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

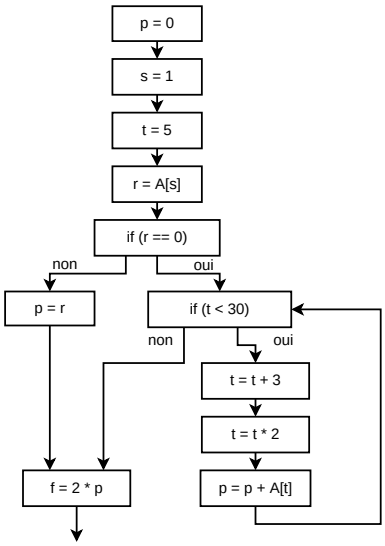
On rappelle ce qu'est le *graphe de flot de contrôle* (CFG) d'un programme :
- Chaque instruction est représentée par un *nœud* du graphe CFG.
- Si une instruction x peut être suivie d'une instruction y, il y a un arc de x vers y.
- Une chaîne de nœuds qui ont tous un seul successeur et un seul prédécesseur (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.

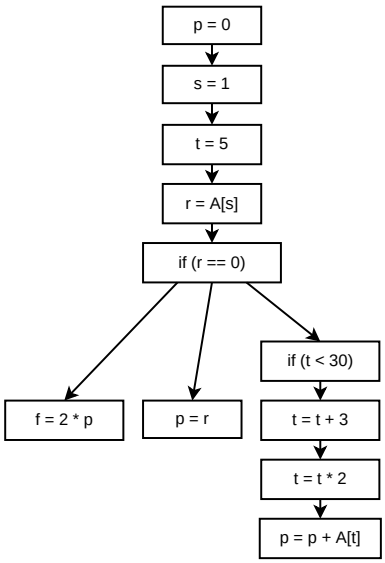
Solution:



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.

Solution:



On remarque que le graphe n'est pas un arbre couvrant du graphe CFG.

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.

Solution:

On a évidement que x domine y s'il domine tous ses prédecesseurs.
D'où l'équation de point fixe : $Dom\,(y) = \bigcap_{\forall z \in Pred(y)} Dom\,(z) \cup \{y\}$

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.

Solution:

Premier algorithme :

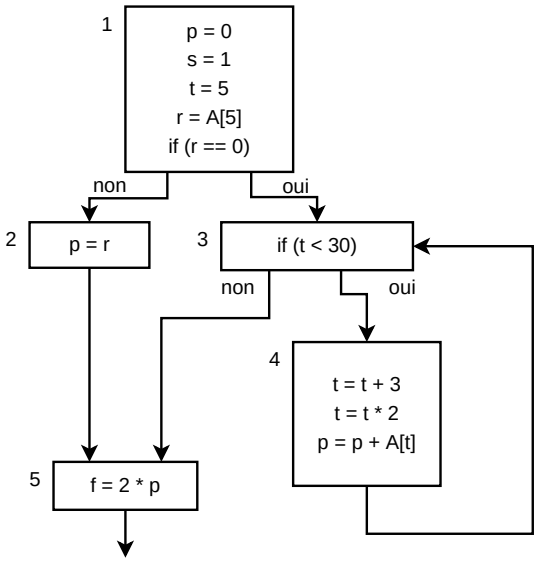
$Dom\,(entree) \leftarrow entree$
pour tout $x \in \mathcal{L}$, $x \neq entree$ **faire**
    $Dom\,(x) \leftarrow \mathcal{L}$
D' initialisé à $\perp$ sur toutes ses composantes
tant que $D' \neq Dom$ **faire**
    $D' \leftarrow Dom$
    pour tout $y \in \mathcal{L}$ **faire**
        $Dom\,(y) \leftarrow \bigcap_{\forall x \in Pred(y)} Dom\,(x) \cup \{y\}$

Second algorithme (plus efficace) :

$w \leftarrow \mathcal{L}$
pour tout $x \in \mathcal{L}$
    $Dom\,(x) \leftarrow \mathcal{L}$
tant que $w \neq \emptyset$
    $x \leftarrow$ **extraire** w
    $t \leftarrow \{x\} \cup \bigcap_{\forall y \in Pred(x)} Dom\,(y)$
    si $t \subsetneq Dom\,(x)$ **alors**
        $Dom\,(x) \leftarrow t$
        $w \leftarrow w \cup succ(x)$

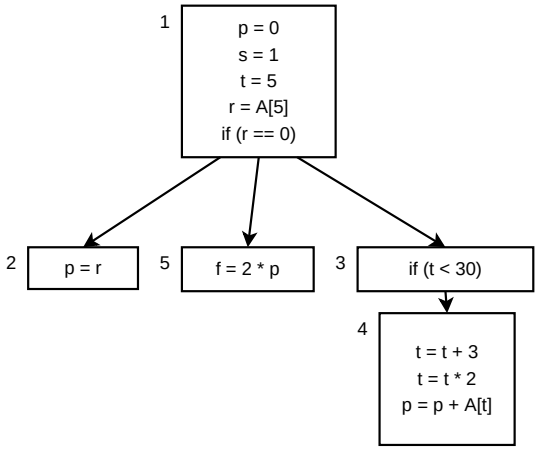
Q 1.5 Faites tourner votre algorithme sur le programme donné.

Solution: *Pour simplifier, on se contente de considérer les blocs de base comme suit :*



x	w (*x extrait*)	$Dom\,(1)$	$Dom\,(2)$	$Dom\,(3)$	$Dom\,(4)$	$Dom\,(5)$	*nouveau* w
-	$\{1,2,3,4,5\}$	$\{1,2,3,4,5\}$	$\{1,2,3,4,5\}$	$\{1,2,3,4,5\}$	$\{1,2,3,4,5\}$	$\{1,2,3,4,5\}$	-
1	$\{2,3,4,5\}$	$\{1\}$	-	-	-	-	$\{3,4,5\}$
2	$\{3,4,5\}$	-	$\{1,2\}$	-	-	-	$\{4,5\}$
3	$\{4,5\}$	-	-	$\{1,3\}$	-	-	$\{3,4,5\}$
3	$\{4,5\}$	-	-	$\{1,3\}$	-	-	-
4	$\{5\}$	-	-	-	$\{1,3,4\}$	-	$\{3,5\}$
3	$\{4,5\}$	-	-	$\{1,3\}$	-	-	-
5	$\emptyset$	-	-	-	-	-	$\emptyset$

« - » signifie qu'on est pas passé par une étape pouvant le modifier.
On obtient donc un graphe de dominance cohérent avec la question 1-2



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 $Dom\,(x)$.

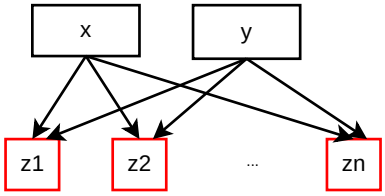
Solution:
On a clairement $sDom\,(x) = Dom\,(x) \setminus \{x\}$.

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.

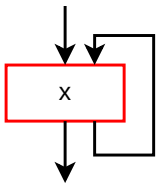
Q 1.7 Dessinez un graphe de flot de contrôle tel qu'il contienne les nœuds x, y et $z_1 \ldots z_n$ et tel que $DF\,(x) = \{z_1, \ldots, z_n\}$.

Solution:



Q 1.8 x peut-il être dans $DF(x)$?

Solution:



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

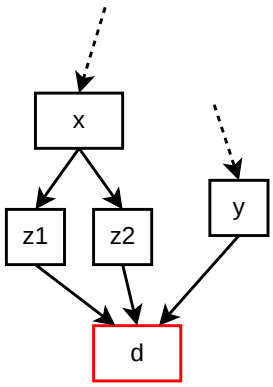
Solution:
Si $y \in DF(x)$ et n'a qu'un seul prédécesseur :
– *soit le prédécesseur est dominé par x et donc $y \in sDom(x)$ d'où $y \notin DF(x)$*
– *soit le prédécesseur n'est pas dominé par x d'où $y \notin DF(x)$*
Dans les deux cas on aboutit à une contradiction, donc si $y \in DF(x)$ alors il a plusieurs prédécesseurs, c'est à dire que les nœuds de $DF(x)$ sont nécessairement des nœuds de confluence.

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

Solution:
L'exemple de la question 1-7 convient aussi pour cette question, d'ailleurs mis à part x, si $y \in DF(x)$ il y a toujours un prédécesseur non dominé par x sinon y serait strictement dominé par x, et donc on aurait la contradiction $y \notin DF(x)$ par définition.

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.

Solution:



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.

Solution:
Grâce à la question 1-9, on sait que les nœuds qui n'ont qu'un seul prédécesseur ne sont dans aucune frontière de dominance, c'est pourquoi DF est à $\emptyset$.
Et un nœud x ayant plusieurs successeurs donnés, l'algorithme remonte et on ajoute les nœuds de l'arbre de dominance des successeurs de x jusqu'à tomber sur $iDom(x)$ car :
- *x ne peut être que dans la frontière de dominance d'un nœud qui domine ses prédécesseurs,*
- *tous les nœuds dominant les prédécesseurs de x dominées par $iDom(x)$ ne dominent pas strictement x par définition de $iDom(x)$,*
- *les autres sont soit $iDom(x)$ soit ses dominateurs, il dominent strictement x, donc x n'est pas dans leur frontières.*

1.3 Analyse de vivacité et d'interférence

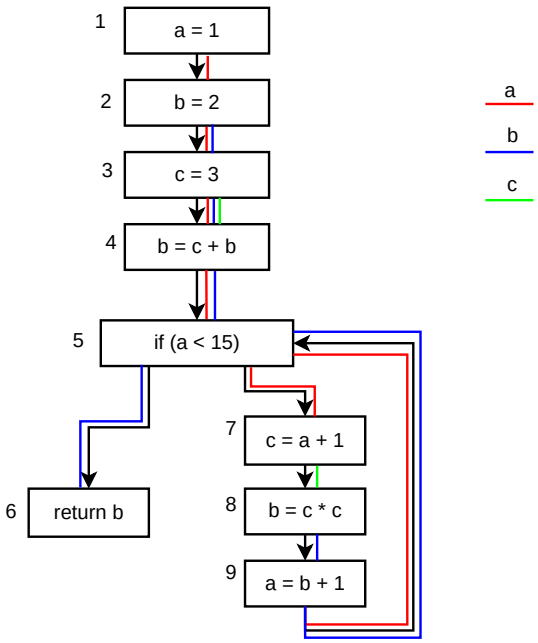
On rappelle d'abord quelques définitions.
- Une assignation à une variable *définit* cette variable. On note $Def(x)$ l'ensemble des variables définies au nœud x.
- Une instruction dans laquelle une variable apparaît en partie droite *utilise* cette variable. On note $Use(x)$ l'ensemble des variables 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) {
```

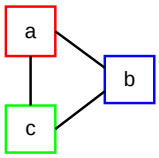
```
    c = a + 1;
    b = c * c;
    a = b + 1;
  }
  return b;
```

Solution:



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

Solution: *On voit immédiatement sur l'exemple précédent que les trois variables interfèrent deux à deux.*



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.

Solution:

Pour commencer on remarque :
$LiveIn\,(x) = (LiveOut\,(x) \setminus Def\,(x)) \cup Use\,(x)$
$LiveOut\,(x) = \bigcup_{y \in Succ(x)} LiveIn\,(y)$
On utilise donc un algorithme de point fixe inspiré de la seconde version de l'algorithme de calcul des dominateurs.

$w \leftarrow \mathcal{L}$
pour tout $x \in \mathcal{L}$
    $LiveOut\,(x) \leftarrow \emptyset$
    $LiveIn\,(x) \leftarrow Use\,(x)$
tant que $w \neq \emptyset$
    $x \leftarrow$ **extraire** w
    $t \leftarrow \bigcup_{y \in Succ(x)} LiveIn\,(y)$
    si $t \subsetneq LiveOut\,(x)$ **alors**
        $w \leftarrow w \cup pred(x)$
        $LiveIn\,(x) \leftarrow (LiveOut\,(x) \setminus Def\,(x)) \cup Use\,(x)$
        $LiveOut\,(x) \leftarrow t$

Concrètement on définit une variable comme vivante partout où elle est utilisée et on fait remonter jusqu'à sa définition.

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

Solution:
Pour bien comprendre l'algorithme, on pourra se restreindre à l'observation d'une seule variable, par exemple ici b.
b est utilisée en 3,6 et 9.
Ceci impose une vivacité LiveIn qui remonte au prédécesseur en tant que LiveOut. La vivacité LiveOut devient LiveIn et se propage de la même façon jusqu'à être arretée si la variable est justement définie au nœud où on arrive.
– *de 3 à 2*
– *de 6 à 4 et 8 en passant par 5, et 9*
– *de 9 à 8*
En ce qui concerne la vivacité de b dans les cas tels que l'arête entre 8 et 9, qu'on traite les prédécesseurs de 6 avant ceux de 9 ou inversement, on ne fera remonter la modification qu'une fois.

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

Solution: *On aura notamment besoin du graphe d'interférences pour l'allocation de registre (qu'il faut voir comme une coloration de graphe).*