RAIRO
INFORMATIQUE THÉORIQUE

P. SALLÉ

Note sur la sémantique des structures de contrôle

RAIRO – Informatique théorique, tome 13, n° 2 (1979), p. 185-188.

<http://www.numdam.org/item?id=ITA_1979__13_2_185_0>

R.A.I.R.O. Informatique théorique/Theoretical Informatics
(vol. 13, n° 2, 1979, p. 185 à 188)

NOTE SUR LA SÉMANTIQUE DES STRUCTURES DE CONTRÔLE (*)

par P. SALLE (1)

Communiqué par J.-F. PERROT

Résumé. — *Cette courte note est consacrée à la correction d'une erreur parue dans les articles de F. Nozick et B. Robinet sur la sémantique des structures de contrôle et des transformations de programmes.*

Abstract. — *This short note is devoted to correcting an error in the Nozick-Robinet papers about the semantics of control structures and program transformations.*

Nous rectifions une erreur de détail dans les articles de F. Nozick et B. Robinet « Une sémantique de structures de contrôle » [4] et de B. Robinet « un modèle fonctionnel des structures de contrôle » [5] et nous montrons que les résultats obtenus en matière de transformations de programmes ne sont pas remis en cause.

Rappelons que dans le premier article les auteurs définissent une sémantique σ_E du surlangage EXEL et prouvent sa correction vis-à-vis de la sémantique « naïve » de ce langage telle qu'elle résulte de sa définition dans [1]. σ_E est définie inductivement dans un environnement de calcul ou contexte muni d'une structure de pile, que nous désignerons ci-après par x, en utilisant les éléments suivants :

— les opérateurs de Böhm et Jacopini [2], à savoir

$$\pi(a, b)(x) = b(a(x))$$

$$\Delta(\alpha, a, b)(x) = \begin{cases} a(x) & \text{si } \alpha(x) \text{ est vrai} \\ b(x) & \text{si } \alpha(x) \text{ est faux} \end{cases}$$

$$\varphi(\alpha, a)(x) = a^n(x)$$

avec n le plus petit entier tel que $\alpha(a^n(x))$ soit faux

$$\Lambda(x) = x;$$

(*) Reçu octobre 1978.
(1) Université Paul-Sabatier, Laboratoire Langages et Systèmes informatique.

R.A.I.R.O. Informatique théorique/Theoretical Informatics, 0399-0540/1979/185/$ 4.00

— deux opérations d'empilement

$$\left.\begin{array}{l} x \xrightarrow{T} \langle t, x \rangle \\ x \xrightarrow{F} \langle f, x \rangle \end{array}\right\} \quad t, f \text{ valeurs booléennes;}$$

— une opération de dépilement

$$\omega(\langle v, x \rangle) = t \Leftrightarrow v = t$$
$$= f \Leftrightarrow v = f$$

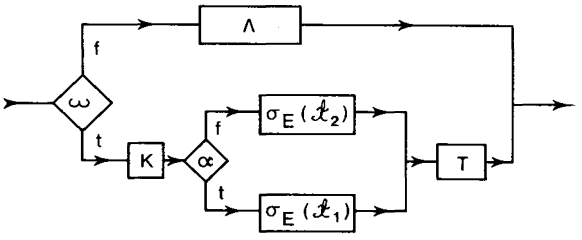
— un prédicat ω dont la valeur est celle du sommet de la pile

$$\omega(\langle v, x \rangle) = t \Leftrightarrow v = t = f \Leftrightarrow v = f.$$

La sémantique d'un programme EXEL $\mathscr{P}$ évalué dans le contexte x est alors $\sigma_E(\mathscr{P})x$ où $\sigma_E(\mathscr{P})$ est obtenu en composant les sémantiques des instructions élémentaires du langage qui forment le programme et où la pile de contexte initial x est $\langle t, x' \rangle$ où x' désigne l'état initial de la mémoire. Enfin la sémantique de chaque instruction élémentaire comporte un test sur le sommet de la pile x, qui détermine suivant la valeur du prédicat ω si cette instruction doit être exécutée ou non. Les instructions élémentaires en question correspondent aux structures de contrôle d'EXEL : itération, sortie indexée, alternative et composition. C'est dans la sémantique de l'alternative proposée en [4] que gît le lièvre. Nos auteurs donnent :

$$\sigma_E(\underline{\text{SI}}\ \alpha\ \underline{\text{ALORS}}\ \mathscr{A}_1\ \underline{\text{SINON}}\ \mathscr{A}_2\ \underline{\text{IS}})$$
$$= \Delta(\omega, \pi(\pi(K, \Delta(\alpha, \sigma_E(\mathscr{A}_1), \sigma_E(\mathscr{A}_2))), T), \Lambda) \quad (1)$$

qui correspond au diagramme



On en déduit que :

(*a*) $\sigma_E(\underline{\text{SI}}\ \alpha\ \underline{\text{ALORS}}\ \mathscr{A}_1\ \underline{\text{SINON}}\ \mathscr{A}_2\ \underline{\text{IS}})\langle t, x \rangle$

$= \sigma_E(\mathscr{A}_1)x$ si $\alpha(x)$ est vrai

$= \sigma_E(\mathscr{A}_2)x$ si $\alpha(x)$ est faux

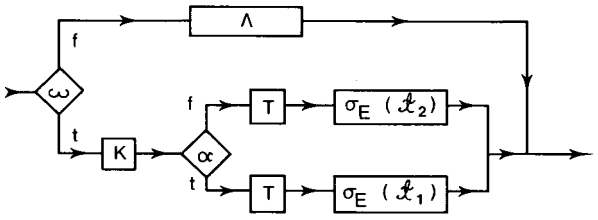
ce qui est presque, mais pas tout à fait le résultat souhaité. En effet, de par la définition récursive de σ_E, une instruction n'est exécutée que si le sommet de la pile de contexte x est la valeur t; en particulier afin que le prédicat α puisse porter sur l'état de la mémoire, il faut avant son évaluation dépiler la valeur t au sommet de x, puis après cette évaluation la restaurer afin d'exécuter $\sigma_E(\mathscr{A}_1)$ ou $\sigma_E(\mathscr{A}_2)$ selon la valeur de α trouvée. En d'autre termes la sémantique de l'alternative doit satisfaire

$$(b)\quad \sigma_E(\underline{\text{SI}}\ \alpha\ \underline{\text{ALORS}}\ \mathscr{A}_1\ \underline{\text{SINON}}\ \mathscr{A}_2\ \underline{\text{IS}})\langle t, x\rangle$$
$$= \sigma_E(\mathscr{A}_1)\langle t, x\rangle \quad \text{si}\ \ \alpha(x)\ \text{est vrai}$$
$$= \sigma_E(\mathscr{A}_2)\langle t, x\rangle \quad \text{si}\ \ \alpha(x)\ \text{est faux}$$

Nous proposons donc la définition suivante :

$$\sigma_E(\underline{\text{SI}}\ \alpha\ \underline{\text{ALORS}}\ \mathscr{A}_1\ \underline{\text{SINON}}\ \mathscr{A}_2\ \underline{\text{IS}})$$
$$= \Delta(\omega, \pi(K, \Delta(\alpha, \pi(T, \sigma_E(\mathscr{A}_1)), \pi(T, \sigma_E(\mathscr{A}_2)))), \Lambda) \quad (2)$$

qui correspond au diagramme :



D'ailleurs dans la preuve du théorème énonçant la correction de σ_E les auteurs utilisent implicitement la définition (*b*). Il n'y a donc pas lieu de la modifier et le théorème reste vrai. Dans les exemples de traduction il conviendra de remplacer systématiquement (1) par (2).

Dans [5] on définit une interprétation du langage EXEL dans le langage CUCH [3] Σ_E par composition de σ_E avec une interprétation des diagrammes de Bohm et Jacopini dans le langage CUCH. On y retrouve la définition de σ_E qu'il convient de corriger en remplaçant (1) par (2). Il s'en suit qu'il faut également remplacer la traduction de σ_B ((1)) par σ_B ((2)) c'est-à-dire remplacer :

$$\Sigma_E(\underline{\text{SI}}\ \alpha\ \underline{\text{ALORS}}\ \mathscr{A}_1\ \underline{\text{SINON}}\ \mathscr{A}_2\ \underline{\text{IS}})$$
$$= \text{Si top } (\circ(\circ \text{pop (if } \alpha\ \Sigma_E(\mathscr{A}_1)\ \Sigma_E(\mathscr{A}_2))) \text{ (empile 0)}) \quad (3)$$

par

$$\neg\,(\circ \text{pop}\ (\text{if}\ \alpha\ (\circ(\text{empile}\,\underline{0})\ \Sigma_E(\mathscr{A}_1))\ (\circ(\text{empile}\,\underline{0})\ \Sigma_E(\mathscr{A}_2)))) \qquad (4)$$

Ces différentes modifications sont sans effet sur les preuves de transformations de programmes.

BIBLIOGRAPHIE

1. J. ARSAC, L. NOLIN, G. RUGGIU et J.P. VASSEUR, *Le système de programmation structurée Exel*, Rev. Techn. Thomson-C.S.F., vol. 6, n° 3, 1974, p. 715-736.
2. C. BOHM et G. JACOPINI, *Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules*, Comm. A.C.M., vol. 9, n° 5, 1966, p. 365-371.
3. C. BOHM, *The CUCH as a Formal and Description Language*, Formal Language, Description Languages for Computer Programming, T. B. STEEL, Ed., North Holland Pub., 1966, p. 179-197.
4. B. ROBINET et F. NOZICK, *Sémantique des structures de contrôle*, R.A.I.R.O. Informatique théorique, vol. 11, n° 1, 1977, p. 63-74.
5. B. ROBINET, *Un modèle fonctionnel des structures de contrôle*, R.A.I.R.O. Informatique théorique, vol. 11, n° 3, 1977, p. 213-236.