Examen – Compilation

Yves Robert

Mercredi 9 Janvier 2002, 13h-16h

Tous les exercices sont indépendants.

1 Equations data-flow

Considérons le code suivant :

```
<S0>          a := 1
<S1>          b := 2
<S2>    L1    e := 0
<S3>          c := a+b
<S4>          d := c-a
<S5>          if (...) goto L3
<S6>    L2    d := b*d
<S7>          if (...) goto L3
<S8>          d := a+b
<S9>          e := e+1
<S10>         goto L2
<S11>   L3    b := a+b
<S12>         e := c-a
<S13>         if (...) goto L1
<S14>         a := b*d
<S15>         b := a-d
```

Question 1 *Donner le graphe de flot de contrôle (CFG) avec les blocs de base.*

Question 2 *Donner les variables vivantes à la sortie de chaque bloc.*

Question 3 *Donner les expressions disponibles.*

Question 4 *La propagation de constante est-elle possible ? si oui, la mettre en oeuvre.*

Question 5 *Existe-t-il des sous-expressions communes ? si oui, les éliminer.*

2 Allocation de registres

Considérons le graphe de flot de contrôle donné à la Figure 1.

Question 6 *Calculer le graphe d'interférences de registres, le nombre minimal de registres nécessaires, et l'allocation des variables aux registres.*

Question 7 *A quelles conditions peut-on échanger les places de deux instructions consécutives $x = y + z$ et $t = u + v$ dans le code intermédiaire, sans modifier le résultat final ?*

Question 8 *Montrer comment diminuer le nombre de registres nécessaires sur le graphe de flot de contrôle donné à la Figure 1 en effectuant uniquement des inversions légales d'instructions.*

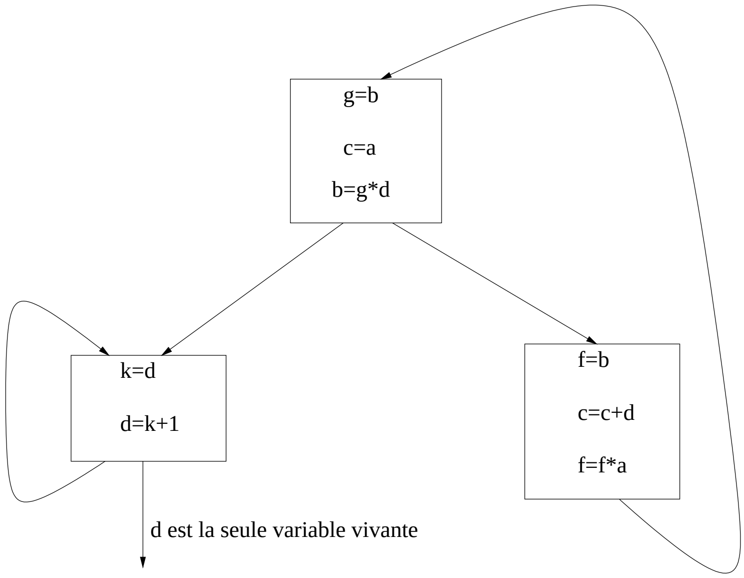

FIG. 1 – Graphe de flot de contrôle, avec les variables vivantes en sortie

3 Boucles, invariants de boucles, et variables d'induction

Une boucle dans le graphe de flot de contrôle $G = (V, E)$ (orienté) est un ensemble de noeuds S (incluant un noeud particulier h) qui vérifie les propriétés suivantes :

- $\forall s \in S$, il existe un chemin orienté dans G de s à h
- $\forall s \in S$, il existe un chemin orienté dans G de h à s
- $\forall g \in V - S, \quad (\exists s \in S, \quad (g, s) \in E) \Longrightarrow (s = h)$

Un graphe de flot de contrôle contient un noeud particulier s_0 sans prédécesseur. On dit qu'un noeud d domine un noeud n si tous les chemins de s_0 à n passent par d.

Question 9 *Donner un algorithme pour déterminer $D(n)$, l'ensemble des noeuds qui dominent n (on donnera des règles locales pour le calcul de $D(n)$ et on en déduira l'algorithme).*

Un pré-entête d'une boucle est un noeud dont l'unique successeur est h et tel que l'unique prédécesseur de h qui n'est pas dans S est le pré-entête.

Question 10 *Montrer qu'il est toujours possible de rajouter un pré-entête à une boucle.*

Dans la suite, on va essayer de repérer les instructions de S dont le résultat est le même à chaque pas de la boucle et de les ajouter au pré-entête. Bien évidement, cette opération est extrêmement complexe et on se contentera de conditions suffisantes.

Supposons que la définition $d: \quad t = a + b$ apparaît dans la boucle et que les valeurs de a et b ne sont pas modifiées par l'exécution de la boucle. On dira que la définition d est invariante par S et que a et b sont des invariants de la boucle S. On souhaite ajouter d au pré-entête de la boucle, pour éviter des calculs redondants.

Question 11 *Donner des conditions suffisantes pour affirmer que $d: \quad t = a + b$ est invariante par S.*

Question 12 *Supposons que $d: \quad t = a + b$ est invariante par S. Donner des conditions suffisantes pour que le déplacement de d dans l'en-tête ne modifie pas le résultat final.*

Dans la suite, nous supposons que la durée d'exécution d'une multiplication est largement supérieure à celle d'une addition. Nous dirons que i est une variable d'induction primaire de la boucle S si et seulement si les seules instructions de S modifiant i sont de la forme $i = i + c$ ou $i = i - c$, où c est un invariant de boucle.

Nous dirons que j est une variable d'induction indirecte associée à i si et seulement si j n'est affectée qu'une seule fois dans S et qu'après cette affectation, la valeur de j est toujours de la forme $d * i + e$, où d et e sont des invariants de S.

Question 13 *Quel est l'intérêt de repérer les variables d'induction indirecte ?*

Question 14 *Donner des conditions suffisantes permettant d'affirmer qu'une variable est une variable d'induction indirecte.*

Question 15 *Dans le code suivant, montrer les variables d'induction primaire et les variables d'induction indirecte. Montrer comment optimiser le code en réalisant un minimum de multiplications à l'intérieur de la boucle. Nous supposerons que b, d et e sont des invariants de boucle.*

$$
\begin{array}{ll}
s = 0 & \\
i = 0 & \\
L_1 : & \text{if } i \geq n \text{ goto } L_2 \\
 & j = 4 * i \\
 & k = j + e \\
 & l = d * k \\
 & x = M[l] \\
 & s = s + x \\
 & i = i + 1 \\
 & \text{goto } L_1 \\
L_2 : & \ldots.
\end{array}
$$

4 Enregistrements d'activation

On va montrer qu'il est possible de limiter le nombre d'enregistrements d'activation lors de l'évaluation de certaines fonctions récursives.

Soit f une fonction dont le corps contient un appel à une fonction g. On dira qu'un appel à g est terminal si la dernière chose réalisée par le code de f est cet appel à g. Une fonction est récursive terminale si tous les appels récursifs sont terminaux.

Question 16 *Proposer des fonctions récursives terminales pour calculer $puiss(n) = 2^n$ et $fact(n) = n!$. Montrer l'évolution de la pile contenant les enregistrements d'activation de $fact$ lors du calcul de $4!$. Indiquer comment une gestion astucieuse des enregistrements d'activation peut permettre de limiter leur nombre (dans le cas d'appels récursifs terminaux).*

Question 17 *On suppose maintenant qu'on dispose de deux fonctions f et g dont les pseudo codes sont donnés ci-dessous :*

$$f(x : int, \quad acc : int) \quad : \quad int \qquad \{if \;\; x > 0 \;\; then \;\; g(x-1, acc+2) \quad else \quad acc \quad endif\};$$

$$g(x : int, \quad acc : int) \quad : \quad int \qquad \{if \;\; x > 0 \;\; then \;\; f(x-1, acc+5) \quad else \quad acc \quad endif\};$$

Dans ce cas, les appels à f et g ne sont pas directement récursifs, mais ils sont terminaux. Comment étendre la technique développée à la question précédente dans ce contexte (on pourra différencier les cas où l'invocation initiale de ces fonctions se fait à travers un appel à f ou à g) ?