Devoir à la maison, cours de Master

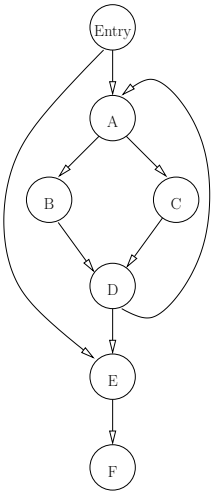
Alain Darte/Fabrice Rastello

23 novembre 2007

Les 5 premières parties sont des exercices d'application du cours, avec quelques questions plus pointues, la dernière est plus théorique.

1 Autour des graphes de flot de contrôle

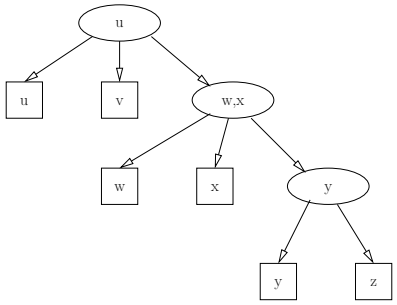
Question 1. *Calculer, en justifiant, la frontière de dominance itérée ("iterated dominance frontier", iDF) du nœud C dans le CFG suivant :*



Question 2. *Quelle est la différence entre la frontière de dominance itérée et l'ensemble de points de confluence itéré ("iterated join set", iJS) (définition dans [2]). Illustrer sur l'exemple. Discuter aussi de la différence entre iDF et iJS pour la construction de SSA.*

Question 3.

- *Donner le graphe de flot de contrôle (CFG) de la forêt de nids de boucles ("loop nested forest") suivante qui utilise le modèle de Steensgaard [4]. Y a-t-il plusieurs solutions ?*

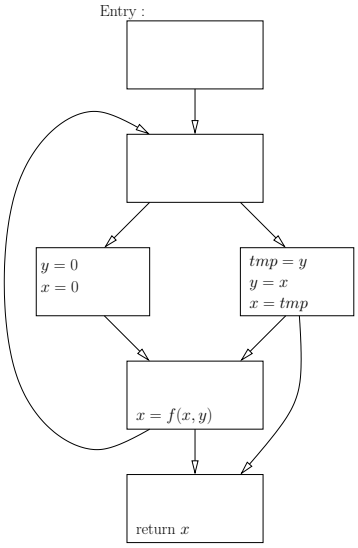


- *Donner la forêt correspondante dans le modèle de Havlak. Y a-t-il plusieurs solutions ?*

– *Modifier le CFG (en changeant la cible de certains arcs) de sorte que le graphe résultant soit réductible. Quelle est la complexité de l'algorithme de détection de boucles dans le cas réductible ? Justifier.*

2 Autour de la forme SSA

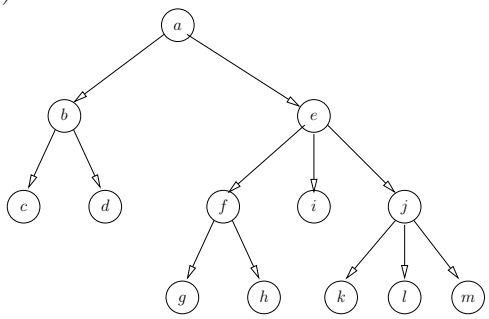
Question 1. *Passer le code suivant en "static single assignment form" (SSA form) avec propriété de dominance :*



Question 2. *Que donne l'algorithme "hash based global value numbering" de Briggs et al. [1] sur ce code ? Discuter le cas des ϕ.*

3 Autour de l'étiquetage de graphes

Question 1. *Proposer un étiquetage (labeling) de graphe [3] simple pour le graphe suivant permettant de répondre efficacement à une requête de plus proche ancêtre commun ("least common ancestor", LCA, ou "nearest common ancestor", NCA). Appliquer à la requête $\mathsf{lca}(f, \mathsf{lca}(l, m))$.*



4 Autour du pipeline logiciel

On refait les calculs sur l'exemple du cours :

```
L400 :
  ld[r26], r27
    nop
  add r27, 6740, r26
  ld 0x1A54[r27], r27
    nop
  sub.f r27, r25, r0
  bne L400
    nop
L399 :
```

à compiler pour le processeur LANai 3.0 (voir les transparents du cours sur le pipeline logiciel pour ses spécificités). En bref, ce processeur possède une unique unité générale, pipelinée, telle que toute opération arithmétique est de durée 1 et toute autre opération de durée d (on considérera dans les exemples le cas $d = 2$ et le cas $d = 3$), les effets de bord (modification des registres ou branchement) étant pris en compte à la fin du d-ème cycle.

On suppose dans un premier temps que l'allocation des registres est déjà effectuée (ainsi, les deux loads du programme écrivent dans r27 par exemple).

Question 1. *Analyser à nouveau le code et donner le graphe de dépendances correspondant, avec délais et distances de dépendances tous deux sur les arcs. Ne pas oublier les anti dépendances et "output" dépendances.*

Question 2. *Donner un pipeline logiciel optimal pour ce code et cette architecture. Même question si on avait deux unités semblables au lieu d'une. Même question avec trois unités dédiées, une par catégorie d'opérations (une pour les opérations mémoire load/store, une pour les branchements, une pour les opérations arithmétiques).*

Question 3. *Revisiter la question précédente si on s'autorise à changer l'allocation de registres (sauf r0 néanmoins dont dépend le branchement). Expliquer les différences (ou non) pour l'exemple et dans le cas général.*

5 Équations récurrentes uniformes et des boucles

Question 1. *Donner un exemple de système d'équations récurrentes uniformes, défini sur un espace de dimension 3, pour lequel la décomposition de Karp, Miller et Winograd ne nécessite qu'un appel récursif (profondeur 1 donc). Ordonnancer et ré-écrire ce système à l'aide de boucles, une boucle séquentielle externe et des boucles parallèles internes. Même question pour la profondeur 2 et deux boucles séquentielles externes. Même*

question pour la profondeur 3 et trois boucles séquentielles. Même question pour la profondeur 4 (dans ce cas, le système est considéré comme non calculable).

On traite à présent des boucles DO dans un langage impératif (style Fortran). On suppose que les dépendances sont représentées non pas à l'aide de vecteurs de direction (vecteur dont les composantes appartiennent à l'ensemble $\mathbb{Z} \cup (\mathbb{Z} \times \{+, -, *\})$ mais simplement par le **niveau** des dépendances. Si une opération $(T, \vec{j})$ dépend d'une opération $(S, \vec{i})$, on ne considère que les dimensions correspondant à des boucles communes et on appelle niveau la première dimension commune pour laquelle $\vec{i}$ et $\vec{j}$ diffèrent (si $\vec{i} = \vec{j}$ sur les dimensions communes, on dit que le niveau est ∞, la dépendance a lieu dans le corps de la boucle, pas entre deux différentes itérations). En d'autres termes, un niveau égal à i correspond à un vecteur de direction dont les $i-1$ premières composantes valent 0, la i-ème vaut 1, et les autres valent $*$, sauf dans le cas du niveau ∞ qui correspond à un vecteur de direction nul.

Question 2. *Reprendre l'exemple des transparents du cours avec les vecteurs de direction, calculer les niveaux de dépendances et, avec cette information sur les dépendances, le ré-écrire en identifiant boucles séquentielles et parallèles.*

Question 3. *En simplifiant l'algorithme de Karp, Miller et Winograd, proposer un algorithme de parallélisation de boucles, élémentaire, qui traite du cas de dépendances représentées par niveau.*

6 Coloration des graphes et coalescing

Soit $G = (V, E)$ un graphe non orienté, de sommets V et arêtes E. On note $\chi(G)$ le nombre chromatique de G, $\Delta(G)$ le degré maximal d'un sommet de G, $\delta(G)$ le degré minimal d'un sommet de G et $\lambda(G) = \max_{G' \subseteq G} \delta(G')$.

On s'intéresse aux procédés de coloration de G par schéma d'élimination, c'est-à-dire qu'étant donné un ordre des sommets $v_1, \ldots, v_n$, on colorie les sommets par indices décroissants en choisissant pour v_i une couleur non encore utilisée par ses voisins déjà coloriés (c'est-à-dire d'indice supérieur).

Question 1. *Expliquer pourquoi* $\chi(G) \leq \Delta(G) + 1$.

Question 2. *Montrer que* $\chi(G) \leq \max_i d^o(v_i, G_i) + 1$ *où* $d^o(v_i, G_i)$ *est le degré de* v_i *dans* $G_i = G \setminus \{v_1, \ldots, v_{i-1}\}$ *le sous-graphe partiel de* G *induit par les sommets* v_j, $j \geq i$.

La quantité $\max_i \mathrm{d}^o(v_i, G_i)$ est définie pour un ordre particulier des sommets $v_1, \ldots, v_n$. On peut faire de même pour tout autre ordre, donné par une permutation π de $[1..n]$, et on définit alors $\mathrm{col}(G)$ comme le minimum parmi toutes ces permutations :

$$\mathrm{col}(G) = 1 + \min_{\pi} \max_{i} \mathrm{d}^o(v_{\pi(i)}, G_{\pi(i)}) \text{ où } G_{\pi(i)} = G \setminus \{v_{\pi(1}, \ldots, v_{\pi(i-1)}\}$$

On a donc $\chi(G) \leq \mathrm{col}(G)$.

Question 3. *Montrer que si* π *est une permutation de* $[1..n]$ *telle que* $v_{\pi(i)}$ *est choisi de degré minimal dans* $G_{\pi(i)} = G \setminus \{v_{\pi(1)}, \ldots, v_{\pi(i-1)}\}$, *alors* $col(G) = 1 + \max_i \delta(G_{\pi(i)}) = 1 + \lambda(G)$.

Question 4. *Montrer que si G est chordal, alors $\chi(G) = col(G)$ et qu'on peut trouver une telle coloration optimale à partir de tout ordre $v_1, \ldots, v_n$ tel que le degré de v_i dans $G \setminus \{v_1, \ldots, v_{i-1}\}$ est inférieur strictement à $\chi(G)$.*

On s'intéresse à présent au problème de coalescing d'une unique affinité ("copie"). Étant donné un graphe $G = (V, E)$, k-colorable, et deux sommets particuliers u et v, la question est de décider s'il existe une coloration de G avec k couleurs telle que u et v ont la même couleur.

Question 5. *Montrer que pour répondre à la question, on peut commencer par fusionner u et v et se ramener au cas d'un graphe dont tous les sommets sont de degré supérieur ou égal à k.*

On suppose à présent que G est k-colorable, chordal et que tous ses sommets, sauf éventuellement u et v, sont de degré supérieur ou égal à k.

Question 6. *Montrer que G est un graphe d'intervalles. Pour cela, on montrera que G a exactement deux sommets simpliciaux (u et v), qu'on peut définir un schéma d'élimination simplicial avec $v_1 = u$, $v_n = v$, puis que cet ordre définit un ordre partiel pour le graphe complémentaire (propriété manquante pour qu'un graphe chordal soit un graphe d'intervalles).*

Question 7. *Montrer que le schéma d'élimination défini en choisissant à chaque étape un sommet v_i de degré minimal dans $G \setminus \{v_1, \ldots, v_{i-1}\}$ fournit un schéma d'élimination simplicial, donc une représentation des sommets du graphe sous forme d'intervalles.*

Si, dans la représentation par intervalles de G, il n'y a pas exactement k intervalles en vie à certains points, on rajoute des intervalles artificiels ("dummy") de longueur 1. On peut supposer alors que G est de "largeur" k en tout point.

Question 8. *Montrer que G est k-colorable avec u et v de la même couleur si et seulement si il existe une séquence d'intervalles contigus (chaque intervalle finit là où commence le suivant) telle que le premier est u et le dernier est v. Essayer d'en déduire un algorithme rapide permettant de décider s'il existe une coloration de G avec k couleurs, avec u et v de la même couleur, et fournissant une telle coloration.*

Références

[1] Preston Briggs, Keith D. Cooper, and L. Taylor Simpson. Value numbering. *Software Practice and Experience*, 27(6) :701–724, 1997.

[2] R. Cytron, J. Ferrante, B. Rosen, M. Wegman, and K. Zadeck. Efficiently computing static single assignment form and the control dependence graph. *ACM Transactions on Programming Languages and Systems*, 13(4) :451–490, 1991.

[3] Dov Harel and Robert Endre Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM J. Comput.*, 13(2) :338–355, 1984.

[4] G. Ramalingam. On loops, dominators, and dominance frontiers. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 24(5) :455–490, 2002.