Calcul de dépendances dans les boucles (2)
Extension du modèle
Analyse “floue”

F. Thomasset
INRIA - Rocquencourt
Projet A3

http://www-rocq.inria.fr/~thomasse/dea/

http://www-rocq.inria.fr/~thomasse/dea/articles.html

12 décembre 2001

Comment étendre le modèle linéaire ?

Pour un programme général, le système de dépendances est formé d'inéquations dans les entiers : indécidable !

```
a(x ** n) = ...
... = a(y ** n + z ** n)
```

Arithmétique de Presburger décidable

On ne sait pas résoudre les systèmes de dépendances dans tous les cas possibles :
Approximations (conservatives) nécessaires

Une extension au langage : boucles while

Exemples de non linéarité

Exemple 1 :

```
    DO X1 = 1, N
S1:    s = ...
       DO X2 = 1 WHILE (P(X1,X2))
S2:       s = ... s ...
       END DO
R:     ... = ... s ...
    END DO
```

Exemple 2 :

```
    DO X = 1, N
       IF (P(X)) THEN
S1:       s = ...
       ELSE
S2:       s = ...
       ENDIF
    END DO
R:  IF (N>0) THEN ... = ... s ...
```

Exemple 3 :

```
S1:  ii = ...
     DO x = 1, n
S2:     A(ii) = ...
S3:     ii = ...
S4:     A(ii) = ...
     END DO
     DO x = 1, n
R:      ... = ... A(x)
     END DO
```

Exemple 4 :

```
     IF (P) THEN
S1:     jlow = 2, jup = jmax = - 1
     ELSE
S2:     jlow = 1, jup = jmax
     ENDIF
     DO x = 1, n
S3:     A(jlow: jup) = ...
        IF (P) THEN
R:        ... = A(2: jmax - 1)
        ENDIF
     END DO
```

Analyse "floue"

(*"Fuzzy Array Data flow Analysis = FADA"*)

Hypothèses :

– structures de données : tableaux d'entiers, réels,...

– structures de contrôle :
boucles `DO, WHILE`, `IF..THEN..ELSE..`

– assignations à des éléments de tableau ou à des variables scalaires

– Pas de pointeur

– Pas d'alias

Exemple 1 : quelle est la source de $\langle$R,x$\rangle$?

```
    DO X1 = 1, N
S1:    s = ...
       DO X2 = 1 WHILE (P(X1,X2))
S2:       s = ... s ...
       END DO
R:     ... = ... s ...
    END DO
```

2 possibilités : S1 ou S2

Une difficulté : on ne connaît pas le nombre de tours de la boucle `while`

Donnons un nom à cette quantité :
$\alpha =$ nombre d'itérations de la boucle `while`

$Q_2^1(x) \;=\; \{\; \vec{z} \mid z_1 = x, 1 \leq z_2, z_2 = \alpha \;\}$

Max de cet ensemble : if $\alpha > 0$ then $\langle \mathtt{S2}, x, \alpha \rangle$ else $\perp$

Max des 2 sources : if $\alpha > 0$ then $\langle \mathtt{S2}, x, \alpha \rangle$ else $\langle \mathtt{S1}, x \rangle$

Pour calculer $Q^p_{s_i,t}(\vec{b})$
$= \{\ \vec{u}\ : \vec{u} \in L^p_{s_i,t}(\vec{b}), c_1(\vec{b},\vec{u}), \ldots\ \}$:

- Contraintes linéaires : $L^p_{s_i,t}(\vec{b})$

- Contraintes "non linéaires" : $c_1(\vec{b},\vec{u}), \ldots$

$$D_{s_i,t}(\vec{b}) = \{\ \vec{u}\ : c_1(\vec{b},\vec{u}), \ldots\ \}$$

$$Q^p_{s_i,t}(\vec{b}) = L^p_{s_i,t}(\vec{b}) \cap \{\ \vec{x}\ : \vec{x} \in D_{s_i,t}(\vec{b})\ \}$$

$$\hat{Q}^p_{s_i,t}(\vec{b},\vec{x}) = L^p_{s_i,t}(\vec{b}) \cap \{\ \vec{x}\ \}$$

$$\underset{\ll}{\mathrm{Max}}\ Q^p_{s_i,t}(\vec{b}) = \underset{\vec{x} \in D_{s_i,t}(\vec{b})}{\mathrm{Max}} \left(\underset{\ll}{\mathrm{Max}}\ \hat{Q}^p_{s_i,t}(\vec{b},\vec{x})\right)$$

$\underset{\ll}{\mathrm{Max}}\ \hat{Q}^p_{s_i,t}(\vec{b},\vec{x})$ est calculable par PIP
(avec $\vec{x}$ en paramètre)

Donnons un nom au vecteur qui réalise le maximum :
$\alpha^p_{s_i,t}(\vec{b}) = \text{Max } L^p_{s_i,t}(\vec{b}) \cap D_{s_i,t}(\vec{b})$

$\underset{\ll}{\text{Max}}\ \hat{Q}^p_{s_i,t}(\vec{b}, \alpha^p_{s_i,t}(\vec{b}))$: calculable par PIP
en fonction de $\alpha^p_{s_i,t}(\vec{b})$

Une approximation : donner à $\alpha^p_{s_i,t}(\vec{b})$ toutes les valeurs possibles : approximation très floue !

Exemple 1
dépendance de $\langle \mathtt{S2}, x_1, x_2 \rangle$ vers $\langle \mathtt{R}, y \rangle$
à la profondeur 1?

$$D_2(y) = \{ \; \vec{x} \;\; : \mathtt{P}(x_1, x_2) \; \}$$

$$\hat{Q}^1_2(\vec{x}, y) = \{ \; \vec{z} \;\; : \begin{array}{l} 1 \le z_1 \le n \\ 1 \le z_2 \\ z_1 = x_1 = y \\ z_2 = x_2 \end{array} \; \}$$

$$\underset{\ll}{\mathrm{Max}} \; \hat{Q}^1_2(\vec{x}, y) = \text{if } x_1 = y \begin{array}{l} \\ \text{then if } 1 \le x_2 \\ \qquad \text{then } \langle \mathtt{S2}, \vec{x} \rangle \\ \qquad \text{else } \perp \\ \text{else } \perp \end{array}$$

source à la profondeur p :
max $\{ \; \langle \mathtt{S2}, \underset{\ll}{\mathrm{Max}} \; \hat{Q}^1_2(\vec{x}, y) \rangle, \; \vec{x} \in D_2(y) \; \}$

$\vec{\alpha}(y) = [\alpha_1(y), \alpha_2(y)]$:
paramètre réalisant le maximum du max en $\vec{x}$

$\vec{\alpha}(y)$: paramètre réalisant le maximum en $\vec{x} \in D_2(y)$

source de $\langle \texttt{R,y} \rangle$:

$$
\begin{array}{ll}
\text{if } \alpha_1(y) = y & \text{then if } 1 \leq \alpha_2(y) \\
& \qquad \text{then } \langle \texttt{S2},\ \alpha_2(y) \rangle \\
& \qquad \text{else } \langle \texttt{S1}, y \rangle \\
& \text{else } \langle \texttt{S1}, y \rangle
\end{array}
$$

En général on ne sait pas calculer $\vec{\alpha}(y)$

Jusqu'ici aucune approximation n'a été faite.

En donnant à $\vec{\alpha}$ toutes les valeurs possibles :
source "floue" (*"fuzzy"*) :

$$
\begin{array}{l}
\{ \;\; \text{if } x_1 = y \;\wedge\; 1 \leq x_2 \text{ then } \langle \texttt{S2},\vec{x} \rangle \\
\qquad\qquad\qquad\qquad\quad \text{else } \langle \texttt{S1}, y \rangle \\
\quad \mid \vec{x} \in Z^2 \\
\}
\end{array}
$$

Pour améliorer la précision :
utiliser des propriétés sémantiques :

- Dans un `if..then..else`,
  une seule branche est exécutée
  (dans une itération donnée)

- Dans un `while`,
  si l'itération j est exécutée,
  les itérations précédentes ($< j$)
  sont aussi exécutées.

Exemple 2 :

```
    DO X = 1, N
       IF (P(X)) THEN
S1:       s = ...
       ELSE
S2:       s = ...
       ENDIF
    END DO
R:  IF (N>0) THEN ... = ... s ...
```

Source de R : $\{\ \langle$S1,N$\rangle$, $\langle$S2,N$\rangle\ \}$

1/ *Fuzzy Array Data Flow Analysis*
(Collard, Barthou, Feautrier 1995)
`www.prism.uvsq.fr/public/bad/ps/jpdc96.ps.gz`

Feautrier et al.
`www.prism.uvsq.fr/rapports/1998/document_1998_6.ps.gz`

Thèse de D. Barthou :
`www.prism.uvsq.fr/public/bad/ps/these.ps.gz`

2/ Application de ω-test :
Introduire des variables auxiliaires
pour représenter les termes non-linéaires
Pugh et Wonnacott 1994
`ftp://ftp.cs.umd.edu/pub/papers/papers/ncstrl.umcp/CS-TR-3372/CS-TR-3372.ps.Z`

```
DO I = 1, M
   A(2 * I + 1, TAB(I)) =
        A(2 * I, TAB(J)) + 1
END DO
```

$$\begin{cases} 1 \leq i, i' \leq M \\ 2i' + 1 = 2i \\ ti' = tj \text{ (représentent les termes non linéaires)} \end{cases}$$