RAIRO. INFORMATIQUE THÉORIQUE

IRÈNE GUESSARIAN

Les tests et leur caractérisation syntaxique

RAIRO. Informatique théorique, tome 11, n° 2 (1977), p. 133-156

<http://www.numdam.org/item?id=ITA_1977__11_2_133_0>

R.A.I.R.O. Informatique théorique/Theoretical Computer Science
(vol. 11, n° 2, 1977, p. 133 à 156).

LES TESTS ET LEUR CARACTÉRISATION SYNTAXIQUE (1)

par Irène GUESSARIAN (2)

Communiqué par M. NIVAT

Résumé. — *Nous présentons ici une contribution à l'étude de l'équivalence des schémas de programme « sous conditions ».*

1. INTRODUCTION

Il est admis que la programmation n'est plus l'art d'écrire d'abord un programme approximatif, puis de le mettre au point par essais successifs en machine. On tend aujourd'hui, au contraire, à mettre l'accent sur la nécessité, d'une part d'écrire la preuve qu'un programme est correct (i.e. qu'il se termine, fait bien ce à quoi on le destine, etc...) en même temps que le programme lui-même ([11, 33]), d'autre part d'obtenir des programmes optimaux et de le démontrer ([2, 3, 13]). Pour ce faire, on doit bien sûr éliminer les spécifications particulières d'un langage donné ou d'une machine donnée et ne garder que les caractéristiques et les structures essentielles du programme. On rejoint ainsi le cadre de la théorie élaborée par Ianov ([19]) à la fin des années cinquante, reprise quelques années plus tard par Rutledge et Mac Carthy ([22, 30]) et largement développée ensuite ([12, 23, 25, 28, 29, 31]).

1. Schémas et interprétations

En effet, nombre de propriétés couramment étudiées des programmes (l'élimination des appels récursifs ([17, 32]), des branchements ([2, 3, 13]), etc...) ne dépendent pas de la nature explicite, par exemple arithmétique ou logique, des opérations et des données du programme, mais seulement de sa structure. C'est pourquoi Ianov ([19]) a introduit les schémas de programme en dissociant les deux éléments sous-jacents à la notion de programme :

— au niveau de la structure, le *schéma de programme*, qui représente intuitivement la structure de contrôle (branchements, etc...) du programme qui demeure lorsqu'on « oublie » la nature et le sens des fonctions de base et des variables et qu'on les considère comme des symboles purement formels.

(1) Reçu mai 1976 et en version définitive janvier 1977.

(2) CNRS et Laboratoire d'Informatique Théorique et de Programmation, Université Paris VII, Tour 45-55.

– au niveau de la sémantique, l'*interprétation*, qui restitue l'information nécessaire sur la nature des symboles formels, pour que le schéma devienne un programme susceptible de faire un calcul.

Donnons un exemple qui illustre la relation entre un programme et les deux éléments – schéma de programme et interprétation – qui le constituent. Considérons le programme suivant, dû à Arsac ([2]) et écrit sous la forme d'une procédure récursive dans un langage simplifié proche d'Algol :

P : $\varphi(n, k) = si\ k = 1\ alors\ n$
$\quad sinon\ si\ k$ pair $alors\ \varphi(n \times n, k/2)$
$\quad\quad sinon\ n \times \varphi(n \times n, k/2)$

P est associé au schéma de programme :

Σ : $\varphi(n, k) = si\ k = a\ alors\ n$
$\quad sinon\ si\ t(k)\ alors\ \varphi(m(n, n), d(k))$
$\quad\quad sinon\ m(n, \varphi(m(n, n), d(k)))$

et à l'interprétation I, qui est définie par :

– a, k et n prennent leurs valeurs dans $\mathbf{N}$

– $\forall x, y \in \mathbf{N}$ $\quad I(a) = 1$
$\quad I(d)(x) = x/2$ $\quad$ (quotient entier de x par 2)
$\quad I(m)(x, y) = x \times y$

$$I(t)(x) = \begin{cases} \text{vrai} & \text{si } x \text{ est pair} \\ \text{faux} & \text{si } x \text{ est impair} \end{cases}$$

L'interprétation I transforme le schéma Σ en le programme P, qui calcule n^k pour $k > 0$ et diverge pour $k = 0$.

Nombreux sont les auteurs qui se sont penchés sur ces schémas ([4, 21, 31]), leur sémantique ([10, 15]) et ses applications : équivalences de programme ([7, 29]), élimination de la récursivité ([17, 32]), vérification de programmes ([24]), etc... En effet, les propriétés d'un programme indépendantes de la sémantique sont de nature syntaxique et seront donc prouvées de manière plus simple et plus naturelle au niveau de la syntaxe – c'est-à-dire au niveau du schéma de programme. Elles s'appliqueront alors à tous les programmes déduits du schéma de programme au moyen d'une interprétation ([23]).

Si par exemple on considère la fonction calculée par le programme P comme une opération binaire dont on veut prouver la distributivité à droite par rapport à la multiplication, on s'aperçoit que cela résulte des propriétés d'associativité et de commutativité de la multiplication. On peut donc faire cette preuve au niveau syntaxique, c'est-à-dire pour le schéma Σ : on montre ainsi que, si m est associative et commutative, alors la fonction calculée par Σ, considérée

comme une opération binaire, est distributive à droite par rapport à m. On peut alors prendre une autre interprétation I' de Σ, en décidant par exemple que I' ne diffère de I que par $I'(m)$ qui est l'addition. Σ s'interprète alors par I' en P' :

$$P' : \quad \psi(n, k) = \textit{si } k = 1 \textit{ alors } n$$
$$\textit{sinon si } k \text{ pair } \textit{alors } \psi(n + n, k/2)$$
$$\textit{sinon } n + \psi(n + n, k/2)$$

P' calcule $n \times k$ pour $k > 0$ et diverge pour $k = 0$.

Le résultat de distributivité prouvé pour Σ s'interprète aussitôt par I' et permet d'affirmer, sans autre preuve, que l'associativité et la commutativité de l'addition impliquent que la multiplication est distributive à droite par rapport à l'addition.

On voit ainsi l'avantage d'exprimer les propriétés des programmes au niveau syntaxique. C'est pourquoi on est conduit ([26]) à se placer à un niveau d'abstraction légèrement supérieur en n'interprétant pas non plus les tests (i.e. les *si ... alors ... sinon ...*) et à considérer les schémas de programme les plus généraux.

Le programme P est associé dans ce cadre au schéma :

$$\Sigma' : \varphi(n, k) = g(e(k, a), n, g(t(k), \varphi(m(n, n), d(k)), m(n, \varphi(m(n, n), d(k)))))$$

et à l'interprétation I_1 définie comme I avec de plus :

$$\forall x, y, z \in \mathbf{N} \quad , \quad \forall u \in \{ \text{vrai, faux} \}$$

$$I_1(e)(x, y) = \begin{cases} \text{vrai} & \text{si } x = y \\ \text{faux} & \text{sinon} \end{cases}$$

$$I_1(g)(u, y, z) = \begin{cases} y & \text{si } u = \text{vrai} \quad (1) \\ z & \text{si } u = \text{faux} \end{cases}$$

On peut alors prouver dans ce cadre, la distributivité de la solution de Σ' par rapport à m, en supposant, outre l'associativité et la commutativité de m, que la fonction g est interprétée en sorte que :

$$\forall u, x, y, z \qquad m(g(u, x, y), z) = g(u, m(x, z), m(y, z))$$

C'est la seule propriété des tests qui nous soit nécessaire ici.

Cette théorie — dite algébrique — de la sémantique des schémas de programme, introduite par Nivat ([26, 27]), trouve de nombreuses applications

(1) Pour simplifier et clarifier cette introduction, nous omettons momentanément l'élément indéfini (qui rend compte des valeurs pour lesquelles les programmes divergent), nécessaire dans un exposé rigoureux.

dans des domaines aussi variés que l'étude des langages algébriques et des langages d'arbres ([1, 6]) ainsi bien sûr que dans beaucoup de problèmes spécifiques à la programmation, par exemple des problèmes d'affectation et de passage des paramètres ([20]), d'équivalence et de transformation de programmes ([16, 18]), d'optimisation du calcul d'un programme ([5]).

2. Équivalences sémantiques et expression syntaxique

Nous allons nous intéresser plus particulièrement ici à l'un des problèmes les plus importants en théorie de la programmation, celui de l'équivalence sémantique ; dans la suite, $\mathscr{D}$ désigne la classe de toutes les interprétations (*cf.* déf. 3), $\mathscr{C}$ une partie de $\mathscr{D}$ et nous dirons que $\mathscr{C}$ est une famille d'interprétations. Deux schémas de programme sont sémantiquement équivalents modulo une famille $\mathscr{C}$ d'interprétations (en abrégé $\mathscr{C}$-équivalents ou $\equiv \mathscr{C}$), si et seulement si ils calculent la même fonction pour toute interprétation prise dans $\mathscr{C}$ (*cf.* [18]). Nous considèrerons, entre autres, la famille $\mathscr{D}$, la famille $\mathscr{T}$ des interprétations à valeurs dans un domaine totalement ordonné (Th. 2) et la famille $\mathscr{I}_d$ des interprétations discrètes (*cf.* déf. 7) ; on sait en effet ([27]) que ces dernières sont les plus fréquentes et les plus importantes dans la pratique.

Considérons l'exemple suivant qui illustre la notion de $\mathscr{C}$-équivalence. Soient Σ et Σ' les schémas :

$$\Sigma : \quad \varphi(v) = g'(v, v, \varphi(h(v)))$$
$$\Sigma' : \quad \psi(v) = g'(v, v, g'(v, \Omega, \psi(h(v))))$$

Σ et Σ' sont équivalents pour toute interprétation I telle que $I(g')$ soit un test (*cf.* déf. 6), mais ne sont pas équivalents pour toute interprétation. Soit en effet I l'interprétation qui prend ses valeurs dans le domaine $D = \mathbf{N}^2 \cup \{ \omega \}$, muni de l'ordre discret de plus petit élément ω (*cf.* déf. 7), et est définie par : $\forall x, y, z \in D$

$$I(g')(x, y, z) = \begin{cases} y & \text{si } x = (p, n) \in \mathbf{N}^2 \quad \text{et} \quad p > n \\ 2z & \text{si } x = (p, n) \in \mathbf{N}^2 \quad \text{et} \quad p \leqq n \\ \omega & \text{sinon} \end{cases}$$

$$I(h)(x) = \begin{cases} (p, n - p) & \text{si } x = (p, n) \in \mathbf{N}^2 \quad \text{et} \quad p \leqq n \\ \omega & \text{sinon} \end{cases}$$

Pour cette interprétation Σ calcule la fonction qui au couple (p, n) associe le couple $2^{n/p}(p, n \bmod p)$, et Σ' calcule la fonction qui au couple (p, n) associe le couple $4^{n/p}(p, n \bmod p)$, où n/p (resp. $n \bmod p$) désigne le quotient (resp. le reste) de la division entière de n par p.

Pour pouvoir étudier plus facilement l'équivalence sémantique de deux schémas, nous allons chercher à la caractériser au niveau syntaxique, i.e. au

niveau du schéma de programme ; cela nous permettra de prouver cette équivalence en ne calculant que sur les schémas, sans considérer aucune interprétation.

L'équivalence modulo $\mathscr{D}$ — i.e. l'équivalence pour toute interprétation, donc la plus forte — s'exprime aisément au niveau syntaxique en termes de propriétés de F-algèbres (*cf.* déf. 1) ou F-magmas (*cf.* [26]) : en effet, si F' (resp. F) désigne l'ensemble des symboles formels (resp. de fonction) figurant dans un schéma Σ, le comportement de Σ est caractérisé par une partie de l'ensemble des termes bien formés sur l'alphabet F' — qui est une F-algèbre libre ([16, 26]) ; deux schémas sont équivalents modulo $\mathscr{D}$ si et seulement si ils sont caractérisés par la même partie.

Toutefois cette équivalence est trop fine dans la pratique, où l'on a toujours à traiter d'équivalences modulo des sous-familles particulières de $\mathscr{D}$. On s'intéressera donc en général à des équivalences « *sous conditions* », les conditions étant, par exemple, que certain symbole de fonction est toujours interprété commutativement, ou associativement, ou comme un test (*cf.* déf. 6), ou encore que le domaine des interprétations est toujours un domaine discret (*cf.* déf. 7), etc... Pour obtenir une expression syntaxique de l'équivalence de schémas sous conditions, analogue à la caractérisation de la $\mathscr{D}$-équivalence, il faut donc traduire ces conditions au niveau syntaxique — i.e. au niveau de la F-algèbre. Pour un grand nombre de conditions cette caractérisation syntaxique (ou « algébrique ») s'obtient de manière immédiate et tout à fait naturelle. Soit par exemple f un symbole de fonction binaire, la condition « $I(f)$ est associative » s'exprime syntaxiquement par la congruence de F-algèbre (*cf.* déf. 2) engendrée par la relation :

$$f(f(u, v), w) = f(u, f(v, w))$$

Nous traduisons cela en disant que la condition « $I(f)$ est associative » est congruentielle relativement à la famille $\mathscr{D}$ d'interprétations — en abrégé est $\mathscr{D}$-congruentielle ou congruentielle (*cf.* déf. 5) — et que la congruence C_a qui la caractérise est engendrée par le seul axiome : $f(f(u, v), w) C_a f(u, f(v, w))$. De manière équivalente nous dirons aussi que la famille d'interprétations définie par la condition « $I(f)$ est associative » est congruentielle.

Une des motivations du présent article est le résultat suivant : l'équivalence sémantique modulo une famille $\mathscr{C}$ d'interprétations qui est congruentielle est exprimable syntaxiquement : comme dans le cas de l'équivalence sémantique pour toute interprétation, on caractérise un schéma de programme Σ par la partie qu'il engendre dans une F-algèbre libre ; on montre ensuite que deux schémas de programme sont sémantiquement équivalents modulo $\mathscr{C}$ si et seulement si les parties correspondantes se déduisent l'une de l'autre par la congruence qui définit $\mathscr{C}$ (*cf.* [18]). Ceci s'applique en particulier aux deux schémas Σ et Σ' déjà considérés. On en déduit, ce qui est intuitivement clair,

que ces deux schémas sont équivalents modulo $\mathscr{C}$, où $\mathscr{C}$ est la sous-famille de $\mathscr{D}$ définie par la congruence C engendrée par l'axiome :

$$g(v, v, g(v, \Omega, w))\, C\, g(v, v, w).$$

$\mathscr{C}$ est par sa définition même congruentielle; de plus, toute interprétation I telle que $I(g)$ soit un test appartient à $\mathscr{C}$ (*cf.* déf. 8 et th. 1), donc Σ et Σ' sont *a fortiori* équivalents pour toute I telle que $I(g)$ soit un test.

Lorsqu'on s'intéresse, non plus à la famille $\mathscr{D}$ toute entière, mais à une sous-famille $\mathscr{C}$ (ce sera souvent la famille $\mathscr{I}_d$ des interprétations discrètes), la notion de propriété ou de condition congruentielle est remplacée par celle de propriété ou de condition congruentielle relativement à $\mathscr{C}$ (*cf.* déf. 5) — en abrégé $\mathscr{C}$-congruentielle. Il est clair qu'une condition (ou une propriété) est congruentielle si et seulement si elle est $\mathscr{C}$-congruentielle pour toute famille d'interprétations $\mathscr{C}$.

Par exemple, la propriété « $I(f)$ est commutative » est $\mathscr{C}$-congruentielle pour toute $\mathscr{C}$. Elle est caractérisée par la congruence engendrée par le seul axiome : $f(u, v)\, C_c\, f(v, u)$. Soit de plus f_0 un symbole d'arité 0 dans F, la propriété « $I(f_0)$ est élément neutre pour la loi de composition $I(f)$ » est $\mathscr{C}$-congruentielle pour toute $\mathscr{C}$. Par contre on démontre (*cf.* [16]) que si h est un symbole de fonction unaire la propriété : « dom (I) muni de la loi de composition $I(f)$ est un groupe (resp. un groupe abélien) d'élément neutre $I(f_0)$ et où $I(h)(d)$ est le symétrique de d pour tout d dans dom(I) » n'est $\mathscr{C}$-congruentielle pour aucune $\mathscr{C}$ non triviale ($\mathscr{C}$ est triviale si elle est réduite à une seule interprétation de domaine réduit au seul élément ω).

On trouvera d'autres exemples et contre-exemples ultérieurement (*cf.* §2, §3 prop. 2 et 3).

Un des résultats du présent article est de montrer qu'une condition non congruentielle peut toutefois être $\mathscr{C}$-congruentielle pour les familles $\mathscr{C}$ auxquelles on s'intéresse (*cf.* th. 1 et th. 2). Ceci nous permet d'obtenir une caractérisation syntaxique de l'équivalence sous conditions pour les familles d'interprétations et les conditions auxquelles on s'intéresse. Par exemple, le théorème 1 nous permet d'affirmer que la condition P_t : « $I(g)$ est un test total » est $\mathscr{I}_d$-congruentielle bien qu'elle ne soit pas congruentielle (*cf.* corollaire de la prop. 2). Nous savons d'autre part que l'équivalence sémantique modulo $\mathscr{I}_d$ s'exprime syntaxiquement (*cf.* [18]) et que l'on peut généraliser le résultat de caractérisation syntaxique de l'équivalence modulo les familles congruentielles à l'équivalence modulo les familles $\mathscr{I}_d$-congruentielles (*cf.* [9, 16]). Nous en déduisons donc que l'équivalence sémantique pour toute I à valeurs dans un domaine discret et telle que $I(g)$ soit un test total est exprimable syntaxiquement. Ce résultat reste vrai si on remplace la famille $\mathscr{I}_d$ par la famille $\mathscr{T}$ (*cf.* th. 2).

3. Plan du présent travail

Dans cet article, nous étudions l'expression syntaxique de l'une des propriétés les plus importantes en programmation, la condition : P : « l'interprétation I est telle que $I(g)$ soit un test » (*cf.* déf. 6) — où g est un symbole de fonction ternaire fixé. Les tests sont en effet privilégiés dans pratiquement tous les langages de programmation sous forme de conditionnelles « *si ... alors ... sinon ...* ». Nous donnons dans la proposition 2 une condition nécessaire sur les familles $\mathscr{C}$ d'interprétations pour que la condition P (ou la condition P_t énoncée ci-dessous) soit $\mathscr{C}$-congruentielle. Nous remarquons que la plus usuelle des familles d'interprétations, la famille $\mathscr{I}_d$ des interprétations discrètes, satisfait à cette condition nécessaire et nous montrons que, bien que P ne soit pas $\mathscr{I}_d$-congruentielle (*cf.* th. 3), P_t : « $I(g)$ est un test total » est $\mathscr{I}_d$-congruentielle (*cf.* th. 1). Nous disons qu'un test est total si, intuitivement, pour une valeur définie de la variable testée, il ne prend que les valeurs vrai ou faux et jamais la valeur indéterminé. La congruence C_t qui caractérise P_t relativement à la famille $\mathscr{I}_d$, caractérise encore P_t relativement à la famille $\mathscr{T}$ des interprétations à valeurs dans un ensemble totalement ordonné (*cf.* th. 2).

Nous donnons ensuite des exemples de familles $\mathscr{C}$ relativement auxquelles les conditions P et P_t ne sont pas congruentielles (prop. 2 et cor., prop. 3, th. 3 et 3 bis).

Nous montrons enfin que la condition P : « $I(g)$ est un test », bien qu'elle ne soit pas $\mathscr{I}_d$-congruentielle, admet une caractérisation syntaxique relativement à la famille $\mathscr{I}_d$ des interprétations discrètes (*cf.* th. 4). Toutefois, cette caractérisation, plus complexe qu'une simple congruence, fait intervenir une propriété du second ordre des tests qui contient un quantificateur universel portant sur l'ensemble des fonctions du domaine de l'interprétation dans lui-même (*cf.* déf. 10 et déf. 11, 2°). Il nous semble plus difficile d'obtenir dans ce cas, par les méthodes précédemment esquissées, une caractérisation syntaxique de l'équivalence sémantique modulo les interprétations discrètes telles que $I(g)$ soit un test.

2. DÉFINITIONS

Soit F un ensemble de symboles de fonctions : chaque f de F est muni d'une arité $a(f)$ avec $a(f)$ entier. F contient un symbole privilégié Ω d'arité 0 (le symbole « indéterminé »).

DÉFINITION 1 : Une *F-algèbre* est un ensemble D, avec, pour chaque f dans F, une application totale $\mu(f)$ de $D^{a(f)}$ dans D.

Un *morphisme* φ d'une F-algèbre D dans une F-algèbre D' est une application telle que :

$$\forall f \in F \quad \varphi(\mu(f)(d_1, \ldots, d_{a(f)})) = \mu'(f)(\varphi(d_1), \ldots, \varphi(d_{a(f)}))$$

Soit V un ensemble, disjoint de F, de symboles de variables. Il existe une F-algèbre libre unique (à isomorphisme près) contenant V : c'est l'ensemble des termes bien formés sur l'alphabet $V \cup F$ (*cf.* [8]) : nous la notons $M(F, V)$. Nous considérons sur $M(F, V)$ l'ordre le plus grossier compatible avec la structure de F-algèbre et ayant Ω pour plus petit élément (i.e. $\Omega \not\geqslant v \; \forall v \in V$). Cet ordre est noté $\prec$.

DÉFINITION 2 : Une *congruence* sur une F-algèbre est une relation d'équivalence compatible avec la structure de F-algèbre.

Nous nous intéresserons surtout aux congruences sur $M(F, V)$ engendrées par un ensemble fini d'axiomes; soit S — l'ensemble d'axiomes — une partie de $M(F, V)$, la congruence engendrée par S est la plus petite congruence contenant S.

Un domaine partiellement ordonné complet (ou *poc*) est un ensemble D, muni d'un ordre $\leqslant$, ayant un plus petit élément ω, et dont toute partie filtrante à droite admet une borne supérieure. (Une partie E de D est filtrante à droite si :

$$\forall\, d, \quad d' \in E \quad \exists\, d'' \in E \quad d \leqslant d'' \qquad \text{et} \qquad d' \leqslant d'')$$

Conformément à la terminologie usuelle dans ce domaine, nous appellerons *dirigées* les parties filtrantes à droite.

DÉFINITION 3 : Une *interprétation dirigée* I de F est la donnée :

— d'un poc D, appelé *domaine* de I, noté dom (I).

— d'une structure de F-algèbre définie sur D par :

. pour toute f dans F une application totale et continue $I(f)$ de $D^{a(f)}$ dans D

. avec : $I(\Omega) = \omega$.

Rappelons que $I(f)$ est continue si, pour toutes parties dirigées $E_1, \ldots, E_{a(f)}$ de D,

$$\{ I(f)(d_1, \ldots d_{a(f)}) / d_i \in E_i, i = 1, \ldots, a(f) \}$$

est une partie dirigée de D et :

$$I(f)(UE_1, \ldots, UE_{a(f)}) = U \{ I(f)(d_1, \ldots, d_{a(f)}) / d_i \in E_i, i = 1, \ldots, a(f) \}$$

où UE désigne la borne supérieure des éléments de E.

Notation : $\mathcal{D}$ désigne la famille des interprétations dirigées.

Toutes les interprétations auxquelles nous nous intéresserons dans la suite sont dans $\mathcal{D}$; nous omettrons donc le qualificatif « dirigée ».

Une *valuation* I_V associée à I est une application totale de V dans le domaine D de I.

Toute valuation I_V associée à I se prolonge de manière unique en un morphisme, encore noté I_V, de la F-algèbre libre $M(F, V)$ dans D. Intuitivement,

$M(F, V)$ représente les schémas de programme sans boucle et le prolongement de I_V à $M(F, V)$ permet de définir la fonction calculée par un schéma de programme sans boucle. Plus précisément, soit m dans $M(F, V)$; la donnée de I et I_V permet de transformer m en un programme (*cf.* Introduction); $I_V(m)$ est alors la valeur calculée par ce programme (*cf.* [26]).

DÉFINITION 4 : Une interprétation I est *compatible* avec une congruence C sur $M(F, V)$ si pour toute valuation I_V associée à I, C est contenue dans le noyau du morphisme I_V, soit :

$$\forall m, m' \in M(F, V) : \qquad mCm' \Rightarrow I_V(m) = I_V(m')$$

DÉFINITION 5 : Soit $\mathscr{C}$ une famille d'interprétations de F contenue dans la famille $\mathscr{D}$ des interprétations dirigées. Une propriété P des interprétations de F est *congruentielle* (resp. *$\mathscr{C}$-congruentielle*) si il existe une congruence C sur $M(F, V)$ telle que :

Pour toute I dans $\mathscr{D}$: I vérifie P ssi I est compatible avec C.

(resp :

Pour toute I dans $\mathscr{C}$: I vérifie P ssi I est compatible avec C).

Exemples et remarque : On vérifie facilement que nombre de propriétés usuelles sont congruentielles. Soient par exemple f et h deux symboles de fonction binaires; la propriété « $I(f)$ est distributive par rapport à $I(h)$ » est $\mathscr{C}$-congruentielle pour toute famille $\mathscr{C}$, et la congruence C_d qui la caractérise est clairement engendrée par le seul axiome :

$$f(u, h(v, w))\, C_d\, f(h(u, v), h(u, w)) \qquad \text{pour} \quad u, v, w \quad \text{dans} \quad V.$$

Par contre, la condition « I est une interprétation discrète » n'est pas $\mathscr{D}$-congruentielle (*cf.* [16]).

Ces exemples, ainsi que ceux donnés dans l'introduction montrent qu'en général, et intuitivement, une propriété que l'on peut définir par quantification universelle sur les éléments des domaines $\mathrm{dom}(I)$ des interprétations I considérées sera $\mathscr{C}$-congruentielle; toutefois la compatibilité avec l'ordre de $\mathrm{dom}(I)$ ajoutera souvent de fortes contraintes sur la structure de $\mathrm{dom}(I)$. Par contre, les propriétés faisant intervenir des quantifications existentielles seront plus rarement $\mathscr{C}$-congruentielles (*cf.* [16], ch. 4).

Il est extrêmement utile de savoir exprimer des propriétés des familles d'interprétations par des congruences. En effet, en théorie des schémas de programme, on n'utilise pratiquement jamais la famille de toutes les interprétations dirigées, mais presque toujours des familles d'interprétations définies par certaines propriétés (par exemple, un symbole de fonction est interprété associativement, ou commutativement, ou comme un test, ou les domaines des interprétations considérées sont discrets — *cf.* déf. 7 —, etc...). On a donc à

étudier l'équivalence des schémas de programme non pas dans toute sa généralité (i.e. pour toute interprétation dirigée) mais sous certaines conditions (qui expriment les propriétés de la famille d'interprétations étudiée) : par exemple, $f(f(u, v), w)$ et $f(u, f(v, w))$ ne sont pas équivalents pour toute interprétation dirigée, mais sont équivalents pour toutes les interprétations où $I(f)$ est associative.

Soit $\mathscr{C}'$ une famille d'interprétations définie par certaines conditions. Un problème central en théorie de la programmation est donc l'étude de l'équivalence de deux schémas de programme pour toutes les interprétations de $\mathscr{C}'$. Rappelons que deux schémas de programmes sont équivalents pour toutes les interprétations de $\mathscr{C}'$ si et seulement si les programmes associés calculent la même chose pour toute interprétation de $\mathscr{C}'$ (*cf.* Introduction). Or, (*cf.* [16]), pour les familles $\mathscr{C}'$ qui sont congruentielles (et pour nombre de familles $\mathscr{C}'$ qui sont $\mathscr{C}$-congruentielles pour des familles $\mathscr{C}$ usuelles), on peut exprimer l'équivalence de deux schémas de programme pour toutes les interprétations de la famille $\mathscr{C}'$ au niveau des approximations finies (par des schémas de programme sans boucle, c'est-à-dire des éléments de $M(F, V)$) des calculs effectués par ces schémas. Cela donne un moyen effectif de vérifier pratiquement que deux schémas de programme sont équivalents pour toute interprétation de $\mathscr{C}'$, bien que ce problème soit habituellement très complexe : c'est en particulier le cas lorsque $\mathscr{C}'$ est une famille d'interprétations où un symbole de fonction ternaire donné est toujours interprété comme un test (*cf.* [14, 21]). Nous allons nous intéresser à ce problème dans la suite de cet article et chercher pour quelles familles $\mathscr{C}$ la propriété, pour un symbole de fonction donné, d'être interprété comme un test est $\mathscr{C}$-congruentielle. Il nous faut au préalable encore quelques définitions.

DÉFINITION 6 : Soit D un poc. Nous appelons *test* une application continue $\bar{g}$ de D^3 dans D telle qu'il existe une partition $\{ V, F, N \}$ de D avec :

$$\forall x, y, z \in D \quad , \quad \bar{g}(x, y, z) = \begin{cases} y & \text{si} \quad x \in V \\ z & \text{si} \quad x \in F \\ \omega & \text{si} \quad x \in N \end{cases} \tag{1}$$

Suivant l'usage, on introduit alors un prédicat unaire p sur D, défini par :

$$\forall x \in D \quad p(x) = \begin{cases} \text{vrai} & \text{si} \quad x \in V \\ \text{faux} & \text{si} \quad x \in F \\ \omega & \text{si} \quad x \in N \end{cases} \tag{2}$$

et on traduit (1) sous la forme :

$$\bar{g}(x, y, z) = \text{si} \quad p(x) \quad \text{alors} \quad y \quad \text{sinon} \quad z$$

Un test $\bar{g}$ est *total* ssi $N = \{ \omega \}$.

Exemples : Prenons pour domaine $D = \mathbf{N} \cup \{ \omega \}$, muni de l'ordre discret de plus petit élément ω (*cf.* déf. 7). Soient :

$$\bar{g}(x, y, z) = \begin{cases} y & \text{si } x \text{ est pair} \\ z & \text{si } x \text{ est impair} \\ \omega & \text{si } x = \omega \end{cases}$$

$$\bar{g}'(x, y, z) = \begin{cases} y & \text{si } x \text{ est pair} \\ z & \text{si } x \text{ est de la forme } 4n + 1 \\ \omega & \text{sinon} \end{cases}$$

$\bar{g}$ est un test total mais non $\bar{g}'$.

Dans la suite, g est un symbole de fonction d'arité 3 fixé dans F. Nous cherchons pour quelles familles $\mathscr{C}$ les propriétés :

$$P = \{ I(g) \quad \text{est un test} \}$$
$$P_t = \{ I(g) \quad \text{est un test total} \}$$

sont $\mathscr{C}$-congruentielles.

On démontre (*cf.* p. 148) que P (resp. P_t) n'est pas congruentielle. Toutefois, comme nous l'avons déjà remarqué, la famille $\mathscr{D}$ de toutes les interprétations dirigées est trop générale, car en pratique on a toujours à travailler avec des sous-familles $\mathscr{C}$ de $\mathscr{D}$. Nous obtenons en particulier des résultats lorsque $\mathscr{C}$ est la famille des interprétations discrètes, les plus usuelles en programmation (*cf.* [26]). Nous étudions séparément dans les deux paragraphes suivants les propriétés P et P_t : nous verrons en effet que leurs comportements sont très différents.

3. LES TESTS TOTAUX

Nous cherchons à voir dans ce paragraphe pour quelles familles $\mathscr{C}$ P_t est $\mathscr{C}$-congruentielle.

DÉFINITION 7 : Une interprétation dirigée est *discrète* si l'ordre $\leqslant$ de son domaine est un ordre discret, i.e. :

$$\forall d, d' \in D : \quad d \leqslant d' \Rightarrow d = d' \qquad \text{ou} \quad d = \omega$$

Notation : Notons $\mathscr{I}_d$ la famille de toutes les interprétations discrètes.

Remarque et exemple : Les interprétations discrètes nous permettent, et ce de manière naturelle, de :

– transformer les fonctions partielles en fonctions totales ; on écrit pour ce faire qu'une fonction non définie en un point prend en ce point la valeur « indéterminée », c'est-à-dire ω.

– formaliser dans le cadre des interprétations dirigées la notion intuitive d'interprétation à valeurs dans un domaine n'ayant aucune structure particulière (*cf.* [26, 27]).

C'est ce que l'exemple suivant met en évidence.

Soit h une fonction partielle de $\mathbf{N}$ dans $\mathbf{N}$. Soit $D = \mathbf{N} \cup \{ \omega \}$, le domaine discret associé à $\mathbf{N}$ (i.e. ordonné par : $x \leqslant x'$ ssi $x = \omega$ ou $x = x'$). C'est un poc et on peut prolonger, de manière canonique, h en une application totale et continue $\bar{h}$ de D dans D, en posant :

$$\forall x \in \mathbf{N} \quad , \quad \bar{h}(x) = \begin{cases} h(x) & \text{si } h(x) \text{ est défini} \\ \omega & \text{si } h(x) \text{ n'est pas défini} \end{cases}$$

$$\bar{h}(\omega) = \inf \{ \bar{h}(x) / x \in \mathbf{N} \}$$

DÉFINITION 8 : Soit C_t la congruence engendrée sur $M(F, V)$ par les ensembles d'axiomes (i) et (ii) suivants :

$$\forall u, v, v' \in V$$

(i) $$g(u, v, v) C_t v \quad \text{et} \quad g(\Omega, v, v') C_t \Omega$$

$$\forall u, v, v', w \in V \cup \{ \Omega \}$$

(ii) $$\begin{cases} g(u, g(u, v, v'), w)\, C_t\, g(u, v, w) \\ g(u, v, g(u, v', w))\, C_t\, g(u, v, w) \end{cases}$$

THÉORÈME 1 : *P_t est $\mathscr{I}_d$-congruentielle c'est-à-dire :*

Pour toute I discrète, $I(g)$ est un test total si et seulement si I est compatible avec C_t.

Preuve : Il est clair que toute interprétation vérifiant P' est compatible avec C_t. Pour vérifier la réciproque il nous faut 3 lemmes : dans ces trois lemmes et dans la suite de la preuve I désignera une interprétation discrète compatible avec C_t. On abrège l'écriture en notant $I(g)(d, d', d'')$ par $(dd'd'')$.

LEMME 1 : *Pour tous $d \neq \omega$, d_0 dans D, on se trouve dans l'un des deux cas*

$$(dd_0\omega) = d_0 \quad \text{et} \quad (d\omega d_0) = \omega$$

ou :

$$(dd_0\omega) = \omega \quad \text{et} \quad (d\omega d_0) = d_0.$$

Preuve : Remarquons que :

$$\Omega \prec g(u, v, \Omega) \prec g(u, v, v)$$

Comme, pour toute I, $I(g)$ est continue donc croissante, on a pour toute valuation I_V associée à I :

$$I_V(\Omega) \leqslant I_V(g(u, v, \Omega)) \leqslant I_V(g(u, v, v))$$

Soit, pour tous d, d_0 dans dom (I) :

$$\omega \leqslant (d\omega d_0) = d_1 \leqslant d_0$$

de même :

$$\omega \leqslant (dd_0\omega) = d_2 \leqslant d_0$$

$\leqslant$ étant un ordre discret, ces inégalités impliquent :

$$d_1 \leqslant d_2 \quad \text{ou} \quad d_2 \leqslant d_1$$

Par exemple $d_1 \leqslant d_2$. Utilisons l'axiome (i) de C_t et la monotonie de $I(g)$, il vient :

$$d_1 = (dd_1d_1) \leqslant (dd_1d_2)$$

Par ailleurs (axiomes (ii) de C_t) :

$$(dd_1d_2) = (d(d\omega d_0)(dd_0\omega)) = (d\omega\omega) = \omega$$

et donc $d_1 = \omega$

De même :

$$d_0 = (d(dd_0\omega)(d\omega d_0)) = (dd_2d_1) \leqslant (dd_2d_2) = d_2$$

Comme nous avons aussi : $d_2 \leqslant d_0$, nous en déduisons :

$$d_0 = d_2.$$

De même : $d_2 \leqslant d_1 \Rightarrow \omega = d_2$ et $d_0 = d_1$.

Q.E.D.

LEMME 2 : $(\exists d_0 \neq \omega/(dd_0\omega) = d_0) \Rightarrow (\forall d_1 (dd_1\omega) = d_1)$

Preuve : Soit $d_1 \in D$.

a) $d_1 = d_0$ ou $d_1 = \omega$: on n'a rien à vérifier

b) $d_1 \neq d_0$ et $d_1 \neq \omega$: d'après le lemme 1, on se trouve dans l'un des 2 cas :

$$(dd_1\omega) = d_1 \quad \text{ou} \quad (d\omega d_1) = d_1$$

Si on avait $(d\omega d_1) = d_1$, on en déduirait :

$$(dd_0d_1) \geqslant (d\omega d_1) = d_1$$

Par ailleurs :

$$(dd_0d_1) \geqslant (dd_0\omega) = d_0.$$

Comme D est discret et que d_0, d_1, ω sont deux à deux distincts, ces deux inégalités ne peuvent se produire simultanément ; par suite :

$$(dd_1\omega) = d_1$$

Q.E.D.

LEMME 2 *bis* :

$$(\exists d_0 \neq \omega / (d\omega d_0) = d_0) \Rightarrow (\forall d_1 (d\omega\, d_1) = d_1$$

LEMME 3 : *Soit d dans* $D - \{\omega\}$:

$$(dd\omega) = d \quad \text{ssi} \quad \forall d_1, d_2 \quad (dd_1 d_2) = d_1$$
$$(dd\omega) = \omega \quad \text{ssi} \quad \forall d_1, d_2 \quad (dd_1 d_2) = d_2$$

Preuve : Les parties « si » de ces deux équivalences sont claires.

Vérifions par exemple la partie « seulement si » de la première équivalence :

$$(dd\omega) = d \qquad \text{implique, par le lemme 2 :}$$
$$\forall d_0 \in D \qquad (dd_0\omega) = d_0$$

Soit en particulier $d_0 = (dd_1 d_2)$, avec d_1 et d_2 quelconques dans D, alors :

$$(d(dd_1 d_2)\omega) = (dd_1 d_2)$$

Mais, par les axiomes (ii) de C_t :

$$(d(dd_1 d_2)) = d_1$$

D'où, pour tous d_1, d_2 dans D :

$$(dd_1 d_2) = d_1$$

La seconde équivalence se vérifie de même.

Q.E.D.

Revenons à la preuve du théorème 1. Posons :

$$V = \{ d \in D - \{\omega\} / (dd\omega) = d \}$$

Par le lemme 1:

$$F = D - (V \cup \{\omega\}) = \{ d \in D - \{\omega\} / (d\omega d) = d \}$$

Nous déduisons alors du lemme 3 que, pour d, d_1, d_2 quelconques dans D :

$$I(g)(d, d_1, d_2) = \begin{cases} d_1 & \text{si} \quad d \in V \\ d_2 & \text{si} \quad d \in F \end{cases}$$

De plus, d'après les axiomes (i) de C_t :

$$I(g)(\omega, d_1, d_2) = \omega$$

$I(g)$ est donc un test total et I vérifie P'.

Q.E.D.

Soit $\mathcal{T}$ la famille des interprétations à valeurs dans un poc dont l'ordre est un ordre total (ie, $\forall d, d' \in D$, $d \leqslant d'$ ou $d' \leqslant d$). Par exemple, toute interpré-

tation de domaine **N**, **Q** ou **R** (avec leur ordre habituel) est dans $\mathscr{T}$. Nous avons de même :

THÉORÈME 2 : *P_t est $\mathscr{T}$-congruentielle et on a : pour toute interprétation I à valeurs dans un domaine totalement ordonné, $I(g)$ est un test total si et seulement si I est compatible avec C_t* (cf. *déf. 8*).

Mais, dès que $I \notin \mathscr{I}_d \cup \mathscr{T}$, la compatibilité avec C_t n'entraîne même plus que $I(g)$ est un test (*cf.* [16]).

Le caractère commun aux familles $\mathscr{I}_d$ et $\mathscr{T}$ qui est nécessaire pour la véracité des théorèmes 1 et 2 est que le produit de deux interprétations (1) de $\mathscr{I}_d$ (resp. $\mathscr{I}$) n'est pas dans $\mathscr{I}_d$ (resp. $\mathscr{T}$) : il est clair que le produit de deux ordres discrets (resp. totaux) n'est pas discret (resp. total).

En effet, on a la :

PROPOSITION 1. *Le produit de deux tests $\bar{g}$ et $\bar{g}'$ n'est un test que dans les cas triviaux suivants :*

- $D = \{ \omega \}$ *ou* $D' = \{ \omega \}$
- $V = D$ *et* $V' = D'$
- $F = D$ *et* $F' = D'$
- $N = D$ *et* $N' = D'$

(où D, V, F, N (resp. D', V', F', N') sont associés à $\bar{g}$ (resp. $\bar{g}'$) conformément aux notations de la définition 6).

Preuve : Supposons que nous ne sommes pas dans l'un des 4 cas ci-dessus. On peut alors trouver d, d_1, d_2 dans D, d', d'_1, d'_2 dans D' tels que :

$$p(d) \neq p'(d') \quad , \quad d_1 \neq d_2 \quad , \quad d'_1 \neq d'_2$$

p (resp. p') est associé à $\bar{g}$ (resp. $\bar{g}'$) par (2)).

Supposons par exemple : $p(d) =$ vrai, $p'(d') =$ faux. Alors :

$$\bar{g} \times \bar{g}' (d \times d', d_1 \times d'_1, d_2 \times d'_2) = d_1 \times d'_2$$

qui est distinct de $d_1 \times d'_1$ et de $d_2 \times d'_2$

Q.E.D.

On en déduit la

PROPOSITION 2. *Soit $\mathscr{C}$ la famille des interprétations dirigées I dont les domaines vérifient certaines conditions U; dès qu'il existe D, D' tous deux non réduits à $\{ \omega \}$ vérifiant U et tels que $D \times D'$ vérifie aussi U, P_t (resp. P_t) n'est pas $\mathscr{C}$-congruentielle.*

(1) C'est-à-dire l'interprétation ayant pour domaine le produit cartésien des deux domaines, muni de l'ordre produit et de la structure de F-algèbre produit (*cf.* [8]).

Exemples : La propriété U_n : toutes les chaînes strictement croissantes de D sont de longueur bornée par n fixé, vérifie l'hypothèse de la proposition 2, alors que U'_n : toutes les chaînes de D sont de longueur n, ne la vérifie pas. Remarquons que U'_2 définit la famille $\mathscr{I}_d$.

Preuve de la proposition 2 :

On peut trouver I et I', ayant pour domaines D et D', et telles que $I(g)$ et $I'(g)$ soient des tests totaux, mais qu'on ne soit dans aucun des trois cas de la proposition 1. En effet on peut obtenir tous les tests de D et D' sous la forme $I(g)$ et $I'(g)$, et D et D' sont tous deux non réduits à $\{ \omega \}$. Alors par la proposition 1, $I \times I'(g) = I(g) \times I'(g)$ n'est pas un test. Si donc P (resp. P_t) était $\mathscr{C}$-congruentielle et était caractérisée dans $\mathscr{C}$ par la compatibilité avec la congruence C, I et I' seraient compatibles avec C, donc aussi $I \times I'$; comme $I \times I'$ appartient à $\mathscr{C}$ (puisque $D \times D'$ vérifie U), $I \times I'(g)$ devrait être un test (resp. un test total), ce qui n'est pas.

Q.E.D.

En prenant $U = \{ \varnothing \}$, on en déduit :

COROLLAIRE : *P et P_t ne sont pas congruentielles.*

Nous déduisons de même de la proposition 1 la

PROPOSITION 3 : *Soit $\mathscr{C}$ une famille d'interprétations stable par produit (resp. ultraproduit) alors P et P_t ne sont pas $\mathscr{C}$-congruentielles (sauf dans les cas triviaux).*

La preuve est tout à fait analogue à celle de la proposition 2 (*cf.* [16], ch. 4 pour préciser les cas triviaux).

4. LES TESTS QUELCONQUES

Nous allons, dans ce paragraphe, montrer que, bien que P ne soit en général pas $\mathscr{C}$-congruentielle, nous pouvons néanmoins obtenir une caractérisation syntaxique de P.

Compte tenu des résultats du paragraphe 2, il est naturel de s'intéresser surtout aux interprétations discrètes : en effet, toutes les familles usuelles d'interprétations contiennent $\mathscr{I}_d$, et si $\mathscr{C}' \subset \mathscr{C}$, toute propriété $\mathscr{C}$-congruentielle est aussi $\mathscr{C}'$-congruentielle. Or nous avons le :

THÉORÈME 3 : *La propriété P : $\{ I(g)$ est un test $\}$ n'est pas $\mathscr{I}_d$-congruentielle.*

Preuve : Nous allons montrer qu'il n'existe pas de congruence C sur $M(F, V)$ telle que :

$$\forall I \in \mathscr{I}_d : \quad I(g) \text{ est un test} \Leftrightarrow I \text{ est compatible avec } C \qquad (3)$$

P s'exprimant en fonction de $I(g)$ seulement, et les interprétations de $\mathscr{I}_d$ étant définies par une propriété de leurs domaines, il est clair qu'on peut faire abstraction des symboles de $F - \{ \Omega, g \}$ et se borner à montrer qu'il n'existe pas de congruence C de $M(\{ g, \Omega \}, V)$ telle que (3).

Dans toute la suite de la preuve nous nous restreignons donc à des congruences sur $M(\{ g, \Omega \}, V)$.

Nous construisons une interprétation discrète I telle que :

1) $I(g)$ n'est pas un test.

2) I est compatible avec toute congruence C qui est vérifiée par toutes les interprétations discrètes I' telles que $I'(g)$ soit un test.

Cela prouve que (2) n'est vérifiée par aucune congruence et démontre donc le théorème.

Le domaine D de I est l'ensemble $\{ \omega, 1, 2 \}$ muni de l'ordre discret de plus petit élément ω. $I(g)$ est définie par :

$$\forall x, y \in D \quad , \quad \forall i, j \in \{ 1, 2 \} \qquad \begin{aligned} I(g)(\omega, x, y) &= \omega \\ I(g)(i, i, x) &= i \\ I(g)(i, j, x) &= \omega \qquad (\text{pour } i \neq j) \end{aligned}$$

1) Il est clair que $I(g)$ n'est pas un test.

2) Soit C une congruence sur $M(\{ g, \Omega \}, V)$.

Comme il nous suffira de ne considérer que des interprétations de domaine D, quel que soit le nombre de symboles de variables figurant dans une règle de C, nous ne disposerons d'une valuation qui ne peut distinguer que les trois valeurs ω, 1, 2. Nous substituons donc aux symboles de variables de V qui figurent dans les règles de C les trois symboles Ω, v_1, v_2 et ce de toutes les manières possibles : nous récrivons ainsi les règles de C, quitte à en augmenter considérablement le nombre, avec uniquement les deux symboles de variables v_1, v_2. Nous ramenons ainsi C à une congruence sur $M(\{ g, \Omega \}, \{ v_1, v_2 \})$, encore notée C : il est en effet clair qu'une interprétation de domaine D est compatible avec la congruence initiale ssi elle est compatible avec la congruence transformée qui, intuitivement, s'obtient en écrivant la compatibilité pour toutes les valuations possibles. De plus, d'après la construction de C, pour toute application φ de $\{ v_1, v_2 \}$ dans $\{ \Omega, v_1, v_2 \}$, l'ensemble des règles obtenues en remplaçant v_i par $\varphi(v_i)$, $i = 1, 2$, dans les règles de C est contenu dans l'ensemble des règles de C. Nous pourrons donc nous borner à considérer la seule valuation :

$$I_V(v_1) = 1 \quad , \quad I_V(v_2) = 2.$$

Nous sous-entendons donc la valuation et notons $I(m)$ au lieu de $I_V(m)$ dans la suite de cette preuve.

Énonçons une définition et 2 lemmes qui nous servirons dans la preuve.

DÉFINITION 9 : Pour $i = 1, 2$, soit G_i la plus petite partie de

$$M(\{ g, \Omega \}, \{ v_1, v_2 \}) \text{ qui :}$$

– contienne v_i

– contienne $g(m_i, m_i', m)(\forall m_i, m_i' \in G_i, \forall m \in M(\{ g, \Omega \}, \{ v_1, v_2 \}))$

G_i est donc défini par la grammaire :

$$G_i \to v_i + g(G_i, G_i, M(\{ g, \Omega \}, \{ v_1, v_2 \}))$$

LEMME 4 : *Soit* m *dans* $M(\{ g, \Omega \}, \{ v_1, v_2 \})$. *Pour* $i = 1, 2$:

$$m \in G_i \Leftrightarrow I(m) = i$$

Preuve : Par récurrence sur la profondeur $p(m)$ de m :

– $p(m) = 1$; alors : $m \in G_i \Leftrightarrow m = v_i \Leftrightarrow I(m) = i$

– supposons le lemme établi si $p(m) \leqslant n$ et soit $m = g(m_1, m_2, m_3)$ de profondeur $n + 1$.

. Si $m \in G_i$, il en est de même de m_1 et m_2, et donc par la récurrence $I(m_1) = I(m_2) = i$.

Par suite : $$I(m) = I(g)(i, i, I(m_3)) = i$$

. Inversement, si $I(m) = i$, par la définition de $I(g)$ nous avons nécessairement : $I(m_1) = I(m_2) = i$; donc, par la récurrence, m_1 et m_2 appartiennent à G_i et il en est de même de m.

Q.E.D.

Pour $i = 1, 2$, soit I_i une interprétation de domaine D définie par :

$$\forall x, y \in D \quad , \quad \forall i, j \in \{ 1, 2 \}$$
$$I_i(g)(\omega, x, y) = \omega$$
$$I_i(g)(i, x, y) = x$$
$$I_i(g)(j, x, y) = \omega \qquad \text{pour } j \neq i$$

Il est clair que $I_i(g)$ est un test.

LEMME 5 : *Soit* m *dans* $M(\{ g, \Omega \}, \{ v_1, v_2 \})$, *pour* $i = 1, 2$:

$$m \in G_i \Leftrightarrow I_i(m) = i$$

Preuve : comme pour le lemme 4.

Conséquence :

$$m \notin G_i \Leftrightarrow I_i(m) = j \qquad \text{ou} \quad I_i(m) = \omega$$

Revenons à la preuve de la seconde condition vérifiée par I.

Soit $m C m'$ une congruence quelconque qui est vérifiée par toute I' discrète telle que $I'(g)$ soit un test ; comme $I_1(g)$ et $I_2(g)$ sont des tests, I_1 et I_2 vérifient $m C m'$ et nous avons

$$I_1(m) = I_1(m') \qquad \text{et} \qquad I_2(m) = I_2(m')$$

Nous en déduisons, en examinant les divers cas possibles, que I satisfait aussi $I(m) = I(m')$ et donc que I est compatible avec C.

Soit en effet i dans $\{ 1, 2 \}$; si $m \in G_i$, le lemme 5 implique : $I_i(m) = i$. Mais, comme $I_i(m) = I_i(m')$, nous avons aussi $I_i(m') = i$ et donc : $m' \in G_i$. Par symétrie, $m' \in G_i \Rightarrow m \in G_i$.

Donc, pour $i = 1, 2$:

$$m \in G_i \Leftrightarrow m' \in G_i$$

Nous n'avons alors que trois possibilités :

— pour $i = 1$ (resp. 2), m et m' sont dans G_i ; alors par le lemme 4 :

$$I(m) = I(m') = i$$

— m n'est pas dans $G_1 \cup G_2$ et m' non plus.

Par le lemme 4 : $I(m) \neq i$ et $I(m) \neq j$, et de même pour m' ; alors :

$$I(m) = I(m') = \omega$$

Q.E.D.

Nous pouvons, de manière analogue au Théorème 3, montrer le :

THÉORÈME 3 bis : La propriété P : $\{ I(g)$ est un test $\}$ n'est pas $\mathcal{T}$-congruentielle.

Il suffit de remplacer G_1 par G_0 tel que :

$$m \in G_0 \Leftrightarrow I(m) = \omega$$

Nous laissons au lecteur le soin de rétablir les détails de la preuve.

Nous obtenons toutefois une caractérisation syntaxique de P, qui s'exprime par une compatibilité forte (en un certain sens du second ordre) avec un système d'axiome S, en supposant que F contient au moins un symbole f d'arité $\geqslant 1$ et distinct de g. Pour simplifier l'écriture nous supposons f unaire.

DÉFINITION 10 : Soit S le système d'axiomes suivant :

$$\forall u, v, v', w \in V \cup \{ \Omega \}$$

1. $g(u, v, v) \prec v$

2. $\begin{cases} g(u, f(v), w) \, C \, g(u, f(g(u, v, v')), w) \\ g(u, v, f(w)) \, C \, g(u, v, f(g(u, v', w))) \end{cases}$

Soit C la congruence engendrée par les axiomes 2.

DÉFINITION 11 : Une interprétation dirigée I de $\{ g, \Omega \}$ est *fortement compatible* avec S, ssi pour tous d, d', d_1, d_2 dans $D = \mathrm{dom}(I)$:

1) $I(g)(d, d', d') \leqslant d'$ (i.e. I est compatible avec l'axiome 1).

2) Pour toute application continue φ de D dans D :

$$I(g)(d, \varphi(d_1), d_2) = I(g)(d, \varphi(I(g)(d, d_1, d')), d_2)$$
$$I(g)(d, d_1, \varphi(d_2)) = I(g)(d, d_1, \varphi(I(g)(d, d', d_2)))$$

Notons encore $I(g)(d, d', d'')$ par $(dd'd'')$, et énonçons quelques lemmes qui serviront à prouver le théorème 4.

LEMME 6 : *Soit I une interprétation dirigée fortement compatible avec les axiomes 2 de S, alors I est compatible avec la congruence C' engendrée sur* $M(\{g, \Omega\}, V)$ *par, pour* $u, v, v', w \in V \cup \{\Omega\}$:

$$\text{(ii)} \qquad \begin{cases} g(u, g(u, v, v'), w)\, C'\, g(u, v, w) \\ g(u, v, g(u, v', w))\, C'\, g(u, v, w) \end{cases}$$

Preuve : Il suffit d'écrire que I est fortement compatible avec les axiomes 2 de S en employant pour φ l'application identité de dom(I) dans lui-même.

LEMME 7 : *Soit I une interprétation dirigée fortement compatible avec S, pour tous d, d', d'' dans dom* (I) :

$$(dd'\omega) = \omega \Rightarrow (dd''\omega) = \omega$$
$$(d\omega d') = \omega \Rightarrow (d\omega d'') = \omega$$

Preuve : Soit φ une application continue de D dans D telle que :

$$\varphi(\omega) = \omega \qquad \text{et} \qquad \varphi(d') = d''$$

Vérifions par exemple la première implication :

$$(d\varphi((dd'\omega))\omega) = (d\omega\omega)$$
$$= \omega \qquad (\text{car } I \text{ est compatible avec l'axiome 1 de } S)$$

D'autre part, I étant fortement compatible avec les axiomes 2 de S :

$$(d\varphi((dd'\omega))\omega) = (d\varphi(d')\omega) = (dd''\omega)$$

d'où :

$$(dd''\omega) = \omega$$

Q.E.D.

LEMME 8 : *Soit I une interprétation dirigée fortement compatible avec S, pour tous d, d', d'' dans dom* (I) :

$$(ddd) = \omega \Rightarrow (dd'd'') = \omega$$

Preuve : Soit d quelconque dans dom(I). Supposons que :
$(ddd) = \omega$. $I(g)$ étant monotone, nous en déduisons :

$$(dd\omega) = (d\omega d) = \omega$$

d'où par le lemme 7, pour tous d', d'' dans dom (I) :

$$(dd'\omega) = (d\omega d'') = \omega$$

en appliquant le lemme 6 nous en déduisons :

$$(dd'd'') = (d(dd'\omega)(d\omega d'')) = (d\omega\omega) = \omega$$

Q.E.D.

LEMME 9 : *Toute interprétation discrète compatible avec l'axiome 1 de S vérifie :*

$$\forall d, d' \in \text{dom}(I) : \quad (dd'd') = \omega \qquad \text{ou} \qquad (dd'd') = d'$$

LEMME 10 : *Soit I une interprétation discrète fortement compatible avec S, alors : pour tous d, d' ≠ ω, d'' dans dom (I) :*

$$(dd'\omega) = d' \Rightarrow (d\omega d'') = \omega$$
$$(d\omega d') = d' \Rightarrow (dd''\omega) = \omega$$

Preuve : Vérifions par exemple la première implication. Par le lemme 9 et la monotonie de $I(g)$, $(d\omega d')$ ne peut valoir que ω ou d', pour tous d, d' dans dom (I).

Soit d' dans dom $(I) - \{ \omega \}$; supposons :

$$(dd'\omega) = d' \qquad \text{et} \qquad (d\omega d') = d'$$

$I(g)$ étant monotone :

$$(dd'd') = d'$$

$I(g)$ étant compatible avec l'axiome 1 de S :

$$(d\omega\omega) = \omega$$

Mais alors, par le lemme 6 :

$$d' = (dd'd') = (d(d\omega d')(dd'\omega)) = (d\omega\omega) = \omega$$

d'où une contradiction. Donc : $(d\omega d') = \omega$.

Alors, par le lemme 7 pour tout d'' dans dom (I)

$$(d\omega d') = \omega$$

Q.E.D.

Nous pouvons maintenant démontrer le :

THÉORÈME 4 : *Soient g d'arité 3 dans F, f d'arité 1 dans F.*

Soit I une interprétation discrète de F. $I(g)$ est un test si et seulement si I est fortement compatible avec S (cf. déf. 10 et 11).

Preuve : La partie « seulement si » est claire.

Vérifions la partie « si ». Soit donc I dans $\mathscr{I}_d$ fortement compatible avec S; soit D le domaine de I.

Posons :
$$V = \{ d \in D/(dd\omega) = d \}$$
$$F = \{ d \in D/(d\omega d) = d \}$$

Par le lemme 10, V et F sont disjointes. Si donc :
$$N = D - (V \cup F)$$
$\{ V, F, N \}$ est une partition de D. D'autre part :
$$N = \{ d \in D/(ddd) = \omega \} \tag{4}$$

En effet, si d n'appartient pas à $V \cup F$, par le lemme 9 :
$$(dd\omega) = (d\omega d) = \omega$$
donc, en appliquant le lemme 6, puis la compatibilité de I avec l'axiome 1 de S :
$$(ddd) = (d(dd\omega)(d\omega d)) = (d\omega\omega) = \omega$$
Inversement, si $(ddd) = \omega$, comme $I(g)$ est monotone :
$$(dd\omega) = (d\omega d) = \omega$$
donc d est dans N.

Soit d dans V; le lemme 7 implique que, pour tout d' dans D : $(dd'\omega) = d'$. Par suite, $I(g)$ étant monotone :
$$(dd'd'') \geqslant (dd'\omega) = d'.$$
Donc :
$$d \in V \Rightarrow \forall d', d'' \in D : \quad (dd'd'') = d'$$
de même :
$$d \in F \Rightarrow \forall d', d'' \in D : \quad (dd'd'') = d''$$

Enfin, d'après la caractérisation (4) de N et le lemme 8
$$d \in N \Rightarrow \forall d', d'' \in D : \quad (dd'd'') = \omega$$

$I(g)$ est donc de la forme :
$$\forall d, d', d'' \in D \quad I(g)(d, d', d'') = \begin{cases} d' & \text{si } d \in V \\ d'' & \text{si } d \in F \\ \omega & \text{si } d \in N \end{cases}$$
C'est donc bien un test. Q.E.D.

On vérifie facilement qu'on ne peut affaiblir les conditions du théorème 4. L'axiome 1 est nécessaire (*cf.* [16]); d'autre part, la preuve du théorème 3 montre qu'on ne peut remplacer la compatibilité forte avec les axiomes de S,

très contraignante, par une compatibilité simple avec une congruence : en effet, l'interprétation I construite dans le théorème 3 est compatible avec l'axiome 1 de S, comme on le vérifie facilement; or, $I(g)$ n'est pas un test.

On obtient de manière similaire le :

THÉORÈME 4 *bis* : *Soient g d'arité 3 dans F, f d'arité 1 dans F et I dans* $\mathcal{T}$. $I(g)$ *est un test si et seulement si I est fortement compatible avec S.*

Il suffit de remplacer dans la preuve du Théorème 4 le lemme 9 par le :

LEMME 9 *bis* : *Soit I dans* $\mathcal{T}$; *si I est fortement compatible avec S, pour tous d, d' dans dom(I), (dd'*ω*) et (d*ω*d') ne peuvent prendre que les valeurs* ω *ou d'.*

On constate de même qu'on ne peut affaiblir les hypothèses du Théorème 4 bis.

REMERCIEMENTS

Je remercie vivement J. F. Perrot qui a accepté de relire ce texte et dont les judicieuses suggestions m'ont permis d'améliorer notablement la présentation.

BIBLIOGRAPHIE

1. A. ARNOLD et M. DAUCHET, *Bitransductions de forêts*, Proceedings of the 3rd Int. Col. « Automata, Languages, Programming », Edinburgh Univ. Press, Edinburgh, 1976, p. 74-86.
2. J. ARSAC, *Nouvelles leçons de programmation*, Publications de l'Institut de Programmation, n° 75-29, Université Paris 6, Paris, 1975.
3. E. ASHCROFT et Z. MANNA, *The translation of GO TO programs to WHILE programs*, Proceedings of the IFIP conference 1971, North-Holland, Amsterdam, 1971, p. 250-255.
4. J. W. DE BAKKER et D. SCOTT, *A theory of programs*, IBM seminar, Vienne, 1969, manuscrit non publié.
5. G. BERRY et J. J. LÉVY, *Minimal and optimal computations of recursive programs*, à paraître.
6. G. BOUDOL, *Langages d'arbres algébriques*, à paraître dans la RAIRO, Paris, 1977.
7. A. K. CHANDRA, *On the properties and applications of program schemes*, Ph. D. Thesis, Stanford, 1973.
8. P. M. COHN, *Universal algebra, Harper's series in modern mathematics*, Londres, 1965.
9. B. COURCELLE et I. GUESSARIAN, *On algebraic families of interpretations*, à paraître.
10. G. COUSINEAU et J. M. RIFFLET, *Langages d'interprétation des schémas récursifs*, RAIRO, série rouge, vol. 9, Paris, 1975, p. 21-42.
11. O. J. DAHL, E. DIJKSTRA et C. A. R. HOARE, *Structured programming*, APIC studies on data processing, n° 8, Academic Press, Londres, 1972.

12. R. W. FLOYD, *Assigning meanings to programs, Proceedings of a symposium in applied mathematics*, Vol. 19, *Mathematical aspects of computer science*, Rhodes Island, Amer. Math. Soc., 1967, p. 19-32.
13. R. W. FLOYD et D. E. KNUTH, *Notes on avoiding GO TO statements, Information processing letters*, **1**, 1971, p. 23-31.
14. E. P. FRIEDMANN, *Deterministic languages and monadic recursion schemes*, Ph. D. Thesis, Harvard University, 1974.
15. S. A. GREIBACH, *Theory of program structures : schemes, semantics and verification, Lecture notes in computer science*, Vol. 36, Springer-Verlag, 1975.
16. I. GUESSARIAN, *Schémas de programme récursifs polyadiques : équivalences et classes d'interprétations*, Thèse d'État, Université Paris 7, Paris, 1975.
17. I. GUESSARIAN. *Équivalences dans l'algèbre des schémas de programme, Proceedings of the 1st Programming symposium, Lecture notes in computer science*, Vol. 19, Springer-Verlag, 1974, p. 204-220.
18. I. GUESSARIAN, *Semantic equivalence of program schemes and its syntactic characterization, Proceedings of the 3rd Int. Col.* «Automata, Languages, Programming», Edinburgh Univ. Press, Edinburgh, 1976, p. 189-200.
19. I. IANOV, *The logical schemes of algorithms, Problems of cybernetics*, Pergamon Press, 1960, p. 82-140.
20. L. KOTT, *Sémantique algébrique d'un langage de programmation type ALGOL*, à paraître dans la RAIRO, Paris, 1977.
21. D. LUCKHAM, D. PARK et M. PATERSON, *On formalized computer programs*, J. Comp. Sys. Sc., Vol. 4, 1970, 220-249.
22. J. MAC CARTHY, *A basis for a mathematical theory of computation, Computer programming and formal systems*, P. Braffort, D. Hirschberg Eds., North-Holland, Amsterdam, 1963, p. 33-70.
23. Z. MANNA, *Mathematical theory of computation*, Mac Graw Hill, Londres, 1975.
24. Z. MANNA, S. NESS et J. VUILLEMIN, *Inductive methods for proving properties of programs*, C. Ass. Comp. Mac. 16, 1973, p. 491-502.
25. R. MILNER, *Equivalences on program schemes*, J. Comp. Sys. Sc., Vol. 4, 1970, p. 205-219.
26. M. NIVAT, *On the interpretation of recursive polyadic program schemes, Atti del convegno di informatica teorica*, Symposia Mathematica, Vol. 15, Rome, 1975, p. 256-281.
27. M. NIVAT, *Formalisation de la sémantique des langages de programmation, Bulletin de liaison de la recherche en informatique et automatique*, n° 23, IRIA, Paris, 1976, p. 2-13.
28. D. PARK, *Fixpoint induction and proofs of program properties*, Machine Intelligence, Vol. 5, Edinburgh University Press, Edinburgh, 1970, p. 59-78.
29. M. PATERSON, *Equivalence problems in a model of computation*, Ph. D. Thesis, Artificial intelligence laboratory, M.I.T., 1967.
30. J. D. RUTLEDGE, *On Ianov's program schemata*, J. Ass. Comp. Mac., Vol. 11, 1964, p. 1-9.
31. D. SCOTT, *The lattice of flow diagrams, Symposium on semantics of algorithmic languages, Lecture notes in mathematics*, Vol. 188, Springer-Verlag, 1971, p. 311-372.
32. H. R. STRONG, *Translating recursion equations into flowcharts*, J. Comp. Sys. Sc., Vol. 5, 1971, p. 254-285.
33. N. WIRTH, *Systematic programming*, Prentice Hall, 1973.