Analyse data-flow, frontière de dominance

bogdan.pasca@ens-lyon.fr

16 Nov 2010

Note : Thanks to Boissinot Benoit for his contribution to this subject

1 Analyse data-flow

1.1 Dominance

On rappelle ce qu'est le *graphe de flot de contrôle* (CFG) d'un programme :
- Chaque instruction est représentée par un *nœud* du graphe CFG.
- Si une instruction x peut être suivie d'une instruction y, il y a un arc de x vers y.
- Une chaîne de nœuds qui ont tous un seul successeur et un seul prédécesseur (sauf le premier et le dernier) est appelée *bloc de base* (*basic block*).
- Le nœud de début (le seul n'ayant pas de prédécesseur) est appelé *nœud d'entrée* (*entry*).

Q 1.1 Voici un programme C :

```
p = 0;
s = 1;
t = 5;
r = A[s];
if (r == 0) {
  while (t < 30) {
    t = t + 3;
    t = t * 2;
    p = p + A[t];
  }
} else {
  p = r;
}
f = 2 * p;
```

Dessinez le graphe de flot de contrôle de ce programme. Éliminez d'abord les structures `while` et `if...else` en les remplaçant par des branchements conditionnels (`if...goto`) et inconditionnels (`goto`). Indiquez les blocs de base.

La relation de *dominance* dans un graphe de flot de contrôle est définie comme suit : on dit que le nœud x domine y ssi tout chemin du nœud d'entrée à y passe par x. Cette relation est un ordre partiel sur les nœuds du graphe de flot de contrôle ; sa représentation graphique est un arbre. On note $Dom\,(x)$ l'ensemble des *dominateurs* de x, c'est-à-dire l'ensemble des nœuds dominant x. Le *dominateur immédiat* de x est noté $iDom\,(x)$.

Q 1.2 Dessinez l'arbre de dominance du programme donné ci-dessus.

Q 1.3 Un nœud se domine lui-même : $x \in Dom(x)$. Donnez une condition nécessaire pour que $x \in Dom(y)$ en fonction des prédécesseurs de y et leurs dominateurs respectifs. Transformez la condition en une équation de calcul de point fixe.

Q 1.4 Donnez un algorithme efficace pour le calcul des ensembles des dominateurs de chaque nœud. Soyez attentifs à son initialisation et prouvez sa correction.

Q 1.5 Faites tourner votre algorithme sur le programme donné.

Q 1.6 On dit que le nœud x domine *strictement* le nœud y ($x \in sDom(y)$) si tout chemin du nœud d'entrée *au début de* y traverse x. Exprimez $sDom(x)$ en fonction de $Dom(x)$.

1.2 Frontière de dominance

Soit x un nœud du graphe de flot de contrôle. La *frontière de dominance* de x, notée $DF(x)$ est l'ensemble des nœuds non strictement dominés par x mais ayant un prédécesseur dominé par x.

Q 1.7 Dessinez un graphe de flot de contrôle tel qu'il contienne les nœuds x, y et $z_1 \dots z_n$ et tel que $DF(x) = \{z_1, \dots, z_n\}$.

Q 1.8 x peut-il être dans $DF(x)$?

Q 1.9 Montrez que les nœuds de $DF(x)$ sont nécessairement des nœuds de confluence, c'est-à-dire ayant plusieurs prédécesseurs.

Q 1.10 Un nœud de $DF(x)$ n'a pas nécessairement seulement des prédécesseurs dominés par x. Donnez un exemple illustrant ce point.

Q 1.11 En revanche, un nœud de $DF(x)$ peut avoir plusieurs prédécesseurs dominés par x. Montrez aussi ce cas de figure dans un exemple.

Q 1.12 Un algorithme pour le calcul de la frontière de dominance est le suivant :

pour tout nœud x **faire**
  $DF(x) \leftarrow \emptyset$
pour tout nœud x **faire**
  si x **a plusieurs prédécesseurs alors**
    pour tout prédécesseur p **de** x
      $y \leftarrow p$
      tant que $p \neq iDom(x)$
        $DF(y) \leftarrow DF(y) \cup \{x\}$
        $y \leftarrow iDom(y)$

Démontrez la correction de cet algorithme.

1.3 Analyse de vivacité et d'interférence

On rappelle d'abord quelques définitions.

- Une assignation à une variable *définit* cette variable. On note $Def(x)$ l'ensemble des variables définies au nœud x.
- Une instruction dans laquelle une variable apparaît en partie droite *utilise* cette variable. On note $Use(x)$ l'ensemble des variables utilisées en un nœud x.
- Une variable est *vivante* (*live*) *sur une arête* (x, y) du graphe de flot de contrôle ssi il existe un chemin d'une définition à une utilisation de cette variable passant par l'arête (x, y).

- Une variable est *vivante à l'entrée* (*live-in*) d'un nœud x si elle est vivante sur une des arêtes entrant en x. Elle est *vivante en sortie* (*live-out*) d'un nœud x si elle est vivante sur une des arêtes sortant de x. Un note $LiveIn\,(x)$ et $LiveOut\,(x)$ les ensembles des variables vivantes respectivement en entrée ou en sortie d'un nœud x.
- La *durée de vie* (*live-range*) d'une variable est l'ensemble des arêtes du graphe de flot de contrôle où cette variable est en vie.
- Deux variables *interfèrent* si leurs durées de vie s'intersectent.
- Le *graphe d'interférence* des variables d'un programme est le graphe suivant : on associe à chaque variable un sommet du graphe et on met une arête entre deux sommets ssi les variables correspondantes interfèrent.

Q 1.13 Calculez (à la main) l'ensemble des variables vivantes pour chaque nœud du graphe de flot de contrôle correspondant au programme suivant :

```
a = 1;
b = 2;
c = 3;
b = c + b;
while (a < 15) {
  c = a + 1;
  b = c * c;
  a = b + 1;
}
return b;
```

Q 1.14 Dessinez le graphe d'interférence pour le programme donné ci-dessus.

Q 1.15 Inspirez-vous de l'algorithme de calcul des dominateurs pour donner un algorithme pour calculer la vivacité des variables d'un programme. Regardez en particulier son initialisation. Prouvez sa terminaison.

Q 1.16 Appliquez votre algorithme au programme de la première question.

Q 1.17 Où aura-t-on notamment besoin du graphe d'interférences ?