantiques sont totalement ordonnées par l'inclusion (si on les considère comme des ensembles d'arbres) :

$$\text{Val}_{\text{op}} \subseteq \text{Val}_{\text{den}} \subseteq \text{Val}_{\text{alg}}$$

et les inclusions peuvent être strictes dans le cas général.

2. Tout arbre peut être considéré comme une fonction (ou un opérateur) sur le domaine. Les sémantiques étant des ensembles d'arbres, peuvent être considérées comme des ensembles de fonctions. Or, de façon naturelle, tout ensemble de fonctions définit une fonction multivoque (nous retrouvons là une des notions utilisées par Arnold et Nivat [7, 8]). Nous montrons alors que nos trois sémantiques définissent des fonctions multivoques, identiques, et donc sont indistinguables d'un point de vue opératoire.

3. Dans le cas des schémas de Greibach, nos trois sémantiques sont égales. Ce résultat unifie les théories (calculs terminés ou réussis, domaines métriques ou ordonnés) qui sont à l'origine de ce travail.

Cet article est organisé de la façon suivante : après une partie préliminaire dans laquelle nous rappelons les définitions classiques utilisées, nous définissons dans la seconde partie les trois sémantiques et les comparons en tant qu'ensembles d'arbres. Dans la troisième partie, nous montrons qu'elles engendrent des fonctions multivoques identiques; puis en conclusion, nous étudions le cas des schémas de Greibach.

1. DÉFINITIONS PRÉALABLES

Nous rappelons ici les définitions, maintenant classiques, des arbres, et les différents outils s'y rapportant.

Soit G un **alphabet gradué** (c'est-à-dire un ensemble fini de symboles avec arité) et soit a sa **fonction arité**, $a : G \rightarrow \mathbb{N}$.

On note $G = G_0 \cup G_1 \cup \dots \cup G_n$ avec $G_i = a^{-1}(\{i\})$.

Soit $X = \{x_i / i \in \mathbb{N}^*\}$ un ensemble de **symboles de variables**; on note :

$$X_n = \{x_i / 1 \leq i \leq n\}.$$

DÉFINITION 1.1 : Arbres finis.

L'ensemble $M(G, X)$ des **arbres finis** construits sur G et X est défini inductivement par :

- $G_0 \cup X \subset M(G, X)$;
- Si $t_1, \dots, t_p \in M(G, X)$ et si $f \in G_p$ alors $f(t_1, \dots, t_p) \in M(G, X)$. \square

Nous allons maintenant munir $M(G, X)$ d'une distance, ce qui permettra ensuite de compléter l'espace topologique obtenu, et de définir l'ensemble des arbres finis et infinis.

Soit Ω un symbole d'arité 0 qui n'est ni dans G ni dans X ; on définit la **coupure à la profondeur n** de l'arbre t , $\alpha_n(t)$ par :

$$\alpha_0(t) = \Omega, \quad \forall t \in M(G, X)$$

$$\alpha_{n+1}(t) = \begin{cases} t & \text{si } t \in G_0 \cup X \\ f(\alpha_n(t_1), \dots, \alpha_n(t_p)) & \text{si } t = f(t_1, \dots, t_p) \text{ avec } f \in G_p. \quad \square \end{cases}$$

DÉFINITION 1.2 : Distance sur les arbres.

On définit sur $M(G, X)$ une **distance ultramétrique d** par :

$$\forall t, t' \in M(G, X) : d(t, t') = 2^{-\inf \{n/\alpha_n(t) \neq \alpha_n(t')\}}. \quad \square$$

On peut compléter l'espace métrique $(M(G, X), d)$ ce qui donne l'espace ultramétrique complet $(M^\infty(G, X), d)$ des arbres finis et infinis. (On trouvera tous les détails sur cet espace dans [3].)

REMARQUE : Si l'on ne prend qu'un nombre fini de variables, X_n , l'ensemble obtenu $(M^\infty(G, X_n))$ est compact (rappelons que G est fini). \square

NOTATION : Si A est une partie de $M^\infty(G, X)$, nous noterons \bar{A} son adhérence topologique.

Soit $P = \mathbb{N} - \{0\}$ l'alphabet infini dont les lettres sont les nombres entiers positifs. Nous allons définir les arbres comme des fonctions partielles du monoïde libre P^* dans $G \cup X$. (On considérera pour cela que chaque variable a pour arité 0.)

DÉFINITION 1.3 : Arbres comme fonctions partielles.

On appelle **arbre** toute fonction partielle t de P^* dans $G \cup X$ dont le **domaine** $\text{dom}(t)$ vérifie :

- (i) $\text{dom}(t)$ est clos par préfixe;
- (ii) $\forall u \in \text{dom}(t) : [u.i \in \text{dom}(t) \Leftrightarrow 1 \leq i \leq a(t(u))]$.

Un **arbre** sera dit **fini** si son domaine est fini et **infini** sinon. \square

REMARQUES : Les arbres définis comme fonctions partielles sont bien les mêmes que ceux définis au début de cette section. Par conséquent, dans la suite, nous utiliserons indifféremment l'une ou l'autre définition.

Le mot vide de P^* (qui représente la racine d'un arbre) est noté Λ .

Les éléments de $\text{dom}(t)$ sont appelés les nœuds de t .

Si $t(u) = g \in G \cup X$, on dit que u est une occurrence de g dans t .

Le sous-arbre t/u de t issu du nœud u est l'arbre t' tel que

$\forall v \in P^*, t'(v) = t(u.v)$ si $t(u.v)$ est défini. \square

Nous introduisons à présent la notion de substitution d'ensembles d'arbres (on trouvera une présentation rigoureuse de cette notion dans la théorie des magmoïdes d'Arnold et Dauchet [4]).

DÉFINITION 1.4 : Substitution dans les arbres.

- (i) Nous dirons que l'arbre t est d'arité k , s'il est élément de $M^\infty(F, X_k)$.
- (ii) Soient t un arbre fini d'arité k et t_1, \dots, t_k des arbres finis quelconques. On définit la composition de t et de $\langle t_1, \dots, t_k \rangle$ par :

$$t \bullet \langle t_1, \dots, t_k \rangle = \begin{cases} t & \text{si } t \in G_0 \\ t_i & \text{si } t = x_i \in X_k \\ f(u_1 \bullet \langle t_1, \dots, t_k \rangle, \dots, u_p \bullet \langle t_1, \dots, t_k \rangle), & \\ & \text{si } t = f(u_1, \dots, u_p) \text{ avec } f \in G_p \end{cases}$$

$t \bullet \langle t_1, \dots, t_k \rangle$ est l'arbre obtenu en substituant, dans t , chaque occurrence de x_i par t_i .

- (iii) On étend ce produit de composition aux ensembles d'arbres :

Soient $P \subseteq M(G, X_k)$ et $Q \subseteq M(G, X)^k$, on pose $P \bullet Q = \{t \bullet Q / t \in P\}$ avec

$$t \bullet Q = \begin{cases} \{t\} & \text{si } t \in G_0 \\ Q_i & \text{si } t = x_i \in X_k \\ \{f(t'_1, \dots, t'_p) / t'_i \in t_i \bullet Q\}, & \\ & \text{si } t = f(t_1, \dots, t_p) \text{ avec } f \in G_p. \end{cases}$$

(Dans l'écriture, on confondra souvent un singleton avec l'élément qu'il contient.)

- (iv) Finalement on peut étendre ce produit aux arbres infinis et ensembles d'arbres infinis.

Posons pour $P \subseteq M^\infty(G, X)$:

$$B_\varepsilon(P) = \{t \in M(G, X) / \exists T \in P : d(T, t) < \varepsilon\}.$$

Alors si $P \subseteq M^\infty(G, X_k)$ et si $Q \subseteq M^\infty(G, X)^k$, on définit :

$$P \bullet Q = \bigcap \{ \overline{B_\varepsilon(P) \bullet \langle B_\varepsilon(Q_1), \dots, B_\varepsilon(Q_k) \rangle} / \varepsilon > 0 \}.$$

Cette définition appliquée à des singletons nous donne la composition des arbres infinis. \square

PROPOSITION 1.1 (Arnold et Nivat [6]) : **Ce produit de composition est associatif et distributif à droite par rapport à la réunion (on dit aussi qu'il est additif à gauche).**

2. SÉMANTIQUES DES SCHÉMAS DE PROGRAMMES

Un **schéma de programme** Σ est un ensemble d'équations

$$\begin{cases} \varphi_i(x_1, \dots, x_{ni}) = \tau_i \\ \text{avec pour } 1 \leq i \leq k, \tau_i \in M(F_+ \cup \Phi, X_{ni}), \varphi_i \in \Phi_{ni} \end{cases}$$

où

- F est l'alphabet gradué des **symboles terminaux** (représentant les fonctions élémentaires de la machine);
- Φ est l'alphabet gradué des **symboles non terminaux** (représentant des symboles de procédures, l'arbre τ_i correspondant à un symbole φ_i est le corps de la procédure);
- **or** est un symbole d'arité 2 représentant le **choix non déterministe**.

Si $A \subseteq F \cup \Phi$, on note $A_+ = A \cup \{\text{or}\}$, $A_n = a^{-1}(\{n\}) \cap A$, où a est la fonction arité de $F_+ \cup \Phi \rightarrow \mathbb{N}$ (en d'autres termes, A_n est l'ensemble des éléments de A d'arité n).

- X est un ensemble de **variables** et on note $X_n = \{x_1, \dots, x_n\}$ et $X_0 = \emptyset$.

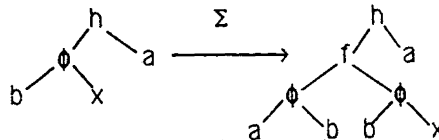
On peut, à partir d'un schéma de programme, définir deux **relations de dérivation** sur les arbres :

- La première est dite **déterministe**, et est la plus petite relation réflexive et transitive, compatible avec la structure de magma, engendrée par l'ensemble des couples (φ_i, τ_i) sur le $F_+ \cup \Phi$ -magma $M(F_+ \cup \Phi, X)$; on la note $\xrightarrow{\Sigma}$.

- La seconde relation de dérivation est obtenue en rajoutant les couples $(\text{or}(x_1, x_2), x_1)$ et $(\text{or}(x_1, x_2), x_2)$; elle est dite **non déterministe** et on la note $\xrightarrow{\Sigma^+}$.

Ces relations de dérivation formalisent la règle de recopie des programmes récurifs; par exemple :

Si $\Sigma : \varphi(x, y) \rightarrow f(\varphi(a, x), \varphi(b, y))$



DÉFINITION 2.1 : Dérivation des arbres.

On dit que t se *dérive* en t' dans Σ (resp. Σ^+), ce que l'on note $t \xrightarrow{\Sigma} t'$ (resp. $t \xrightarrow{\Sigma^+} t'$) si :

$$- t = t'$$

ou bien

$$- t = \varphi_i(t_1, \dots, t_{n_i}) \text{ et } t' = \tau_i \bullet \langle t'_1, \dots, t'_{n_i} \rangle, \text{ avec } \varphi_i \in \Phi_{n_i}, \text{ et pour tout } j :$$

$$t_j \xrightarrow{\Sigma} t'_j \quad (\text{resp } t_j \xrightarrow{\Sigma^+} t'_j)$$

ou bien

$$- t = G(t_1, \dots, t_n) \text{ et } t' = G(t'_1, \dots, t'_n), \text{ avec } G \in (F_+ \cup \Phi)_n, \text{ et pour tout } i :$$

$$t_i \xrightarrow{\Sigma} t'_i \quad (\text{resp } t_i \xrightarrow{\Sigma^+} t'_i)$$

ou bien (pour Σ^+ seulement)

$$- t = \underline{\text{or}}(t_1, t_2), \text{ avec } t' = t'_1 \text{ ou } t' = t'_2, \text{ et } t_i \xrightarrow{\Sigma^+} t'_i.$$

Une *dérivation* dans Σ^+ (resp. Σ) est définie comme une suite d'arbres $(t_n)_{n \in \mathbb{N}}$ tels que pour tout $n \in \mathbb{N}$:

$$t_n \xrightarrow{\Sigma^+} t_{n+1} \quad (\text{resp. } t_n \xrightarrow{\Sigma} t_{n+1}). \quad \square$$

Nous verrons plus loin une méthode plus fine pour représenter les dérivations.

Remarquons dès à présent que si t est d'arité n et si $t \rightarrow t'$, alors t' est d'arité n . \square

Notions de dérivations achevées

On dit, de façon très naturelle, qu'une *dérivation finie* $(t_n)_{n \leq N}$ est *achevée* si l'une des deux conditions suivantes (trivialement équivalentes) est vérifiée :

- L'arbre t_N est terminal i.e. ne contient plus aucune occurrence de non terminal (ni de $\underline{\text{or}}$ dans le cas de $\xrightarrow{\Sigma^+}$).

- On ne peut appliquer aucune règle de dérivation à l'arbre t_N (et donc, le calcul a effectué toutes les réécritures possibles en partant de t_0).

Ces deux définitions de l'achèvement d'une dérivation finie, vont, dans le cas des dérivations infinies, prendre des significations différentes.

Définissons tout d'abord notre notion de résultat d'une dérivation.

DÉFINITIONS 2. 2:

(i) Soit P une partie de $M^\infty(F, X_{n_1}) \times \dots \times M^\infty(F, X_{n_k})$; on définit l'application π_P de $M(F_+ \cup \Phi, X)$ dans $\mathcal{P}(M^\infty(F, X))$ par:

$$\pi_P(t) = \begin{cases} \{t\} & \text{si } t \in F_0 \cup X \\ f \bullet \langle \pi_P(t_1), \dots, \pi_P(t_n) \rangle, & \text{si } t = f(t_1, \dots, t_n) \text{ avec } f \in F_n \\ P_i \bullet \langle \pi_P(t_1), \dots, \pi_P(t_{n_i}) \rangle, & \text{si } t = \varphi_i(t_1, \dots, t_{n_i}) \text{ et } \varphi_i \in \Phi_{n_i} \\ \pi_P(t_1) \cup \pi_P(t_2), & \text{si } t = \underline{\text{or}}(t_1, t_2). \end{cases}$$

(ii) On note $D = \langle M^\infty(F, X_{n_1}), \dots, M^\infty(F, X_{n_k}) \rangle$. On peut montrer facilement (par induction sur t), que si $t \xrightarrow{\Sigma} t'$ ou bien si $t \xrightarrow{\Sigma^+} t'$, alors $\pi_D(t) \supseteq \pi_D(t')$ voir [7, 8].

(iii) Si $\delta = (t_n)_{n \leq N}$ est une dérivation, on appelle **résultat de δ** et on note $\text{res}(\delta)$ l'ensemble $\bigcap \{ \pi_D(t_n) / n \leq N \}$. Il est à noter que $\text{res}(\delta)$ est un ensemble compact non vide [7, 8, 16].

Si δ est une dérivation finie achevée, on remarque que $\text{res}(\delta)$ est réduit à un singleton; appliquons cela aux dérivations infinies et nous obtenons la notion de **dérivation réussie** (introduite par Arnold et Nivat [7, 8]).

(iv) La **dérivation δ** est dite **réussie** si son résultat est réduit à un singleton (cette notion est discutée dans [8]). \square

La seconde notion d'achèvement des dérivations est due à Boudol [10, 12], c'est la notion de **calcul terminé**. Pour la définir, nous avons besoin d'une structure plus fine sur les dérivations; nous devons savoir, en plus du fait que t se dérive en t' , quelles sont les règles qui ont été appliquées à t pour obtenir t' , et en quelles occurrences elles ont été appliquées. Un **calcul** (défini formellement au paragraphe 2. 1) nous apporte ces informations.

REMARQUE: La notion de calcul contient celle de dérivation et donc nous pouvons appliquer aux calculs les définitions et les résultats relatifs aux dérivations. \square

Nous dirons alors que deux **calculs finis** sont **équivalents par permutation**, s'ils effectuent les mêmes dérivations, peut-être à des instants différents. Un calcul fini δ sera dit \leq -**inférieur** à un calcul δ' si δ peut se prolonger en un calcul équivalent par permutation à δ' .

REMARQUE: un calcul fini achevé ne peut pas se prolonger et donc, il est maximal pour le préordre \leq . \square

Le **préordre et l'équivalence par permutations** s'étendent de façon naturelle aux calculs infinis, et permettent alors de définir les **calculs terminés** (calculs qui sont maximaux pour le préordre \leq).

Le lecteur se reportera à Boudol [10, 12] pour une présentation rigoureuse de la théorie des calculs, ou bien à Naudin [16] pour une version plus simple de cette théorie.

Dans cette partie, nous définissons successivement :

- la **sémantique opérationnelle** d'un schéma de programme, à l'aide de **calculs terminés**;
- la **sémantique dénotationnelle**, par une méthode de **point fixe**;
- la **sémantique algébrique**, obtenue en **interprétant partiellement** les arbres obtenus au cours d'une dérivation.

Nous donnons ensuite les relations d'inclusion existant entre ces différents ensembles d'arbres.

2.1. Sémantique opérationnelle

DÉFINITION 2.3: Une **donnée de réécriture** σ sur un arbre fini $t \in M(F_+ \cup \Phi, X)$ est une fonction partielle de $\text{dom}(t)$ dans l'ensemble des règles Σ^+ , telle que si σ est définie en w , alors w est une occurrence de règle de Σ^+ dans t [on représente souvent une donnée de réécriture comme un ensemble de couples $\langle w, r \rangle$ où w est une occurrence de règle et $r = \sigma(w)$]. \square

Si σ est une donnée de réécriture sur t , on note $\underline{\sigma}(t)$ le résultat de l'application de σ à t (qui consiste à réécrire simultanément toutes les occurrences de non terminaux et de **or** dans t selon la règle donnée par σ).

REMARQUE: si $\langle w, r \rangle \in \sigma$, et si le symbole figurant à l'occurrence w dans t est non terminal, alors la règle $r = \sigma(s)$ est déterminée de façon unique (par Σ); seule la réécriture d'un symbole **or** fait apparaître le non déterminisme du schéma. \square

Exemple :

$$\Sigma: \begin{cases} \varphi = f(a, b) & \text{règle } r_1 \\ \psi(x_1) = g(x_1, \varphi) & \text{règle } r_2. \end{cases}$$

Soient $t = \psi(\varphi)$ un arbre, et $\sigma = \{ \langle \Lambda, r_2 \rangle, \langle 1, r_1 \rangle \}$ une donnée de réécriture sur t , on a alors :

$$t \xrightarrow{\sigma} \underline{\sigma}(t) = g(f(a, b), \varphi).$$

DÉFINITION 2.4 : Un *calcul*, fini ou infini, sur t est une suite $\delta = (\sigma_n)_{n \in \mathbb{N}}$ telle que :

$$\delta_0 = t;$$

σ_i est une donnée de réécriture sur δ_i ;

$$\delta_{i+1} = \sigma_i(\delta_i). \quad \square$$

REMARQUE : Si $\delta = (\sigma_n)_{n \in \mathbb{N}}$ est un calcul, $(\delta_n)_{n \in \mathbb{N}}$ est une dérivation au sens défini plus haut; et réciproquement, si $(t_n)_{n \in \mathbb{N}}$ est une dérivation, alors il existe des données de réécritures σ_n telles que $\delta = (\sigma_n)_{n \in \mathbb{N}}$ soit un calcul vérifiant $\delta_n = t_n$. Nous passerons souvent de l'une à l'autre de ces notions, sans autre précision. \square

NOTATIONS : Soit $R = \Sigma$ ou Σ^+ un schéma de calculs :

- $\Delta_R^*(t)$ est l'ensemble des calculs finis à partir de t .
- $\Delta_R^\infty(t)$ est l'ensemble des calculs finis ou infinis à partir de t .

Si $\delta = (\sigma_n)_{n \in \mathbb{N}}$:

- $\delta \upharpoonright^j$ désigne le calcul $(\sigma_n)_{n < j}$.
- $\delta \upharpoonright_i$ désigne le calcul $(\sigma_n)_{n \geq i}$.
- $\delta \upharpoonright_i^j$ désigne le calcul $(\sigma_n)_{i \leq n < j}$.

Si $\delta = (\sigma_n)_{n < N}$ est un calcul fini, alors N est appelé longueur du calcul et δ_N est noté $\underline{\delta}$.

Si, de plus, $\delta' = (\sigma'_n)_{n \in \mathbb{N}} \in \Delta_R^\infty(\underline{\delta})$, on note $\delta; \delta'$ le calcul $(\sigma'_n)_{n \in \mathbb{N}}$ avec

$$\begin{cases} \sigma'_n = \sigma_n & \text{si } n < N \\ \sigma'_n = \sigma'_{n-N} & \text{si } n \geq N. \end{cases} \quad \square$$

REMARQUE : Dans la suite nous identifierons un calcul de longueur 1 avec la donnée de réécriture qu'il représente. \square

LEMME 2.1 : Si σ et σ' sont deux données de réécriture sur t telles que $\sigma \cup \sigma'$ est aussi une donnée de réécriture sur t , alors l'ensemble

$$\sigma[\sigma'] = \{ \langle w', r \rangle / w : \langle w, r \rangle \in \sigma, w \notin \text{dom}(\sigma'),$$

$$w' \text{ est un résidu de } w \text{ dans } \sigma'(t) \}$$

est une donnée de réécriture sur $\sigma'(t)$ que l'on appelle reste de σ par σ' . \square

REMARQUES : Nous ne définissons pas les résidus d'une occurrence par une donnée de réécriture. Très informellement, l'ensemble des résidus d'une occurrence par une réécriture représente ce qu'est devenue cette occurrence

après application de la donnée de réécriture. Nous allons montrer sur des exemples que cette occurrence a pu :

1. Être dérivée, auquel cas elle n'a pas de résidu. Par exemple, soit $\Sigma : \{ \varphi(x) = f(\varphi(x)) \text{ règle } r \}$.

Alors l'arbre $\varphi(x)$ se réécrit en $f(\varphi(x))$ par application de la donnée de réécriture $\sigma = \langle \Lambda, r \rangle$. L'ensemble des résidus de Λ par σ est \emptyset .

2. Disparaître, et dans ce cas elle n'a pas de résidu. Par exemple, considérons $\Sigma : \{ \varphi(x, y) = f(\varphi(x, a)) \text{ règle } r \}$.

L'arbre $\varphi(x, \varphi(x, y))$ se dérive en $f(\varphi(x, a))$ par application de la donnée $\sigma = \langle \Lambda, r \rangle$; l'occurrence 2 (i. e. le deuxième φ de l'arbre) est effacée par la réécriture, car la deuxième variable du non terminal φ n'apparaît pas dans le membre droit de la règle r . Dans ce cas encore, l'ensemble des résidus de 2 est réduit à \emptyset .

3. Se reproduire. Par exemple si

$$\Sigma : \{ \varphi(x_1) = f(\varphi(x_1), a) \text{ règle } r \},$$

alors

$$t = \varphi(a) \xrightarrow{\sigma = \langle \Lambda, r \rangle} f(\varphi(a), a),$$

et, dans ce cas, l'ensemble des résidus de l'occurrence 1 par σ est $\{1.1\}$ (c'est-à-dire le premier « a » apparaissant dans l'expression $f(\varphi(a), a)$).

4. Être dupliquée. Si

$$\Sigma : \{ \varphi(x_1) = f(\varphi(x_1), x_1) \text{ règle } r \},$$

alors

$$t = \varphi(a) \xrightarrow{\sigma = \langle \Lambda, r \rangle} f(\varphi(a), a),$$

et donc l'ensemble des résidus de l'occurrence 1 par σ est $\{1.1, 2\}$ (il suffit de marquer le symbole situé en 1 dans t pour le montrer). Cela provient de la duplication de la variable x_1 dans le second membre de la règle r . \square

DÉFINITION 2.5 : (i) *Nous dirons que le calcul $\delta = (\sigma_n)_{n < N}$ est un développement de la donnée de réécriture σ sur t si :*

$$\sigma_0 = \sigma \quad \text{et} \quad N = 1$$

ou bien

$\sigma \supseteq \sigma_0$, $\sigma \neq \sigma_0$ et $(\sigma_n)_{1 \leq n < N}$ est un développement de $\sigma[\sigma_0]$.

(ii) Nous dirons que deux calculs finis δ et δ' sont équivalents (par permutation), $\delta \equiv \delta'$ ssi il existe des calculs $\delta_0, \dots, \delta_n$ tels que

$$\delta_0 = \delta \quad \text{et} \quad \delta_n = \delta',$$

et pour tout p :

$$\delta_p = \gamma_1; \gamma; \gamma_2 \quad \text{et} \quad \delta_{p+1} = \gamma_1; \gamma'; \gamma_2$$

où γ et γ' sont deux développements d'une même donnée de réécriture.

(iii) On définit alors le **préordre** \leq sur les calculs finis.

Pour $\delta, \delta' \in \Delta_R^*(t)$: $\delta \leq \delta'$ ssi $\exists \delta''$ tel que $\delta; \delta'' \equiv \delta'$.

On a alors la propriété : $\delta \equiv \delta'$ ssi $\delta \leq \delta'$ et $\delta' \leq \delta$.

(iv) Ce préordre s'étend aux calculs infinis de la manière suivante :

Si δ et δ' appartiennent à $\Delta_R^\infty(t)$:

$$\delta \leq \delta' \text{ ssi } \forall i \in N, \exists j \in N : \delta|^{i-1} \leq \delta'|^j.$$

On pose alors : $\delta \equiv \delta'$ ssi $\delta \leq \delta'$ et $\delta' \leq \delta$. \square

THÉORÈME (Boudol [10, 12]) : L'ensemble $(\Delta_R^\infty(t)/\leq, \leq)$ est un ordre partiel complet. C'est le complété de $(\Delta_R^*(t)/\leq, \leq)$. Par conséquent (lemme de Zorn), tout calcul est \leq -inférieur à un calcul \leq -maximal. On dit alors qu'un calcul est terminé s'il est \leq -maximal; on note $\overline{\Delta_R^\infty(t)}$ l'ensemble des calculs terminés à partir de l'arbre t .

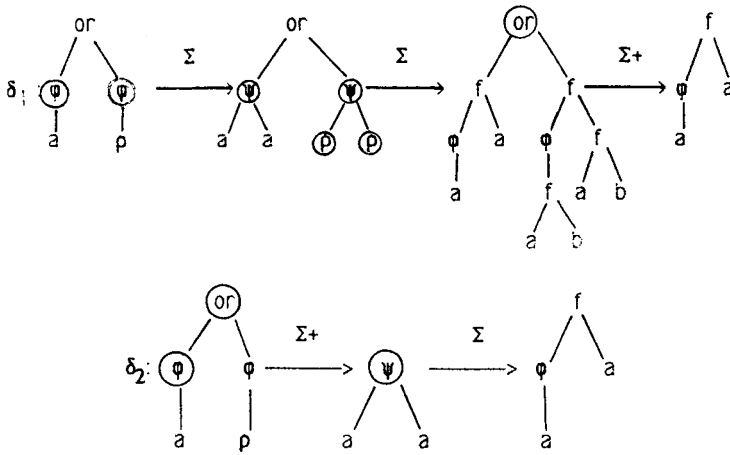
REMARQUE : Dans toute la suite de cet article, nous parlerons de calculs terminés sans préciser s'il s'agit de calculs terminés dans le système Σ^+ ou dans le système Σ : il s'agira toujours de calculs terminés dans Σ^+ (i.e. appartenant à $\overline{\Delta_\Sigma^\infty(\bullet)}$). En effet, le système Σ est déterministe et il est facile de voir que, dans ce cas, $(\Delta_\Sigma^\infty(t)/\leq, \leq)$ est un treillis complet et admet donc un élément maximal, dont un représentant est le calcul plein déterministe CPD(t) que nous introduisons plus loin.

Exemples :

1. Soit Σ le schéma

$$\left\{ \begin{array}{l} \varphi(x_1) = \psi(x_1, x_1) \\ \psi(x_1, x_2) = f(\varphi(x_1), x_2) \\ \rho = f(a, b) \end{array} \right.$$

Soient δ_1 et δ_2 les calculs suivants (les symboles dérivés sont entourés)

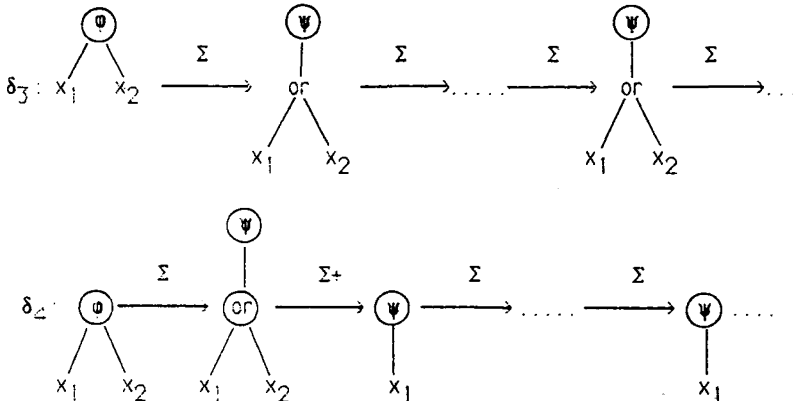


δ_1 et δ_2 sont des calculs équivalents (les dérivations de δ_1 que n'a pas faites δ_2 sont cachées par l'effacement de la branche gauche de l'arbre lors de la dérivation du or).

2. Soit Σ le schéma

$$\begin{cases} \varphi(x_1, x_2) = \psi(\underline{\text{or}}(x_1, x_2)) \\ \psi(x_1) = \psi(x_1). \end{cases}$$

Soient δ_3 et δ_4 les calculs suivants :



δ_3 et δ_4 ne sont pas équivalents; δ_4 est un calcul terminé de $\varphi(x_1, x_2)$ et δ_3 lui est strictement \leq -inférieur.

On trouvera une étude rigoureuse de ce dernier exemple dans [16]. \square

Ce préordre sur les calculs se comporte bien vis-à-vis du résultat, comme le montre le lemme suivant :

LEMME 2.2 : Soient δ et δ' deux calculs du même arbre tels que $\delta \leq \delta'$. Alors

$$\text{res}(\delta) \supseteq \text{res}(\delta').$$

Démonstration :

$$\delta \leq \delta' \Leftrightarrow \forall i, \exists j, \delta|^{i} \leq \delta'|^{j} \Rightarrow \forall i, \exists j, \underline{\delta}_i \rightarrow \underline{\delta}'_j$$

donc $\pi_D(\underline{\delta}_i) \supseteq \pi_D(\underline{\delta}'_j)$.

Par conséquent

$$\forall i, \pi_D(\underline{\delta}_i) \supseteq \text{res}(\delta')$$

et donc $\bigcap \{ \pi_D(\underline{\delta}_i) / i \in N \} = \text{res}(\delta) \supseteq \text{res}(\delta')$. \square

DÉFINITION 2.6 : On appelle *sémantique opérationnelle* d'un arbre t , et on note $\text{Val}_{\text{op}}(t)$, l'ensemble réunion des résultats de tous les calculs terminés de t

$$\text{Val}_{\text{op}}(t) = \bigcup \{ \text{res}(\delta) / \delta \in \overline{\Delta_{\Sigma^+}^{\infty}}(t) \}. \quad \square$$

Pour clore ce paragraphe, donnons quelques propriétés des calculs terminés (démontrées dans [16]).

LEMME 2.3 : Soit δ' un calcul terminé de t , soit δ un calcul déterministe de t [c'est-à-dire un calcul $\delta \in \Delta_{\Sigma^+}^{\infty}(t)$], alors :

- (i) $\delta \leq \delta'$;
- (ii) si de plus δ est fini, il existe un calcul terminé δ'' tel que

$$\delta; \delta'' = \delta'.$$

2.2. Sémantique dénotationnelle

Nous la définissons, classiquement, par une méthode de point fixe. Les résultats essentiels sur cette sémantique peuvent être trouvés dans Arnold et Nivat [6].

Nous allons uniquement rappeler ces résultats.

Soit $D_{\Sigma} = \mathcal{P}(M^{\infty}(F, X_{n_1})) \times \dots \times \mathcal{P}(M^{\infty}(F, X_{n_k}))$.

$D = M^{\infty}(F, X_{n_1}) \times \dots \times M^{\infty}(F, X_{n_k})$ est l'élément maximum de D_{Σ} pour l'inclusion.

On définit l'application Σ de D_{Σ} dans lui-même par :

$$\forall P \in D_{\Sigma}, \quad \Sigma(P) = \langle \pi_p(\tau_1), \dots, \pi_p(\tau_k) \rangle \quad (\text{cf. définition 2.2}).$$

PROPOSITION (Arnold et Nivat [6]) : $v(\Sigma) = \bigcap \{ \Sigma^n(D) / n \in \mathbb{N} \}$ est le plus grand point fixe de l'application Σ , et est appelé sémantique dénotationnelle du schéma de programme Σ . \square

Pour tout t dans $M(F_+ \cup \Phi, X)$ nous définissons la **sémantique dénotationnelle** de l'arbre t comme l'ensemble

$$\text{Val}_{\text{den}}(t) = \pi_{v(\Sigma)}(t)$$

2.2.1. Réalisation de la sémantique dénotationnelle

Nous allons voir maintenant que cette définition n'est pas vide, car il existe un calcul qui réalise la sémantique dénotationnelle i. e. :

Pour tout arbre t , il existe un calcul δ de t tel que : $\text{res}(\delta) = \text{Val}_{\text{den}}(t)$.

DÉFINITION 2.7 : L'application Σ de $M(F_+ \cup \Phi, X)$ dans lui-même est définie par :

$$\Sigma(t) = \begin{cases} t, & \text{si } t \in F_0 \cup X \\ f(\Sigma(t_1), \dots, \Sigma(t_n)), & \text{si } t = f(t_1, \dots, t_n) \text{ et } f \in (F_+)_n \\ \tau_i \langle \Sigma(t_1), \dots, \Sigma(t_{ni}) \rangle, & \text{si } t = \varphi_i(t_1, \dots, t_{ni}) \text{ et } \varphi_i \in \Phi_{ni}, \quad \square \end{cases}$$

La suite $(\Sigma^n(t))_{n \in \mathbb{N}}$ définit alors un calcul, le **calcul déterministe plein** à partir de t (qui consiste à réécrire à chaque étape tous les symboles non terminaux figurant dans l'arbre); on le note CPD(t).

LEMME 2.4 :

$$\forall t \in M(F_+ \cup \Phi, X), \quad \forall R \in D_{\Sigma} : \pi_R(\Sigma(t)) = \pi_{\Sigma(R)}(t). \quad \square$$

Démonstration : Elle se fait par induction sur t .

Nous n'examinons que le cas $t = \varphi_i(t_1, \dots, t_{ni})$ avec $\varphi_i \in \Phi_{ni}$

$$\pi_{\Sigma(R)}(t) = (\Sigma(R))_i \cdot \langle \pi_{\Sigma(R)}(t_1), \dots, \pi_{\Sigma(R)}(t_{ni}) \rangle ;$$

$(\Sigma(R))_i$ représente la i -ième composante du vecteur $\Sigma(R)$

$$= \pi_R(\tau_i) \cdot \langle \pi_{\Sigma(R)}(t_1), \dots, \pi_{\Sigma(R)}(t_{ni}) \rangle \quad \text{par définition de } \Sigma$$

$$= \pi_R(\tau_i) \cdot \langle \pi_R(\Sigma(t_1)), \dots, \pi_R(\Sigma(t_{ni})) \rangle \quad \text{par hypothèse d'induction}$$

$$= \pi_R(\tau_i) \cdot \langle \Sigma(t_1), \dots, \Sigma(t_{ni}) \rangle \quad \text{par définition de } \pi_R$$

$= \pi_R(\Sigma(t))$ par définition de Σ . \square

LEMME 2.5 :

$$\forall n \in \mathbb{N}, \quad \forall t \in M(F_+ \cup \Phi, X), \quad \forall P \in D_\Sigma : \pi_\Sigma n_{(P)}(t) = \pi_P(\Sigma^n(t)).$$

Démonstration : Par récurrence sur n . On utilise le lemme précédent en posant

$$R = \Sigma^{n-1}(P). \quad \square$$

LEMME 2.6 : Pour tout t dans $M(F_+ \cup \Phi, X)$, l'application de D_Σ dans $P(M^\infty(F, X_n))$ définie par $R \rightarrow \pi_R(t)$ est continue. (Les espaces de départ et d'arrivée étant munis de la distance de Hausdorff.)

Voir [16] pour la démonstration qui se fait par induction mais nécessite des définitions sans intérêt ici. \square

Nous sommes maintenant en mesure d'énoncer la :

PROPOSITION 2.1 : Le calcul CPD(t) réalise la sémantique dénotationnelle de t , i. e.

$$\text{res}(\text{CPD}(t)) = \text{Val}_{\text{den}}(t).$$

Démonstration : $(\Sigma^n(D))_{n \in \mathbb{N}}$ est une suite convergente dans D_Σ donc d'après le lemme précédent :

$$\begin{aligned} \lim_{n \rightarrow \infty} \pi_\Sigma n_{(D)}(t) &= \pi_{\lim_{n \rightarrow \infty} \Sigma^n(D)}(t) = \text{Val}_{\text{den}}(t) \\ \lim_{n \rightarrow \infty} \pi_\Sigma n_{(D)}(t) &= \lim_{n \rightarrow \infty} \pi_D(\Sigma^n(t)) = \text{res}(\text{CPD}(t)). \quad \square \end{aligned}$$

2.2.2. Comparaison entre sémantiques opérationnelle et dénotationnelle

Nous allons à présent montrer que Val_{op} et Val_{den} sont comparables en tant qu'ensembles d'arbres.

PROPOSITION 2.2 : Pour tout arbre t dans $M(F_+ \cup \Phi, X)$, on a : $\text{Val}_{\text{den}}(t) \supseteq \text{Val}_{\text{op}}(t)$.

Démonstration : D'après le lemme 2.3, pour tout calcul terminé δ , on a : $\delta \supseteq \text{CPD}(t)$, et donc, d'après le lemme 2.2 :

$$\text{res}(\text{CPD}(t)) \supseteq \text{res}(\delta).$$

De la définition de $\text{Val}_{\text{op}}(t)$, on peut alors déduire :

$$\text{Val}_{\text{den}}(t) = \text{res}(\text{CPD}(t)) \supseteq \text{Val}_{\text{op}}(t). \quad \square$$

2.3. Sémantique algébrique

Cette troisième sémantique a été utilisée par Boudol [10, 12] puis Arnold et Nivat [9]. Ici on sépare dans un schéma de programme non déterministe, la réécriture des symboles non terminaux (qui est une opération déterministe) et le choix non déterministe.

Nous allons donc faire des calculs déterministes et définir un résultat intermédiaire sans évaluer les symboles or. Ensuite, nous appliquerons à ces résultats intermédiaires une fonction dont le but est d'évaluer tous les symboles de choix non déterministe.

Nous effectuons donc une factorisation dans le calcul de la sémantique

$$t \xrightarrow{\text{calculs déterministes}} \bullet \xrightarrow{\text{évaluation de or}} \text{Val}_{\text{alg}}(t)$$

d'où le qualificatif d'algébrique que nous employons pour cette sémantique.

DÉFINITION 2.7 : La fonction γ , qui à un arbre non déterministe, associe l'ensemble d'arbres qu'il représente est définie de $M(F_+ \cup \Phi, X)$ dans $\mathcal{P}(M(F \cup \Phi, X))$ par :

$$\gamma(t) = \begin{cases} \{t\} & \text{si } t \in (F \cup \Phi)_0 \cup X \\ f \bullet \langle \gamma(t_1), \dots, \gamma(t_n) \rangle, & \text{si } t = f(t_1, \dots, t_n) \text{ et } f \in (F \cup \Phi)_n \\ \gamma(t_1) \cup \gamma(t_2) & \text{si } t = \underline{\text{or}}(t_1, t_2). \quad \square \end{cases}$$

Par exemple si $t = f(\underline{\text{or}}(a, b), \underline{\text{or}}(c, d))$, alors

$$\gamma(t) = \{f(a, c), f(a, d), f(b, c), f(b, d)\}.$$

On étend γ aux ensembles d'arbres en posant si $A \subset M(F_+ \cup \Phi, X)$ $\gamma(A) = \cup \{\gamma(t) / t \in A\}$, puis, aux arbres infinis et aux ensembles d'arbres infinis de la façon suivante :

si A est ensemble d'arbres infinis (éventuellement un singleton)

$$\gamma(A) = \cap \{ \overline{\cup \{\gamma(t) / d(t, T) < \varepsilon, t \text{ fini}, T \in A\}} / \varepsilon > 0 \}. \quad \square$$

Cette fonction γ possède quelques bonnes propriétés en particulier : elle est additive, i. e. si t_1 et t_2 sont deux arbres quelconques, alors

$$\gamma(\underline{\text{or}}(t_1, t_2)) = \gamma(t_1) \cup \gamma(t_2).$$

(On trouvera dans [16] d'autres propriétés que l'on exige habituellement d'une fonction de ce type.)

γ n'est malheureusement **pas continue** (au sens de la métrique de Hausdorff), mais elle se comporte bien vis-à-vis de l'intersection des ensembles d'arbres comme on le verra avec le lemme 2. 8.

Considérons à présent le calcul CPD(t); nous pouvons l'interpréter partiellement à l'aide de la fonction π définie de $M(F_+ \cup \Phi, X)$ dans $\mathcal{P}(M^\infty(F_+, X))$ comme le plus petit morphisme de F -magmas étendant la flèche

$$\varphi_i \rightarrow M^\infty(F_+, X_{ni}) \quad (\text{avec en particulier : } \pi(\underline{\text{or}}(t_1, t_2)) = \underline{\text{or}}(\pi(t_1), \pi(t_2))).$$

L'ensemble que nous obtenons de cette façon est un ensemble d'arbres, finis ou infinis, qui contiennent des occurrences de symboles $\underline{\text{or}}$. Nous pouvons alors appliquer à cet ensemble la fonction γ qui interprète le symbole $\underline{\text{or}}$, et nous obtenons la sémantique algébrique d'un terme t :

$$\text{Val}_{\text{alg}}(t) = \gamma(\bigcap \{ \pi(\Sigma^n(t)) / n \in N \})$$

REMARQUES : 1. Le terme de sémantique algébrique est habituellement réservé à un autre usage. En effet tout programme P est en réalité l'interprétation dans un domaine D d'un schéma de programme Σ . On peut calculer la sémantique du schéma de programme dans le domaine de Herbrand, puis interpréter cette sémantique dans le domaine D ; on obtient ce que l'on nomme la sémantique algébrique du programme P (voir Arnold et Nivat [8]).

2. La cohérence de la définition précédente de Val_{alg} n'est pas à démontrer; la fonction π a, en effet, des propriétés de croissance et de fermeture tout à fait analogues à celles de la fonction π_p (voir début de section 2).

LEMME 2. 7 : Pour tout $t \in M(F_+ \cup \Phi, X)$, $\gamma(\pi(t)) \cong \pi_D(t)$.

Démonstration : Par induction sur t .

Nous ne traitons que le cas $t = \varphi_i(t_1, \dots, t_{ni})$.

$$\begin{aligned} \gamma(\pi(t)) &= \gamma(M^\infty(F_+, X_{ni}). \langle \pi(t_1), \dots, \pi(t_{ni}) \rangle) \\ &\cong \gamma(M^\infty(F_+, X_{ni}). \langle \gamma(\pi(t_1)), \dots, \gamma(\pi(t_{ni})) \rangle) \\ &\cong M^\infty(F, X_{ni}). \langle \gamma(\pi(t_1)), \dots, \gamma(\pi(t_{ni})) \rangle \\ &\cong M^\infty(F, X_{ni}). \langle \pi_D(t_1), \dots, \pi_D(t_{ni}) \rangle, \quad \text{par hypothèse d'induction} \\ &= \pi_D(t). \quad \square \end{aligned}$$

REMARQUE : Dans la démonstration précédente, nous avons utilisé le fait que :

$$\gamma(A). \langle \gamma(B_1), \dots, \gamma(B_n) \rangle \subset \gamma(A. \langle B_1, \dots, B_n \rangle).$$

Nous démontrons pas ce résultat; on l'obtient par un raisonnement direct en revenant à la définition de γ pour les ensembles d'arbres finis ou infinis. \square

LEMME 2.8 : Soit $(A_i)_{i \in N}$, une suite décroissante de parties fermées de $M^\infty(F_+ \cup \Phi, X)$, alors on a la propriété suivante :

$$\bigcap \{ \gamma(A_i) / i \in N \} = \gamma(\bigcap \{ A_i / i \in N \}).$$

La démonstration utilise le fait que la fonction γ est continue au sens d'Eilenberg-Montgomery [5, 16]. \square

Nous obtenons alors comme corollaire immédiat des deux lemmes précédents, et des définitions de Val_{den} et Val_{alg} , la :

PROPOSITION 2.3 :

$$\text{Val}_{\text{alg}}(t) \cong \text{Val}_{\text{den}}(t). \quad \square$$

En nous résumant, nous obtenons le :

THÉORÈME 2.1 : Pour tout arbre $t \in M(F_+ \cup \Phi, X)$, on a $\text{Val}_{\text{alg}}(t) \cong \text{Val}_{\text{den}}(t) \cong \text{Val}_{\text{op}}(t)$.

De plus, comme le montrent les exemples ci-après, les inclusions inverses sont fausses en général.

Exemple 1 : $\text{Val}_{\text{op}}(t) \neq \text{Val}_{\text{den}}(t)$.

Soit Σ le schéma de programme

$$\begin{cases} \varphi(x_1, x_2) = \psi(\text{or}(x_1, x_2)) \\ \psi(x_1) = \psi(x_1). \end{cases}$$

Posons $t = \varphi(x_1, x_2)$, alors, il est aisé de constater (voir [16] pour une démonstration rigoureuse) que

$$\text{Val}_{\text{den}}(t) = M^\infty(F, X_2)$$

$$\text{Val}_{\text{op}}(t) = M^\infty(F, \{x_1\}) \cup M^\infty(F, \{x_2\}).$$

Dans ces conditions, s'il existe dans F un symbole d'arité supérieure à 1, par exemple f d'arité 2, alors

$$f(x_1, x_2) \in \text{Val}_{\text{den}}(t) - \text{Val}_{\text{op}}(t).$$

Exemple 2 : $\text{Val}_{\text{den}}(t) \neq \text{Val}_{\text{alg}}(t)$.

Soit Σ le schéma de programme :

$$\begin{cases} \psi(x_1) = f(\varphi(a), x_1) \\ \varphi(x_1) = \underline{\text{or}}(\varphi(a), x_1). \end{cases}$$

Posons $t = \psi(x_1)$. On a :

$$\text{Val}_{\text{den}}(t) = \{ f(T, x_1) / T \in M^\infty(F) \}$$

et

$$\text{Val}_{\text{alg}}(t) = \{ f(T, x_1) / T \in M^\infty(F, X) \}.$$

Par conséquent $f(x_1, x_1) \in \text{Val}_{\text{alg}}(t) - \text{Val}_{\text{den}}(t)$.

REMARQUES : 1. Ces deux exemples amènent à penser que si F ne contient que des symboles d'arité 1 ou 0 alors nos trois sémantiques sont égales.

2. **Additivité des sémantiques par rapport au symbole de choix non déterministe or.**

Pour $\text{Sem} = \text{Val}_{\text{op}}, \text{Val}_{\text{den}}, \text{Val}_{\text{alg}}$, on a :

$$\text{Sem}(\underline{\text{or}}(t_1, t_2)) = \text{Sem}(t_1) \cup \text{Sem}(t_2).$$

Ce résultat provient des points suivants :

- l'application π_p est additive par rapport au symbole or (ce qui démontre la propriété pour Val_{den} qui est égal à $\pi_{v(\Sigma)}$);
- tout calcul terminé pour t_1 ou pour t_2 définit un calcul terminé pour or(t_1, t_2), et réciproquement, tout calcul terminé de or(t_1, t_2) définit un calcul terminé de t_1 ou bien de t_2 (ce qui démontre la propriété pour Val_{op});
- la fonction γ est additive par rapport à or (ce qui démontre la propriété pour Val_{alg}).

3. ÉQUIVALENCE DES SÉMANTIQUES

Jusqu'à présent, nous avons donné les définitions de nos sémantiques, et quelques propriétés qui se déduisent immédiatement des définitions. Pour cela, nous les avons considérées comme des ensembles d'arbres finis ou infinis.

En nous plaçant maintenant dans un cadre moins grossier, nous allons montrer que les relations qui les lient sont beaucoup plus fortes.

La structure canonique de F -magma, qui existe sur $M^\infty(F)$ peut s'étendre, grâce au produit de composition, en une structure de $M^\infty(F, X)$ -magma; en effet, tout arbre fini ou infini, $T \in M^\infty(F, X_n)$ peut être considéré comme un

opérateur n -aire sur $M^\infty(F)$, ou de façon équivalente, comme une application de $M^\infty(F)^n$ dans $M^\infty(F)$.

Le produit de composition permet en réalité plus que cela : on peut composer des ensembles d'arbres (et pas seulement des arbres) et donc tout arbre est un opérateur sur $\mathcal{P}(M^\infty(F))$ qui est, de façon naturelle, le domaine d'interprétation universel d'un schéma de programme non déterministe.

Dans la section précédente, nous avons comparé ces ensembles d'opérateurs indépendamment de tout domaine d'application.

À présent, nous allons comparer ces ensembles en tenant compte du fait qu'ils opèrent sur un domaine bien défini, $\mathcal{P}(M^\infty(F))$.

Si A est un ensemble d'arbres de $M^\infty(F, X_n)$ nous définissons :

$$\mathcal{F}_A : \mathcal{P}(M^\infty(F))^n \rightarrow \mathcal{P}(M^\infty(F)) \\ \langle T_1, \dots, T_n \rangle \rightarrow \cup \{ T \bullet \langle T_1, \dots, T_n \rangle / T \in A \}.$$

À partir d'un ensemble de fonctions univoques A , nous obtenons de cette façon une fonction multivoque \mathcal{F}_A (Arnold et Nivat [7, 8] utilisent des fonctions multivoques, définies d'une façon un peu différente, pour exprimer la sémantique d'un schéma de programme récursif non déterministe); nous pouvons donc chercher des relations entre nos sémantiques, en comparant les fonctions multivoques qu'elles engendrent.

Dans la suite, nous noterons $\mathcal{P}_n = \mathcal{P}(M^\infty(F))^n$.

DÉFINITION 3.1 : Soient A et B deux parties de $M^\infty(F, X_n)$; nous dirons que A est équivalente à B ssi les fonctions multivoques engendrées par A et B coïncident sur le domaine $\mathcal{P}(M^\infty(F))$. Formellement :

$$A \approx B \Leftrightarrow \forall C \subset M^\infty(F)^n : A \bullet C = B \bullet C. \quad \square$$

Dans cette partie, nous allons comparer les fonctions multivoques engendrées par nos sémantiques et montrer qu'elles sont égales.

Nous montrerons ainsi que Val_{op} , Val_{den} et Val_{alg} , bien qu'intrinsèquement distincts sont indistinguables (\approx -équivalents) lorsqu'on les applique sur le domaine (ce qui correspond à la seule connaissance pratique que l'on puisse en avoir).

3.1. Comparaison entre $\text{Val}_{\text{alg}}(t)$ et $\text{Val}_{\text{den}}(t)$

LEMME 3.1 : Soit $(P_i)_{i \in N}$ une suite décroissante de parties fermées non vides de $M^\infty(F, X_n)$. Soit A un élément fermé de P_n , alors :

$$\cap \{ P_i \bullet A / i \in N \} = (\cap \{ P_i / i \in N \}) \bullet A.$$

Ce résultat est le corollaire d'un résultat plus général d'Arnold et Nivat [6], faisant intervenir les limites de suites d'ensembles au sens de Painlevé. \square

LEMME 3.2 : Pour tout $t \in M(F_+ \cup \Phi, X)$: $\gamma(\pi(t)) \approx \pi_D(t)$.

Démonstration par induction sur t : Nous n'étudierons que deux cas :

$$t = \underline{\text{or}}(t_1, t_2) \quad \text{et} \quad t = \varphi_i(t_1, \dots, t_{n_i}).$$

Nous supposons donc que l'arité de t est n ; soit $A \in \mathcal{P}_n$; nous devons montrer que :

$$\gamma(\pi(t)) \bullet A = \pi_D(t) \bullet A.$$

1. Si $t = \underline{\text{or}}(t_1, t_2)$, alors

$$\begin{aligned} \pi_D(t) \bullet A &= (\pi_D(t_1) \cup \pi_D(t_2)) \bullet A \\ &= (\pi_D(t_1) \bullet A) \cup (\pi_D(t_2) \bullet A). \\ &\quad \text{car le produit de composition est additif à gauche} \\ &= (\gamma(\pi(t_1)) \bullet A) \cup (\gamma(\pi(t_2)) \bullet A), \\ &\quad \text{par hypothèse d'induction} \\ &= \gamma(\pi(t)) \bullet A. \end{aligned}$$

2. Si $t = \varphi_i(t_1, \dots, t_{n_i})$, avec $\varphi_i \in \Phi_{n_i}$, alors :

$$\pi_D(t) \bullet A = M^\infty(F, X_{n_i}) \bullet \langle \pi_D(t_1), \dots, \pi_D(t_{n_i}) \rangle \bullet A.$$

Puisque, dans $\pi_D(t)$, on substitue toute occurrence de variable par un arbre sans variable, $\pi_D(t) \bullet A$ est inclus dans $M^\infty(F)$.

Or $M^\infty(F, X_{n_i}) \cong M^\infty(F)$ et $M^\infty(F) \bullet A = M^\infty(F)$, par conséquent :

$$\pi_D(t) \bullet A = M^\infty(F).$$

Un raisonnement tout à fait analogue permet de montrer que

$$\gamma(\pi(t)) \bullet A = M^\infty(F), \quad \text{d'où le résultat cherché.} \quad \square$$

Nous pouvons énoncer le premier résultat :

PROPOSITION 3.1 : Pour tout arbre t de $M(F_+ \cup \Phi, X)$, on a $\text{Val}_{\text{den}}(t) \approx \text{Val}_{\text{alg}}(t)$.

Démonstration : D'après le lemme 3.2 :

$$\pi_D(\Sigma^n(t)) \bullet A = \gamma(\pi(\Sigma^n(t))) \bullet A,$$

et donc, d'après le lemme 3.1 :

$$\begin{aligned} \bigcap \{ \pi_D(\Sigma^n(t)) \bullet A / n \in N \} &= \bigcap \{ \pi_D(\Sigma^n(t)) / n \in N \} \bullet A \\ &= \text{Val}_{\text{den}}(t) \bullet A \\ \bigcap \{ \pi_D(\Sigma^n(t)) \bullet A / n \in N \} &= \bigcap \{ \gamma(\pi(\Sigma^n(t))) \bullet A / n \in N \} \\ &= \bigcap \{ \gamma(\pi(\Sigma^n(t))) / n \in N \} \bullet A \\ &= \gamma(\bigcap \{ \pi(\Sigma^n(t)) / n \in N \}) \bullet A \\ &\text{(cela découle du lemme 2.8)} \\ &= \text{Val}_{\text{alg}}(t) \bullet A. \quad \square \end{aligned}$$

3.2. Équivalence de Val_{den} et de Val_{op}

Dans cette section, nous allons montrer l'équivalence, pour tout arbre t , de $\text{Val}_{\text{den}}(t)$ et de $\text{Val}_{\text{op}}(t)$.

La démonstration complète de ce résultat nécessite l'introduction de notions assez complexes telles que la compatibilité entre calculs, et des résultats très particuliers sur les calculs terminés.

Afin de ne pas surcharger cet article de démonstrations longues et fastidieuses, nous admettrons deux résultats (après avoir donné une idée de leur preuve) qui nous permettront de montrer le résultat cherché.

DÉFINITION 2.2 : 1. *On dira qu'une occurrence de $\underline{\text{or}}$ dans t est semi-externe si elle n'est située sous aucun symbole non terminal. Formellement : w est une occurrence semi-externe de $\underline{\text{or}}$ dans t si*

$$\forall v <_{\text{pref}} w, \quad t(v) \in F_+$$

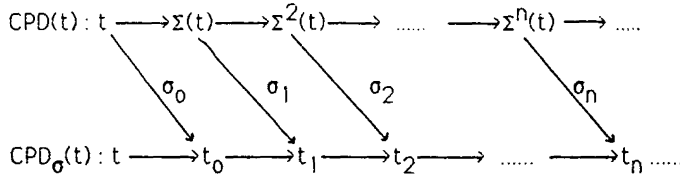
2. *Une donnée de choix σ sur un arbre t est une donnée de réécriture sur t dont le domaine est formé de toutes les occurrences semi-externes de $\underline{\text{or}}$ dans t .*

3. *Un système de choix $(\sigma_i)_{i \in N}$ [pour le calcul $\text{CPD}(t)$] est une suite de données de choix σ_i telles que :*

$$\left\{ \begin{array}{l} \sigma_i \text{ est une donnée de choix sur } \Sigma^i(t) \\ \sigma_{i+1} \supseteq \sigma_i. \quad \square \end{array} \right.$$

[Il faut remarquer, en effet, que si w est une occurrence semi-externe de $\underline{\text{or}}$ dans t , alors c'est aussi une occurrence semi-externe de $\underline{\text{or}}$ dans $\Sigma(t)$.]

NOTATION : Si $\sigma = (\sigma_i)_{i \in N}$ est un système de choix pour CPD (t) , alors nous désignerons par $\text{CPD}_\sigma(t)$ le calcul défini par le diagramme suivant :



3.2.1. Systèmes de choix associés à un calcul terminé

Après ces définitions préliminaires, nous allons voir qu'à tout calcul terminé δ , on peut associer de façon unique un système de choix σ^δ de sorte que :

PROPRIÉTÉ 3.1 : $\text{CPD}_{\sigma^\delta}(t) \leq \delta$.

PROPRIÉTÉ 3.2 : Pour tout i , il existe une donnée de réécriture σ' interne (i. e. qui n'opère que sur des occurrences situées en dessous de symboles non terminaux) telle que :

$$\delta^i \leq \text{CPD}_{\sigma^\delta}(t)^i; \sigma'.$$

Ces deux propriétés nous permettent alors de déduire que :

$$\cup \{ \text{res}(\text{CPD}_{\sigma^\delta}(t)) / \delta \text{ terminé} \} \subset \text{Val}_{\text{op}}(t),$$

où \subset est le préordre induit par l'équivalence \approx sur les ensembles d'arbres.

Nous montrons tout d'abord un lemme utile pour la suite.

LEMME 3.3 : Si t se réécrit en t' , par des règles appliquées en des occurrences situées sous des non terminaux alors :

$$\pi_D(t) \approx \pi_D(t').$$

Démonstration par induction sur t :

- Si $t \in F_0 \cup X$, alors $t' = t$.
- Si $t = f(t_1, \dots, t_n)$ avec $f \in (F_+)_n$, alors $t' = f(t'_1, \dots, t'_n)$, avec pour tout i de 1 à n , t_i se dérive en t'_i par des réécritures internes, et donc par hypothèse d'induction ($\pi_D(t_i) \approx \pi_D(t'_i)$) on obtient le résultat cherché.
- Si $t = \varphi_i(t_1, \dots, t_{ni})$ alors $t' = \varphi_i(t'_1, \dots, t'_{ni})$. On a alors :

$$\pi_D(t) = M^\infty(F, X_{ni}) \bullet \langle \pi_D(t_1), \dots, \pi_D(t_{ni}) \rangle.$$

Or $M^\infty(F) \subseteq M^\infty(F, X_{ni})$ et $M^\infty(F) \bullet A = M^\infty(F)$, par conséquent :

$$M^\infty(F) \subseteq \pi_D(t).$$

De même pour $t' : M^\infty(F) \subseteq \pi_D(t')$.

En conclusion, pour A dans \mathcal{P}_n , n étant l'arité de t :

$$M^\infty(F) \subseteq \pi_D(t) \bullet A \subseteq M^\infty(F),$$

et donc :

$$\pi_D(t) \bullet A = \pi_D(t') \bullet A = M^\infty(F). \quad \square$$

Nous avons vu précédemment (lemme 2.3) que tout calcul terminé est \leq -supérieur à tout calcul déterministe et donc, en particulier, à $\text{CPD}(t)$. Par conséquent, on peut prolonger en un calcul équivalent à δ tout préfixe du calcul $\text{CPD}(t)$:

$$\forall i \in \mathbb{N}, \exists \gamma_i \text{ calcul terminé tel que } \delta \equiv \text{CPD}(t)|^i; \gamma_i.$$

Or nous avons montré dans un travail précédent [5, 16] que tout calcul terminé de t dérive toute occurrence semi-externe de symbole or dans t . Par conséquent, chaque calcul γ_i nous permet de définir une donnée de choix sur $\Sigma^i(t)$ que nous notons σ_i^δ .

On montre facilement que la suite $\sigma^\delta = (\sigma_i^\delta)_{i \in \mathbb{N}}$ est un système de choix pour $\text{CPD}(t)$ (i. e. σ^δ est une suite croissante pour l'ordre par prolongement des fonctions partielles sur P^*), que nous nommerons système de choix associé à δ .

La démonstration de la propriété 3.1 s'achève en rapportant au début de γ_i l'action de σ_i^δ , de la façon suivante : $\gamma_i \equiv \sigma_i^\delta; \gamma'_i$. Par conséquent :

$$\text{CPD}(t)|^i; \sigma_i^\delta \equiv \text{CPD}_{\sigma^\delta}(t)|^i \leq \delta.$$

La démonstration de la propriété 3.2 est plus délicate. Intuitivement $\text{CPD}(t)$ dérive à chaque étape de calcul tous les symboles non terminaux présents dans l'arbre, et aucun calcul terminé ne peut faire plus (sur les non terminaux) en une étape; cela tient à la définition d'une donnée de réécriture. La seule particularité qui distingue un calcul terminé quelconque du calcul CPD tient dans la dérivation des symboles de choix or.

On peut donc étendre un préfixe du calcul CPD , par des réécritures de symbole or (semi-externes, ou internes) en un calcul fini \leq -supérieur au préfixe de même longueur du calcul terminé considéré.

Si l'on sépare ensuite les réécritures de or internes et semi-externes, on obtient exactement l'inégalité donnée dans la propriété 3.2.

Le lemme 2.3 et la définition du résultat d'un calcul et de Val_{op} permettent alors d'énoncer le :

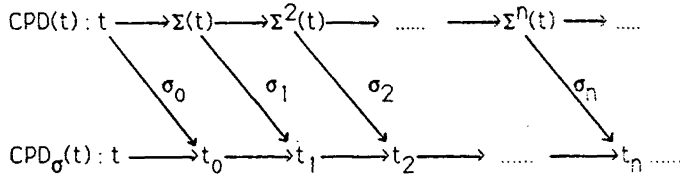
LEMME 3.4 :

$$\cup \{ \text{res}(\text{CPD}_{\sigma^{\delta}}(t)) / \delta \text{ terminé} \} \subseteq \text{Val}_{\text{op}}(t)$$

(i.e. $\cup \{ \dots \} \bullet A \subseteq \text{Val}_{\text{op}}(t) \bullet A$, pour tout A dans \mathcal{P}_n .)

3.2.2. Systèmes de choix pour CPD. Conclusion

Rappelons la construction de $\text{CPD}_{\sigma}(t)$ lorsque σ est un système de choix pour CPD (t) :



LEMME 3.5 : $\text{res}(\text{CPD}(t)) = \cup \{ \text{res}(\text{CPD}_{\sigma}(t)) / \sigma \text{ système de choix pour CPD}(t) \}$.

Démonstration : 1. Nous montrons tout d'abord que :

$$\pi_D(t) = \cup \{ \pi_D(\sigma(t)) / \sigma \text{ donnée de choix sur } t \}$$

- On a bien sûr $\pi_D(\sigma(t)) \subseteq \pi_D(t)$ pour toute donnée de choix σ sur t .
- Montrons par induction que

$$\pi_D(t) = \cup \{ \pi_D(\sigma(t)) / \sigma \text{ donnée de choix sur } t \}.$$

Le seul cas qui pose un problème se présente pour $t = \underline{\text{or}}(t_1, t_2)$:

$\pi_D(t) = \pi_D(t_1) \cup \pi_D(t_2)$, et par hypothèse d'induction

$$= (\cup \{ \pi_D(\sigma_1(t_1)) / \sigma_1 \text{ donnée de choix sur } t_1 \})$$

$$\cup (\cup \{ \pi_D(\sigma_2(t_2)) / \sigma_2 \text{ donnée de choix sur } t_2 \})$$

$= \cup \{ \pi_D(\sigma(t)) / \sigma \text{ donnée de choix sur } t \}$, par définition d'une donnée de choix.

2. D'après ce qui précède, on a :

$$\begin{aligned} \bigcap \{ \pi_D(\Sigma^n(t)) / n \in \mathbb{N} \} &= \text{res}(\text{CPD}(t)) \\ &= \bigcap \{ \bigcup \{ \pi_D(\sigma_n(\Sigma^n(t))) / \sigma = (\sigma_i) \text{ système de choix sur } \text{CPD}(t) \} / n \in \mathbb{N} \} \\ &\cong \bigcup \{ \bigcap \{ \pi_D(\sigma_n(\Sigma^n(t))) / n \in \mathbb{N} \} / \sigma = (\sigma_i) \text{ système de choix sur } \text{CPD}(t) \}. \end{aligned}$$

Pour montrer l'inclusion inverse, prenons t dans

$$\bigcap \{ \bigcup \{ \pi_D(\sigma_n(\Sigma^n(t))) / \sigma = (\sigma_i) \text{ système de choix sur } \text{CPD}(t) \} / n \in \mathbb{N} \}$$

alors, $\forall n \in \mathbb{N}, \exists \sigma = (\sigma_i)$, système de choix sur $\text{CPD}(t)$, tel que :

$$t \in \pi_D(\sigma_n(\Sigma^n(t))).$$

Les données de choix sur $\Sigma^n(t)$ étant en nombre fini, le lemme de Koenig permet de construire un système de choix $\sigma^0 = (\sigma_i^0)$ tel que

$$\forall n \in \mathbb{N}, t \in \pi_D(\sigma_n^0(\Sigma^n(t))),$$

ce qui démontre l'inclusion cherchée, et achève la preuve du lemme. \square

Au vu des lemmes 3.4 et 3.5 et sachant que $\text{Val}_{\text{op}}(t) \subseteq \text{Val}_{\text{den}}(t)$ il nous reste à montrer, pour terminer cette partie, que tout système de choix pour $\text{CPD}(t)$ est un système de choix associé à un calcul terminé, i. e. :

$$\forall \sigma \text{ système de choix pour } \text{CPD}(t), \exists \delta \text{ calcul terminé de } t \text{ tel que } \sigma = \sigma^\delta.$$

Ce résultat provient du fait qu'il existe, pour tout calcul (en particulier pour $\text{CPD}(t)$) un calcul maximal qui est au-dessus [lemme de Zorn appliqué à l'ensemble ordonné complet $\Delta_{\Sigma^+}^\omega(t)$].

Soit δ le calcul terminé qui est au-dessus de $\text{CPD}_\sigma(t)$, pour un système de choix σ ; grâce à la relation : $\text{CPD}(t) \leq \delta$ et au fait que tout calcul terminé dérive toute occurrence semi-externe de $\underline{\text{or}}$ dans t , on peut montrer que $\sigma = \sigma^\delta$, ce qui entraîne le lemme suivant :

LEMME 3.6 :

$$\begin{aligned} \bigcup \{ \text{res}(\text{CPD}_{\sigma^\delta}(t)) / \delta \text{ terminé} \} \\ = \bigcup \{ \text{res}(\text{CPD}_\sigma(t)) / \sigma \text{ système de choix pour } \text{CPD}(t) \}. \end{aligned}$$

On en déduit la proposition :

PROPOSITION 3.2 : Pour tout arbre t de $M(F_+ \cup \Phi, X)$, on a :

$$\text{Val}_{\text{den}}(t) \approx \text{Val}_{\text{op}}(t) \quad \square$$

Et finalement le :

THÉORÈME 2 : *Pour tout arbre t dans $M(F_+ \cup \Phi, X)$:*

$$\text{Val}_{\text{op}}(t) \approx \text{Val}_{\text{den}}(t) \approx \text{Val}_{\text{alg}}(t).$$

4. SCHÉMAS DE GREIBACH

Nous allons à présent conclure ce travail en étudiant le cas particulier des schémas de Greibach.

DÉFINITION 4.1 : Arbres contractants.

Un arbre t de $M(F_+ \cup \Phi, X)$ est dit contractant ssi

$$t \in F_0 \cup X$$

ou

$$t = f(t_1, \dots, t_n) \text{ avec } f \in F_n \text{ et } t_i \in M(F_+ \cup \Phi, X)$$

ou

$$t = \underline{\text{or}}(t_1, t_2) \text{ avec } t_1 \text{ et } t_2 \text{ contractants. } \square$$

DÉFINITION 4.2 : Schémas de Greibach.

Un schéma de programme $\Sigma : \varphi_i(x_1, \dots, x_{n_i}) = \tau_i$ pour $1 \leq i \leq k$ est dit de Greibach si chaque arbre τ_i est contractant. \square

Dans la suite de ce chapitre, nous supposons que Σ est un schéma de Greibach.

LEMME 4.1 : *Pour tout $t \in M(F_+ \cup \Phi, X)$, on a $\text{Val}_{\text{den}}(t) = \text{Val}_{\text{alg}}(t)$.*

Nous ne donnons ici qu'une idée de la démonstration, qui est assez longue (on en trouvera les détails dans [9, 5]).

Nous avons vu, dans le chapitre 2, que la fonction γ n'est pas continue au voisinage de tout arbre.

En revanche, Arnold et Nivat, dans leurs travaux [9], ont montré qu'elle était continue au voisinage de tout arbre « strict » (arbre dont aucune branche ne contient une infinité de symboles or consécutifs), et que l'ensemble $\bigcap \{ \pi(\Sigma^n(t)) / n \in \mathbb{N} \}$ était un ensemble ne contenant qu'un seul arbre (strict d'ailleurs), quel que soit l'arbre t de départ, pourvu que le schéma soit de Greibach.

On peut alors écrire :

$$\gamma(\bigcap \{ \pi(\Sigma^n(t)) / n \in \mathbb{N} \}) = \bigcap \{ \gamma(\pi(\Sigma^n(t))) / n \in \mathbb{N} \},$$

et il ne reste plus qu'à montrer que les deux suites

$$(\gamma(\pi(\Sigma^n(t)))_{n \in \mathbb{N}} \quad \text{et} \quad (\pi_D(\Sigma^n(t)))_{n \in \mathbb{N}}$$

ont même limite, ce qui découle encore du fait que le schéma est de Greibach. \square

DÉFINITION 4.3 : Calculs réussis.

Nous dirons qu'un calcul δ à partir de l'arbre t est réussi ssi $\text{res}(\delta)$ est un singleton. \square

LEMME 4.2 : Si δ est un calcul terminé dans un schéma de Greibach, alors δ est un calcul réussi.

Cela provient du fait, déjà cité dans le chapitre 3, que tout calcul terminé dérive toute occurrence semi-externe de symbole or ou de symbole non terminal. \square

LEMME 4.3 : Si σ est un système de choix sur $\text{CPD}(t)$, alors $\text{CPD}_\sigma(t)$ est un calcul réussi.

$\text{CPD}_\sigma(t)$ dérive, pour les mêmes raisons que celles évoquées ci-dessus, toutes les occurrences semi-externes de or ou de symboles non terminaux (appartenant à Φ). \square

LEMME 4.4 : Pour tout $t \in M(F_+ \cup \Phi, X)$, on a $\text{Val}_{\text{den}}(t) = \text{Val}_{\text{op}}(t)$.

Démonstration : D'après le lemme 3.5,

$$\text{Val}_{\text{den}}(t) = \cup \{ \text{res}(\text{CPD}_\sigma(t)) / \sigma \text{ système de choix pour } \text{CPD}(t) \}$$

De plus $\text{CPD}_\sigma(t)$ est un calcul réussi (lemme 4.3), et donc, si δ est un calcul terminé, plus grand que $\text{CPD}_\sigma(t)$ (i.e. $\text{CPD}_\sigma(t) \leq \delta$), alors :

$$\text{res}(\text{CPD}_\sigma(t)) \supseteq \text{res}(\delta) \neq \emptyset.$$

Or $\text{res}(\text{CPD}_\sigma(t))$ est un singleton, par conséquent $\text{res}(\text{CPD}_\sigma(t)) = \text{res}(\delta)$.

Finalement, nous obtenons, pour tout σ :

$$\text{Val}_{\text{den}}(t) \supseteq \text{Val}_{\text{op}}(t) \supseteq \text{res}(\text{CPD}_\sigma(t)),$$

ce qui montre le résultat. \square

Nous déduisons des lemmes 4.1 et 4.4 le :

THÉORÈME 3 : Si Σ est un schéma de Greibach, et si t est un arbre de $M(F_+ \cup \Phi, X)$, alors :

$$\text{Val}_{\text{op}}(t) = \text{Val}_{\text{den}}(t) = \text{Val}_{\text{alg}}(t).$$

REMARQUE : Dans leurs travaux, Arnold et Nivat [7, 8], montrent que, dans le cas des schémas de Greibach, la sémantique des calculs réussis est égale à la sémantique dénotationnelle. Ce résultat a été montré aussi bien dans le cas d'un domaine métrique [8] que dans le cas d'un domaine ordonné.

Le théorème 3 montre donc que, dans le cas des schémas de Greibach, **sémantique des calculs réussis, sémantique des calculs terminés** (Val_{op}), **sémantique dénotationnelle** (Val_{den}), et **sémantique algébrique** (Val_{alg}) **s'unifient totalement.**

BIBLIOGRAPHIE

1. S. ABRAMSKY, *On Semantics Foundations for Applicative Multiprogramming*, Lecture notes in Computer Science, vol. 137, 1982.
2. A. ARNOLD, *Sémantique de l'appel par valeur*, RAIRO Informatique théorique, vol. 12, n° 2, 1978.
3. A. ARNOLD, *Schémas de programmes non déterministes avec appel synchrone*, in *Program transformations*, 3rd international symposium on programming, Dunod, Paris.
4. A. ARNOLD et M. DAUCHET, *Théorie des magmoïdes I*, RAIRO Informatique théorique, vol. 12, n° 3, 1978.
5. A. ARNOLD, P. NAUDIN et M. NIVAT, *On Semantics of Non-Deterministic Recursive Program Schemes*, in *Algebraic Methods in Semantics*, NIVAT and REYNOLDS Eds., Cambridge University Press.
6. A. ARNOLD et M. NIVAT, *The Metric Space of Infinite Trees. Algebraic and Topological Properties*, *Fundamenta Informaticae*, vol. 4, 1980, p. 445-476.
7. A. ARNOLD et M. NIVAT, *Formal Computations of Non-Deterministic Recursive Program Schemes*, *Math. System Theory*, vol. 13, 1980, p. 219-236.
8. A. ARNOLD et M. NIVAT, *Metric Interpretations of Infinite Trees and Semantics of Non-Deterministic Recursive Program Schemes*, *Theoretical Computer Science*, vol. 11, 1980, p. 181-205.
9. A. ARNOLD et M. NIVAT, *Fair or-Trees and Non-Deterministic Program Schemes*, In 2nd International Workshop on Semantics, Bad-Honnef, 1979, *Bulletin of EATCS*, vol. 8, 1979, p. 104-105.
10. G. BOUDOL, *Sémantique opérationnelle et algébrique des programmes récursifs non déterministes*. Thèse d'État, Université de Paris-VII, 1980.
11. G. BOUDOL, *Une sémantique pour les arbres non déterministes*, In 6th CAAP, Lecture Notes in Computer Science, vol. 112, 1981, p. 147-161.
12. G. BOUDOL, *Computational Semantics of Term Rewriting Systems*, in *Algebraic Methods in Semantics*, NIVAT and REYNOLDS Eds., Cambridge University Press.
13. I. GUESSARIAN, *Algebraic Semantics*, Lecture Notes in Computer Science, vol. 99, Springer-Verlag, 1981.
14. M. C. B. HENNESSY, *Powerdomains and Non Deterministic Definitions*, Lecture Notes in Computer Science, vol. 137, 1982.
15. M. C. B. HENNESSY et G. D. PLOTKIN, *Full Abstraction for a Simple Parallel Programming Language*, Lecture Notes in Computer Science, vol. 74, 1979.
16. P. NAUDIN, *Le problème du non déterminisme dans la sémantique algébrique des schémas de programmes*, Thèse de 3^e cycle, Université de Poitiers, 1982.

17. M. NIVAT, *On the Interpretation of Recursive Polyadic Program Schemes*, Symposia Matematica, vol. 15, 1975, p. 255-281.
18. M. NIVAT, *Interprétation universelle d'un schéma de programmes rékursifs*, Informatica, vol. 7, 1977, p. 9-16.
19. M. NIVAT, *Non-Deterministic Programs: an Algebraic Overview*, Proceedings of IFIP Congress, 1980, North Holland.
20. D. NIWINSKI, *Fixed-Point Characterization of Context-Free ∞ -Languages*, Information and Control, vol. 61, n° 3, 1984.
21. G. PLOTKIN, *A Power Domain Construction*, SIAM Journal on Computing, vol. 5, 1976, p. 452-486.
22. A. POIGNÉ, *On Effective Computations of Non-Deterministic Schemes*, Lecture Notes in Computer Science, vol. 137, 1982.
23. N. POLIAN, *Différents types de dérivations infinies dans les grammaires algébriques d'arbres*, in 6th CAAP, Lectures Notes in Computer Science, vol. 112, 1981, p. 340-349.
24. M. SMYTH, *Powerdomains*, JCSS, vol. 16, 1978, p. 23-26.
25. D. SCOTT, *The Lattice of Flow Diagrams*, Symposium on Semantics of Algorithmic Languages, Lecture Notes in Mathematics, vol. 182, Springer-Verlag, 1971.