

## 1 Exercice I : Sémantique opérationnelle : un débogueur

### 1.5 Sémantique des commandes

**Question 1** Donner la sémantique opérationnelle de la commande `setbrk(n)`

$$((cp, C, \sigma), \beta, \text{setbrk}(n)) \rightarrow ((cp, C, \sigma), \beta \cup \{n\}).$$

**Question 2** Donner la sémantique opérationnelle de la commande `rmbrok(n)`

$$((cp, C, \sigma), \beta, \text{rmbrok}(n)) \rightarrow ((cp, C, \sigma), \beta \setminus \{n\}).$$

**Question 3** Donner la sémantique opérationnelle de la commande `set(x,v)`

$$((cp, C, \sigma), \beta, \text{set}(x, v)) \rightarrow ((cp, C, \sigma[x \mapsto v]), \beta).$$

**Question 4** Donner la sémantique opérationnelle de la commande `print(x)`  $((cp, C, \sigma), \beta, \text{print}(x)) \rightarrow$

$$((cp, C, \sigma), \beta).$$

**Question 5** Donner la sémantique opérationnelle de la commande `step` 2 solutions

–

$$\frac{(pc, C, \sigma) \triangleright (pc', C, \sigma')}{((cp, C, \sigma), \beta, \text{step}) \rightarrow ((cp', C, \sigma'), \beta)}.$$

–

$$((cp, C, \sigma), \beta, \text{step}) \rightarrow \begin{cases} ((pc + 1, C, \sigma[x \mapsto \mathcal{A}[a]_\sigma]), \beta) & \text{si } C(pc) = x := a \\ ((l, C, \sigma), \beta) & \text{si } C(pc) = \text{goto } l \\ ((pc + 1, C, \sigma), \beta) & \text{si } C(pc) = \text{if } b \text{ goto } l \text{ et } \mathcal{B}[b]_\sigma = \text{ff} \\ ((l, C, \sigma), \beta) & \text{si } C(pc) = \text{if } b \text{ goto } l \text{ et } \mathcal{B}[b]_\sigma = \text{tt} \end{cases}$$

**Question 6** Donner la sémantique opérationnelle de la commande `cont`. Indication : s'inspirer des règles vues en TD pour la commande `repeat`.

$$\frac{(pc, C, \sigma) \triangleright (pc', C, \sigma')}{((pc, C, \sigma), \beta, \text{cont}) \rightarrow ((pc', C, \sigma'), \beta)} pc' \in \beta$$

$$\frac{(pc, C, \sigma) \triangleright (pc', C, \sigma') \quad ((pc', C, \sigma'), \beta, \text{cont}) \rightarrow ((pc'', C, \sigma''), \beta)}{((pc, C, \sigma), \beta, \text{cont}) \rightarrow ((pc'', C, \sigma''), \beta)} pc' \notin \beta$$

**Question 6 : une alternative** On peut aussi écrire :

$$\frac{(pc, C, \sigma) \triangleright (pc', C, \sigma')}{((pc, C, \sigma), \beta, \text{cont}) \rightarrow ((pc', C, \sigma'), \beta)} pc' \in \beta$$

$$\frac{(pc, C, \sigma) \triangleright (pc', C, \sigma')}{((pc, C, \sigma), \beta, \text{cont}) \rightarrow ((pc', C, \sigma'), \beta, \text{cont})} pc' \notin \beta$$

**Question 7** Donner la sémantique opérationnelle de la commande `run`. **ATTENTION** : on ne change que le compteur programme, qui devient `1`. Dans l'énoncé, on supposait que l'on "partait" de la configuration initiale. Bien entendu, lorsque l'on fait "run", on ne change pas les points d'arrêt.

$$\frac{((1, C, \sigma), \beta, \text{cont}) \rightarrow ((pc', C, \sigma'), \beta)}{((pc, C, \sigma), \beta, \text{run}) \rightarrow ((pc', C, \sigma'), \beta)}$$

**Question 7 : une alternative**

$$\begin{aligned} ((pc, C, \sigma), \beta, \text{run}) &\rightarrow ((1, C, \sigma), \beta, \text{cont}) & 1 \notin \beta \\ ((pc, C, \sigma), \beta, \text{run}) &\rightarrow ((1, C, \sigma), \beta) & 1 \in \beta \end{aligned}$$

## 1.1 Séquence de "débogage"

**Question 8**

$$\begin{aligned} &((1, C, \sigma), \emptyset, \text{setbrk}(3)) \\ &((1, C, \sigma), \{3\}, \text{cont}) \\ &((3, C, [x \mapsto 8, y \mapsto 12]), \{3\}, \text{cont}) \\ &((3, C, [x \mapsto 4, y \mapsto 12]), \{3\}, \text{cont}) \\ &((3, C, [x \mapsto 8, y \mapsto 12]), \{3\}) \end{aligned}$$

Prolongation : il suffit de rajouter `set(x,12)`.

**Question 9**

$$\begin{aligned} &((1, C, [x \mapsto ?, y \mapsto ?]), \emptyset, \text{setbrk}(3)) \\ &((1, C, [x \mapsto ?, y \mapsto ?]), \{3\}, \text{step}) \\ &((2, C, [x \mapsto 8, y \mapsto ?]), \{3\}, \text{step}) \\ &((3, C, [x \mapsto 4, y \mapsto 12]), \{3\}) \end{aligned}$$

Prolongation : il suffit de rajouter `set(x,12)`.

## 2 Exercice II : Sémantique statique

### 1.5 Sémantique des commandes

**Question 1**

**Expressions**

$$\gamma \vdash n : \text{Int} \quad \gamma \vdash k : \text{Real} \quad \gamma \vdash x : \Gamma(x) \quad \gamma \vdash \text{true} : \text{Bool} \quad \frac{\Gamma \vdash b : \text{Bool}}{\Gamma \vdash \text{not } b : \text{Bool}}$$

$$\frac{\Gamma \vdash E_1 : t_1 \quad \Gamma \vdash E_2 : t_2}{\Gamma \vdash E_1 + E_2 : t_1 \sqcup t_2} \quad t_1, t_2 \in \{\text{Int}, \text{Real}\} \quad \text{ou on rajoute dans Exp} \quad \frac{\Gamma \vdash E : \text{Int}}{\Gamma \vdash E : \text{Real}}$$

$$\frac{\Gamma \vdash E_1 : t_1 \quad \Gamma \vdash E_2 : t_2}{\Gamma \vdash E_1 = E_2 : \text{Bool}} (t_1, t_2 \in \{\text{Int}, \text{Real}\} \vee t_1 = t_2 = \text{Bool}) \quad \text{ou} \quad \frac{\Gamma \vdash E_1 : t_1 \quad \Gamma \vdash E_2 : t_2}{\Gamma \vdash E_1 = E_2 : \text{Bool}} t_1 \sqcup t_2 \neq \text{Bad}$$

## Instructions

$$\frac{\Gamma \vdash E : t}{\Gamma \vdash x := E \mid \Gamma \cup \{x \mapsto t\}} (\Gamma(x) = \text{Void} \vee \Gamma(x) = t \vee \neg(\Gamma(x) = \text{Real} \wedge t = \text{Int}))$$

$$\frac{\Gamma \vdash E : \text{Int}}{\Gamma \vdash x := E \mid \Gamma} (\Gamma(x) = \text{Real})$$

$$\frac{\Gamma \vdash S_1 : \Gamma_1 \quad \Gamma_1 \vdash S_2 : \Gamma_2}{\Gamma \vdash S_1 ; S_2 : \Gamma_2}$$

$$\frac{\Gamma \vdash E : \text{Bool} \quad \Gamma \vdash S_1 : \Gamma_1 \quad \Gamma \vdash S_2 : \Gamma_2}{\Gamma \vdash \text{if } E \text{ then } S_1 \text{ else } S_2 : \Gamma_1 \cup \Gamma_2} \quad \frac{\Gamma \vdash E : \text{Bool} \quad \Gamma \vdash S : \Gamma_1}{\Gamma \vdash \text{while } E \text{ do } S : \Gamma \cup \Gamma_1}$$

## Question 2

## Question 3

$$\frac{\Gamma \vdash E : t}{\Gamma \vdash x := E \mid \Gamma \cup \{x \mapsto t\}}$$

## Question 4 2 solutions :

**Solution 1** On change les configurations : On rajoute un environnement de procédures  $\Gamma_p$  contenant les noms des procédures définies.

$$\frac{(\Gamma, \Gamma_p) \vdash S : \Gamma' \quad (\Gamma', \Gamma_p \cup \{p\}) \vdash D_p : (\Gamma'', \Gamma_p'')}{(\Gamma, \Gamma_p) \vdash \text{proc } p \text{ is } S \text{ endproc}; D_p : (\Gamma'', \Gamma_p'')} (p \notin D_P(D_p)) \quad (\Gamma, \Gamma_p) \vdash \epsilon : (\Gamma, \Gamma_p)$$

$$\overline{(\Gamma, \Gamma_p) \vdash \text{call } p : (\Gamma, \Gamma_p)}^{p \in \Gamma_p}$$

## Solution 2

$$\frac{\Gamma_p[p \mapsto S] \vdash D_p : \Gamma'_p}{\Gamma_p \vdash \text{proc } p \text{ is } S \text{ endproc}; D_p : \Gamma'_p} (p \notin D_P(D_p))$$

$$\frac{(\Gamma, \Gamma_p) \vdash S : \Gamma'}{(\Gamma, \Gamma_p) \vdash \text{call } p : \Gamma'} \Gamma_p(p) = S$$

## 3 Exercice III : Optimisation

**Expressions redondantes** Soit  $\mathcal{E} = \{i < j, i - j, i + j\}$ .

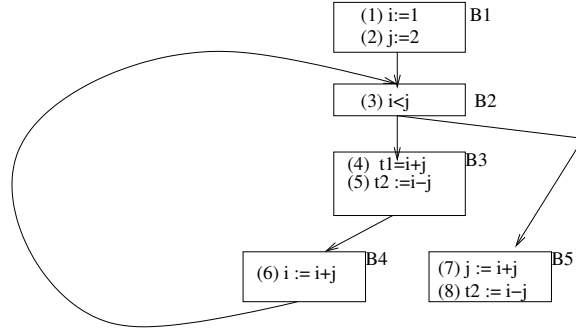


FIG. 1 – Graphe de flot de contrôle initial

	$B1$	$B2$	$B3$	$B4$	$B5$
Gen	$\emptyset$	$i < j$	$i + j, i - j$	$\emptyset$	$i - j$
Kill	$\mathcal{E}$	$\emptyset$	$\emptyset$	$\mathcal{E}$	$\mathcal{E}$
In	$\emptyset$	$\mathcal{E}$	$\mathcal{E}$	$\mathcal{E}$	$\mathcal{E}$
Out	$\emptyset$	$\mathcal{E}$	$\mathcal{E}$	$\emptyset$	$i - j$
In	$\emptyset$	$\emptyset$	$\mathcal{E}$	$\mathcal{E}$	$\mathcal{E}$
Out	$\emptyset$	$i < j$	$\mathcal{E}$	$\emptyset$	$i - j$
In	$\emptyset$	$\emptyset$	$i < j$	$\mathcal{E}$	$i < j$
Out	$\emptyset$	$i < j$	$\mathcal{E}$	$\emptyset$	$i - j$
In	$\emptyset$	$\emptyset$	$i < j$	$\mathcal{E}$	$i < j$
Out	$\emptyset$	$i < j$	$\mathcal{E}$	$\emptyset$	$i - j$

Seul le calcul  $i+j$  est redondant dans  $B4$ . On remplace dans  $B3$ ,  $t1 := i+j$  Par

$u := i+j$   
 $t1 := u$

On remplace dans  $B4$   $i := i+j$  par  $i := u$  Ce qui donne :

**Variables actives**

	$B1$	$B2$	$B3$	$B4$	$B5$
Gen	$\emptyset$	$i, j$	$i, j$	$u$	$i, j$
Kill	$i, j$	$\emptyset$	$u, t1, t2$	$i$	$j, t2$
Out	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
In	$\emptyset$	$i, j$	$i, j$	$u$	$i, j$
Out	$i, j$	$i, j$	$u$	$i, j$	$\emptyset$
In	$\emptyset$	$i, j$	$i, j$	$u, j$	$i, j$
Out	$i, j$	$i, j$	$u, j$	$i, j$	$\emptyset$
In	$\emptyset$	$i, j$	$i, j$	$u, j$	$i, j$

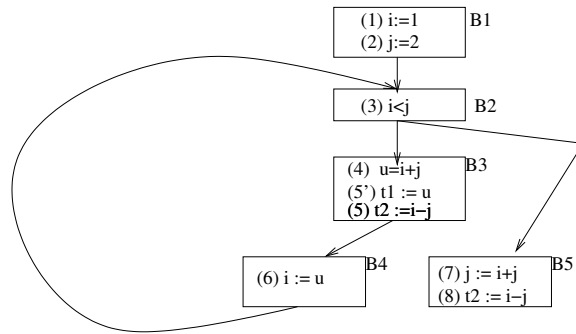


FIG. 2 – Graphe de flot de contrôle optimisé après E.R.

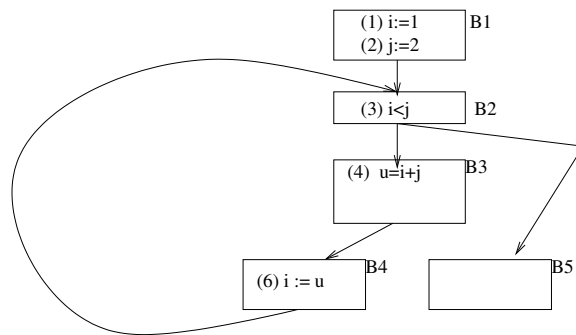


FIG. 3 – Graphe de flot de contrôle optimisé après suppression d'instructions inutiles

## 4 Exercice IV : Génération de code

– Dessiner la pile lors de l'exécution de  $g3$  en représentant uniquement le chaînage dynamique et le chaînage statique.

– On considère la procédure  $p1$ . Donner la séquence de code qui réalise  $z = g2(2)$ .

```

SUB R0,R0,R0
!évaluer et empiler le paramètre 2
ADD R1,R0,2
empiler(R1)
!réservation de 4 octets pour le résultat
ADD SP,SP,-4
!lien statique de ka procédure appelée
empiler(FP) ! LS(g2)
CALL main.p1.g2
ADD SP,SP,4
LD R2,[SP] ! le sommet de pile contient le résultat
ST R2,[FP-12]
ADD SP,SP,8

```

– On considère la procédure  $p2$ . Donner la séquence de code qui réalise  $z = y + p3(p)$ .

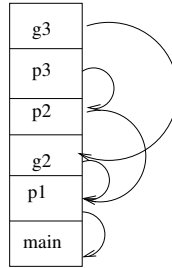


FIG. 4 – Lien statique

```

SUB R0,R0,R0
! on récupère y
LD R1,[FP+8] !LS(p2)=FP(p1)
LD R2,[R1-8]
! on appelle p3(p)
LD R4,[FP+20] ! adr(p)
empiler(R4)
LD R3,[FP+16] ! LS(p)
empiler(R3)
ADD SP,SP-4 ! pour le resultat de p3
empiler(FP) ! LS(p3)
CALL main.p1.p2.p3
ADD SP,SP,4
LD R3,[SP]
ADD R2,R3,R2
! on range le resultat dans z
LD R1,[FP+8]
ST R2,[R1-12]
ADD SP,SP,12

```

– On considère la procédure p3. Donner la séquence de code qui réalise `return p(x+x1+y)`.

```

LD R1,[FP+8] ! LS(p3)
LD R2,[R1+24]! param(x), a 20et 16 on a p et 12 le res
LD R1,[R1+8] ! LS(p2)=FP(p1)
LD R3,[R1-8] ! y
LD R1,[R1+8] ! LS(p1)=FP(main)
LD R4,[R1-4] ! x1
ADD R2,R3,R2 ! R2 :=y+x
ADD R2,R4,R2 ! R2 := x+y+x1
empiler(R2) ! empiler le paramètre
ADD SP,SP,-4 ! place pour le résultat
LD R3,[FP+16]! R3:=LS(p)
empiler(R3) ! le lien statique de la procédure appelée
LD R4,[FP+20]! R4:=adr(p)
CALL R4
ADD SP,SP,4
LD R4,[SP]

```

```

ST R4, [FP+12]
ADD SP, SP, 8
- On considère la procédure g2. Donner la séquence de code qui réalise  $y = p2(x1, g3)$ .
LD R1, [FP+8] ! LS(g2)=FP(p1)
LD R1, [R1+8] ! LS(p1)=FP(main)
LD R2, [R1-4] ! x1
empiler(R2)
SET R2, main.p1.g2.g3
empiler(R2) ! adr(g3)
empiler(FP) ! LS(g3)
ADD SP, SP, -4 ! pour le résultat
LD R1, [FP+8] ! LS(p2)=LS(g2)=FP(p1)
empiler(R1)
CALL main.p1.p2
ADD SP, SP, 4
LD R2, [SP]
ST R2, [R1-8]
ADD SP, SP, 16

```