Université d'Orléans — DEA d'Informatique

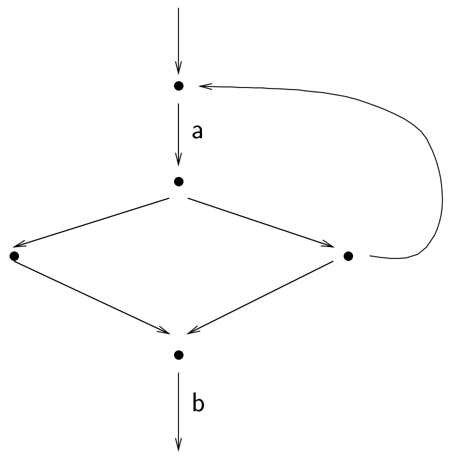
Module "compilation avancée"

Sujet d'examen proposé par C. Eisenbeis et F. Thomasset : corrigé
(Session de rattrapage)

Avril 2001

Exercice 1

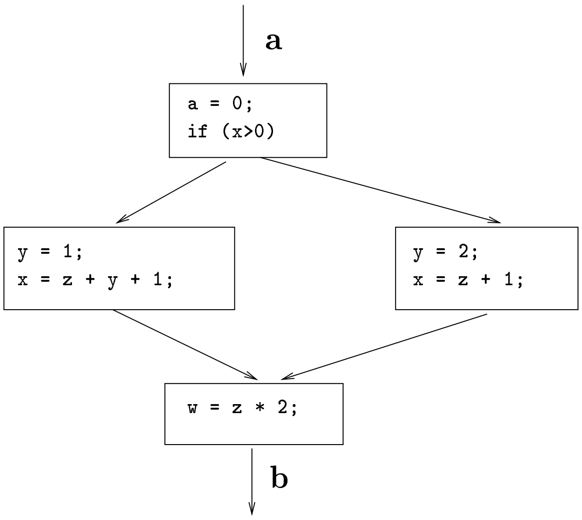
Question 1.1 Voici un contre-exemple, dans lequel les conditions 1. et 2. sont satisfaites, mais pas 4. :



On peut alors sortir, sans passer par b, de la "région" dominée par a et post-dominée par b ; cette région n'a donc pas *"une seule sortie"*. On construit de même un contre-exemple concernant la condition 3.

Question 1.2 Le nombre de régions est borné par le nombre de couples (a, b), soit un ordre de grandeur $O(|E|^2)$ ($|E|$ étant le nombre d'arcs).

Le couple **(a,b)** dans le graphe ci-dessous forme une région SESE canonique ; les autres régions sont formées des blocs de base eux-mêmes.



Question 1.3 Supposons que $R1 = (a, b)$ et $R2 = (c, d)$ sont deux régions canoniques distinctes, et qu'elles contiennent toutes deux un nœud n. Comme a et c dominent n, l'un de ces 2 arcs doit dominer l'autre : supposons donc que a domine c. De même entre b et d : l'un des deux doit post-dominer l'autre. Il convient d'envisager les différents cas de figure possibles :

- b post-domine d : alors la région (a, b) contient (c, d).
- d post-domine b ; nous avons 3 cas possibles, selon les situations respectives de b et c.
  - $b = c$: l'arc $b = c$ devrait alors à la fois dominer et post-dominer le nœud n.
  - $b \neq c$, et c domine b.
    Comme a domine b, b ne domine pas a, donc il existe un chemin de i à a ne contenant pas b ; de plus, comme en outre a domine c et c domine b, il existe un chemin de i à c contenant a, mais pas b ; soit Q la partie de ce chemin de a à c.
    Montrons que b post-domine c : soit P un chemin de c au nœud de sortie s ; P doit contenir b, sinon en joignant Q et P, on obtiendrait un chemin de a à s ne passant pas par b, contredisant l'hypothèse que b post-domine a (par définition d'une région). Donc b post-domine c.
    De même, c post-domine a ; sinon, il existe de i à s passant par a et b, mais sans passer par c, contrairement à l'hypothèse que c domine b.
    Montrons que (c, b) définit une région SESE. Considérons un cycle passant par b ; il passe par a puisque (a, b) est une région SESE, et donc il contient un chemin P' de a à b ; comme a domine c et c domine b, P' doit contenir c. Par conséquent, tout cycle contenant b contient également c. De même, tout cycle contenant c doit contenir d, et donc aussi b.
    Ainsi (c, b) définit une région SESE, et a ne post-domine pas c ; (a, b) n'est pas une région canonique, contrairement à l'hypothèse.
  - $b \neq c$, et c ne domine pas b ; ceci implique l'existence d'un chemin de i à b ne contenant pas c.
    Supposons que c ne post-domine pas b. Alors il existe un chemin P'' de i à s passant par b, mais pas par c ; puisque d post-domine b, ce chemin P'' doit contenir d, masi cela contredit l'hypothèse que c domine d (car (c, d) est une région).
    Donc c post-domine b. Comme b post-domine le nœud n, c post-domine également n ; alors c ne peut pas aussi dominer n, contredisant l'appartenance de n à la région (c, d).

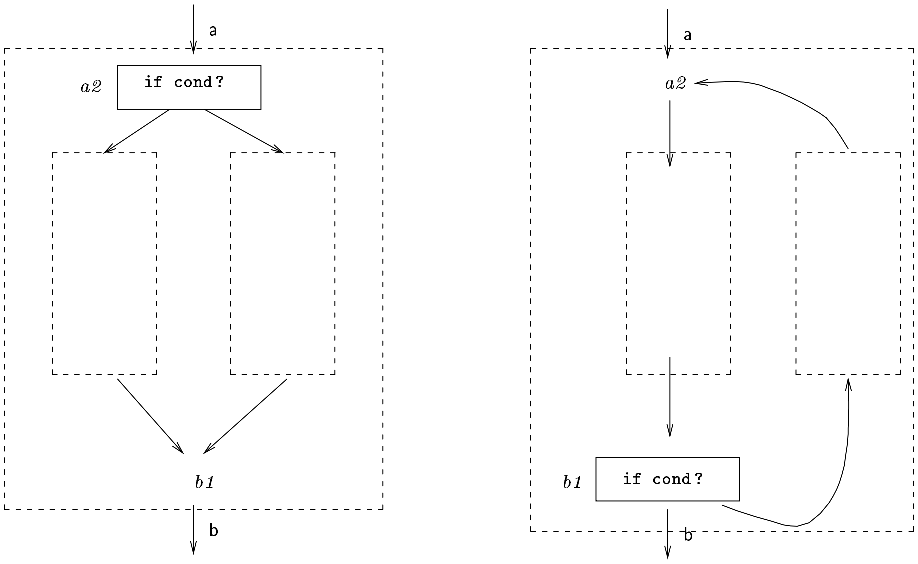
Question 1.4 En supposant un arc d'entrée issu de i, et un arc de sortie menant à s, il existe une région contenant toutes les autres. Étant donné une région canonique R quelconque, les régions qui la contiennent sont ordonnées par inclusion d'après ce qui vient d'être démontré ; il en existe une plus petite que les autres, et contenant R : cette région unique est le père de R dans l'arbre des régions canoniques.

Question 1.5 Soit $R = (a, b)$ une région : comment reconnaître sa structure?

Bloc de base : Tout nœud n de R post-domine a et domine b.

if-then-else : Soit $a = (a1, a2)$ l'arc d'entrée ; il mène à un nœud $a2$ avec deux successeurs. L'arc de sortie $b = (b1, b2)$ provient d'un nœud à deux prédécesseurs. Chaque arc issu de $a2$ est arc d'entrée d'une région SESE, dont l'arc de sortie mène à $b1$. Chaque arc menant à $b1$ est l'arc de sortie d'une région SESE, dont l'arc d'entrée est issu de $a2$. On peut également admettre des "if-then" : l'une des régions ci-dessus est alors vide.

boucle : Soit $a = (a1, a2)$ l'arc d'entrée, et $b = (b1, b2)$ l'arc de sortie ; $a2$ a deux prédécesseurs et un successeur, et $b1$ un prédécesseur et deux successeurs. L'unique arc issu de $a2$ est l'entrée d'une région dont l'arc de sortie mène à $b1$; l'un des arcs issus de $b1$ est l'arc de sortie de la région R, et l'autre est l'arc d'entrée d'une région SESE, dont l'arc de sortie mène à $a2$. Selon que l'une ou l'autre des sous-régions est vide, on peut obtenir la caractérisation d'un repeat ou d'un while.



Question 1.6 Chaque région peut être considérée comme un graphe cfg sur lequel on peut résoudre le problème posé, avant de passer au niveau supérieur. On notera que les régions peuvent des sous-graphes réductibles, même si le graphe du programme est irréductible.

Pour plus d'informations on pourra consulter les publications du groupe de K. Pingali (Cornell) :
Richard Johnson. *Efficient Program Analysis using Dependence Flow Graphs.*
Ph.D. thesis, Cornell University, August 1994.
`http://simon.cs.cornell.edu/Info/Projects/Bernoulli/`.
Richard Johnson, Keshav Pingali. Dependence-Based Program Analysis.
SIGPLAN Notices, 28(6):78–89, June 1993. (PLDI'93)
`http://simon.cs.cornell.edu/Info/Projects/Bernoulli/papers/pldi93.ps`.

Exercice 2

Question 2.1 Il y a une dépendance et une anti-dépendance de `S1` sur elle-même ; en effet $\langle \texttt{S1}, n+i \rangle$ dépend et anti-dépend de $\langle \texttt{S1}, n-i+1 \rangle$.

Question 2.2 Il y avait deux solutions pour paralléliser : soit on construisait deux boucles parallèles, en itérant d'abord de 1 à n, puis de $n + 1$ à $2\,n$; soit on regroupait les itérations de manière à n'avoir que des dépendances indépendantes de la boucle.
Donc première solution :

```
DOALL I = 1, N
   A(I) = A(N*2-I+1) + 1
END DOALL
DOALL I = N+1, N*2
   A(I) = A(N*2-I+1) + 1
END DOALL
```

Deuxième solution :

```
DOALL I = 1, N
   A(I) = A(N*2-I+1) + 1
   A(N*2-I+1) = A(I) + 1
END DOALL
```