Université Bordeaux I **Master Informatique 2013-2014**
IN7M13EX Compilation, Examen
Lionel Clément **11/12/2013 – de 14h00 à 15h30**

Examen de Compilation

Ce sujet comporte 4 pages
documents manuscrits, dictionnaire de langue et feuilles de TD autorisés

Exercice 1 : 5 points

Une expression de type est définie inductivement de la sorte :
- Un type de base : *integer*, *boolean* est une expression de type ;
- Une variable de type $'x$ est une expression de type ;
- Un symbole α est une expression de type ;
- Si T_1 et T_2 sont deux expressions de type, $T_1 \times T_2$ est une expression de type ;
- Si T_1 et T_2 sont deux expressions de type, $T_1 \rightarrow T_2$ est une expression de type ;
- Si T est une expression de type, *constructeur*(T) est une expression de type.
  Où *constructeur* vaut *pointer*, *list* ou *record*

Dans le code suivant où nous avons deux déclarations différentes de la fonction `Add`.

```
1  program Adhoc;
2
3  function Add(x, y: Int ): Int;
4  begin
5      Add := x + y
6  end;
7
8  function Add(s, t: String ): String;
9  begin
10     Add := Concat( s, t )
11 end;
12
13 begin
14     Writeln(Add(1, 2));
15     Writeln(Add('Hello, ', 'World!'));
16 end.
```

Questions

1. Pourquoi la fonction `Add` est-elle polymorphe ?
2. Expliquer comment se réalise le contrôle de type de l'expression `Add(Add(x, y), z)` ? Où `x`, `y`, et `z` sont de type `Int`.
3. Dessiner une représentation arborescente du type de la fonction `Add`

Exercice 2 : 5 points

Soit G une grammaire algébrique $(\{E\}, \{:=, (,), a\}, E, R)$ dont les règles de production R sont les suivantes :

E → E := E
E → (E)
E → a

Le tableau suivant résume l'analyse Earley de l'expression $a := a := (a)$

état	lu	à lire	items
0	ϵ	$a := a := (a)$	0, 0, E → • E := E 0, 0, E → • (E) 0, 0, E → • a
1	a	$:= a := (a)$	0, 1, E → a • 0, 1, E → E • := E
2	$a :=$	$a := (a)$	0, 2, E → E := • E 2, 2, E → • E := E 2, 2, E → • (E) 2, 2, E → • a
3	$a := a$	$:= (a)$	2, 3, E → a • 0, 3, E → E := E • 2, 3, E → E • := E 0, 3, E → E • := E
4	$a := a :=$	(a)	2, 4, E → E := • E 0, 4, E → E := • E 4, 4, E → • E := E 4, 4, E → • (E) 4, 4, E → • a
5	$a := a := ($	$a)$	4, 5, E → (• E) 5, 5, E → • E := E 5, 5, E → • (E) 5, 5, E → • a
6	$a := a := (a$	$)$	5, 6, E → a • 4, 6, E → (E •) 5, 6, E → E • := E
7	$a := a := (a)$	ϵ	4, 7, E → (E) • 2, 7, E → E := E • 0, 7, E → E := E • 4, 7, E → E • := E 2, 7, E → E • := E 0, 7, E → E • := E

Questions

1. Expliquer comment peut-on conclure de ce tableau que la séquence est reconnue par l'analyseur.
2. La séquence produit une analyse ambigüe. En précisant que l'opérateur := est associatif à droite, peut-on modifier l'algorithme de sorte que seule l'analyse attendue soit faite ?

Remarque : Nous pourrons concentrer la démonstration sur la réduction

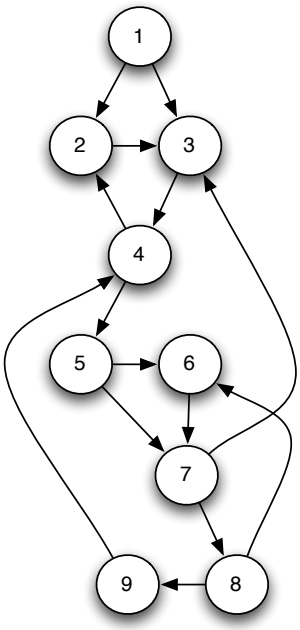
$$\frac{(0, 0, E \to \bullet E := E)(0, 3, E \to E := E\bullet)}{(0, 3, E \to E\bullet := E)}$$

et la transition

$$\frac{(2, 3, E \to E\bullet := E)}{(2, 4, E \to E := \bullet E)}$$

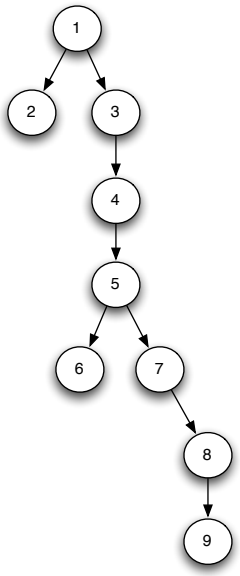
Exercice 3 : 4 points

Soit le graphe de flot de contrôle suivant, où 1 représente le bloc initial :



1. Calculer le graphe des dominants

Réponse



Sommets	1	2	3	4	5	6	7	8	9
prédécesseurs	$\emptyset$	1,4	1,2,7	3,8	4	5,9	5,6	7	8
	1	1 2	1 ~~2~~ 3	1 ~~2~~ 3 4	1 ~~2~~ 3 4 5	1 ~~2~~ 3 4 5 6	1 ~~2~~ 3 4 5 ~~6~~ 7	1 ~~2~~ 3 4 5 ~~6~~ 7 8	1 ~~2~~ 3 4 5 ~~6~~ 7 8 9

2. Est-ce que ce graphe de flot de contrôle contient une ou plusieurs boucles ? Si oui, lesquelles ?

Réponse

(4, 5, 6, 7, 8, 9)

Exercice 4 : 6 points

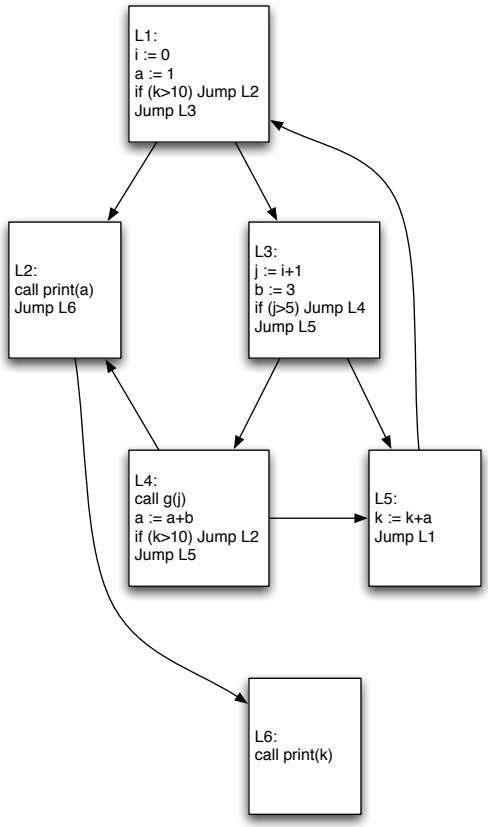
On considère les lignes de code suivantes :

```
L1:
i := 0
a := 1
if (k>10) Jump L2
Jump L3
L2:
call print(a)
Jump L6
L3:
j := i+1
b := 3
if (j>5) Jump L4
Jump L5
L4:
call g(j)
a := a+b
if (k>10) Jump L2
Jump L5
L5:
k:=k+a
Jump L1
L6:
call print(k)
```

Questions

1. Dessiner le graphe de flot de contrôle correspondant.

Réponse



2. Pour chaque bloc de contrôle, indiquer les variables vivantes en entrée et en sortie.

Réponse

	Use	Def	In	Out
B1	k	i, a	k	a, i, k
B2	a	$\emptyset$	a, k	k
B3	i	j, b	i, a, k	j, a, k, b
B4	j, a, k, b	a	j, a, k, b	a, k
B5	a, k	k	a, k	k
B6	k	$\emptyset$	k	$\emptyset$

3. Comment peut-on utiliser ce résultat pour optimiser le code ?

Réponse

En utilisant le fait que i et j ne sont jamais vivantes en même temps, on peut exploiter le même registre pour les enregistrer
$b = 3$ dans le bloc $B3$ peut être exporté dans une en-tête de la boucle $(B1, B3, B4, B5)$