Langages, Traducteurs et Sémantique, M1, 2005/2006
J.-C. Fernandez, Y. Lakhnech, L. Gonnord, F. Lagnier, L. Mounier

LTS

Les trois exercices sont indépendants. Le barème n'est donné qu'à titre indicatif.

Exercice I (9 points) - Sémantique opérationnelle

Partie A - Sémantique opérationnelle

On considère la syntaxe des expressions arithmétiques vue en cours en l'étendant avec deux nouveaux opérateurs : un opérateur de *pré-incrémentation* et un opérateur de *post-incrémentation*. La nouvelle syntaxe des expressions arithmétiques est la suivante :

$$a \quad ::= \quad n \mid x \mid a_1 + a_2 \mid \; ++x \mid x++$$

On note **Aexp'** l'ensemble de ces expressions arithmétiques. Informellement, la valeur de l'expression $x++$ est la valeur de la variable x, puis on associe à x l'ancienne valeur augmentée de 1. De la même manière, la valeur de l'expression $++x$ est obtenue en augmentant de 1 la valeur associée à x, puis retournant la nouvelle valeur. Par exemple, dans un état où x vaut 3, $x++$ vaut 3 et la nouvelle valeur de x est 4 tandis que $++x$ vaut 4 et la nouvelle valeur de x est 4.

On constate que l'évaluation des expressions peut modifier l'état.

Les configurations pour les expressions arithmétiques seront données par (**Aexp'** $\times$ **State**) $\cup$ ($\mathbb{Z} \times$ **State**) (on code la valeur de l'expression, et l'état obtenu par effet de bord).

1. Donnez la sémantique opérationnelle des expressions arithmétiques en supposant que les opérandes d'un opérateur binaire sont évalués de gauche à droite.
2. Que vaut l'expression $x + (x++)$ où x vaut initialement 3 ? Vérifier sur cet exemple la cohérence de vos règles.
3. On modifie les commandes en considérant la syntaxe suivante :

   $$S \quad := \quad x := a \mid x \mathrel{+}:= a \mid S_1; S_2$$

   Informellement, pour évaluer la commande $x \mathrel{+}:= a$ dans la mémoire σ,
   – on évalue tout d'abord a, on obtient ainsi une valeur v et une nouvelle mémoire σ',
   – puis on associe à x la valeur $v + \sigma(x)$.
   Donnez la sémantique opérationnelle pour les constructions $x := a$ et $x \mathrel{+}:= a$.
4. Vérifier sur $x \mathrel{+}:= ++y$ et sur $x \mathrel{+}:= ++x$ avec $\sigma = \begin{cases} x \mapsto 3 \\ y \mapsto 5 \end{cases}$
5. Pour chaque couple d'expressions ou de commandes, dire en justifiant, si elles sont sémantiquement équivalentes.

(a) $++x$ et $(x++)+1$

(b) $x := x++$ et $x := ++x$

(c) $x := x++$ et $x+ := 1$

(d) $x+ := a$ et $x := x + a$ pour $a \in \mathbf{Aexp'}$.

(e) $x+ := a$ et $x := a + x$ pour $a \in \mathbf{Aexp'}$.

Partie B Sémantique axiomatique

Comme nous l'avons vu dans la partie A, l'évaluation d'une expression modifie la mémoire. Dans cette partie, on va définir des triplets de Hoare pour les expressions de **Aexp'**. Ainsi, de manière analogue à ce qui a été vu en cours, on dira que le triplet $\{P\}\ a\ \{Q\}$ est valide si pour tout état σ et tout état σ', $\sigma \models P$ et $(a, \sigma) \longrightarrow (v, \sigma')$ implique $\sigma' \models Q$.

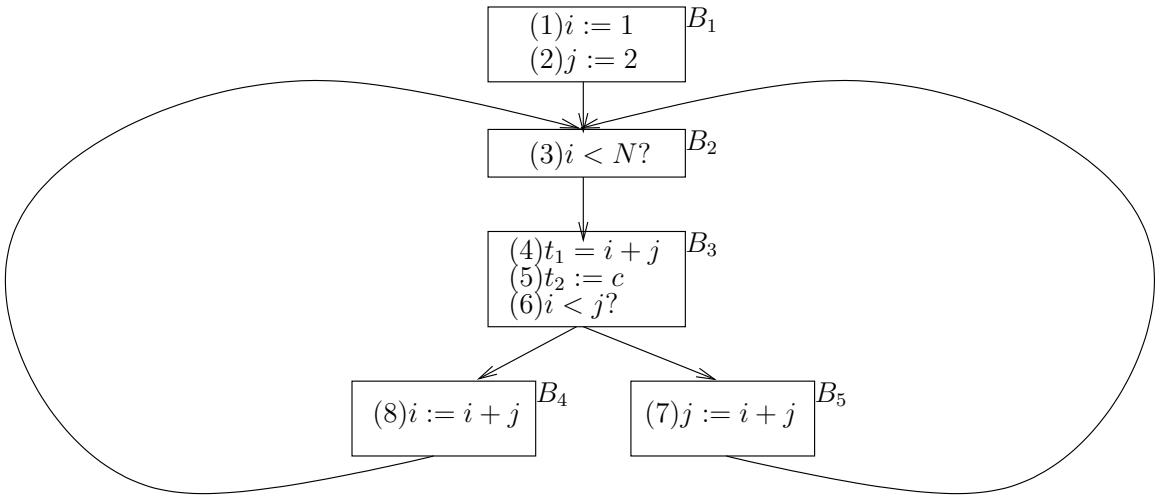
On obtient ainsi un calcul dont nous donnons quelques règles :

Axiomes	$\{P\}$	n	$\{P\}$
	$\{P\}$	x	$\{P\}$
Règles	$\{P[x+1/x]\}$	$x++$	$\{P\}$

Donner une règle pour la pré-incrémentation et une pour l'addition. Remarque ?

Exercice II : Optimisation (6 points)

On considère le graphe de flot de contrôle suivant :



Dans la suite, on fera attention à bien prendre en compte les tests

Expressions disponibles

1. Calculez les ensembles $Gen(b)$ et $Kill(b)$ pour chaque bloc de base b.

2. Calculez les ensembles $In(b)$ et $Out(b)$ pour chaque bloc de base b.
3. Supprimez les calculs redondants. Pour chaque expression disponible e à l'entrée d'un bloc de base, si cette expression existe dans ce bloc de base et si les opérandes ne sont pas modifiés entre le début du bloc et la position de l'expression, alors celle-ci est redondante. Pour la supprimer, on crée une nouvelle variable u, et par une recherche en arrière, on localise les instructions de la forme $x := e$, que l'on remplace par $\begin{cases} u := e \\ x := u \end{cases}$ et on remplace par u le calcul de l'expression, dans le bloc où ce calcul est redondant.

Dans la partie suivante, on considère le graphe ainsi modifié.

Variables actives

1. Calculez les ensembles $Gen(b)$ et $Kill(b)$ pour chaque bloc de base b.
2. Calculez les ensembles $In(b)$ et $Out(b)$ pour chaque bloc de base b.
3. Suppression des instructions inutiles qui sont les affectations $x := e$ telles que x est inactive en fin de bloc et n'est pas utilisée dans le bloc après l'instruction.

Exercice III : Génération de code (5 points)

On considère le programme suivant :

```
main() {
  int x ;
  void P1() {
    int y ;
    void Q2 () {
      int z ;
      z = y+x;
    } /*  end Q2 */
    void P2() {
      y=3;
      Q2 ();
    } /*  end P2 */
    P2();
  } /* end P1 */
  x=2;
  P1();
}/* end main */
```

1. Dessinez la pile lors de l'exécution de `Q2`.
2. Dans la procédure `P2`, donnez la séquence d'instructions associée à `y=3`.
3. Dans la procédure `P2`, donnez la séquence d'instructions associée à l'appel de `Q2`.
4. Dans la procédure `Q2`, donnez la séquence d'instructions associée à `z=y+x`.