Analyse de programme et analyse d'automate

Projet 2004

François Irigoin

École des Mines de Paris - Centre de Recherche en Informatique

20 septembre 2005

Les questions posées

- Pas d'élargissement ? Quel est l'algorithme utilisé dans PIPS ?
- Pourquoi marche-t-il en analyse de programmes scientifiques ?
- Marche-t-il aussi en analyse d'automate ?

Les questions posées

- Pas d'élargissement ? Quel est l'algorithme utilisé dans PIPS ?
- Pourquoi marche-t-il en analyse de programmes scientifiques ?
- Marche-t-il aussi en analyse d'automate ?

Les questions posées

- Pas d'élargissement ? Quel est l'algorithme utilisé dans PIPS ?
- Pourquoi marche-t-il en analyse de programmes scientifiques ?
- Marche-t-il aussi en analyse d'automate ?

Première partie
Algorithme utilisé dans PIPS

Analyse modulaire

- *transformers* comme abstraction des commandes et des procédures
- calculés de bas en haut
- *préconditions* comme abstraction des états
- calculées de haut en bas

Analyse modulaire

- *transformers* comme abstraction des commandes et des procédures
- calculés de bas en haut
- *préconditions* comme abstraction des états
- calculées de haut en bas

Analyse modulaire

- *transformers* comme abstraction des commandes et des procédures
- calculés de bas en haut
- *préconditions* comme abstraction des états
- calculées de haut en bas

Analyse modulaire

- *transformers* comme abstraction des commandes et des procédures
- calculés de bas en haut
- *préconditions* comme abstraction des états
- calculées de haut en bas

Graphe de contrôle hiérarchique

- les parties structurées sont conservées et analysées en tant que telles
- les parties non-structurées sont mise sous la forme de graphe
- un noeud de graphe peut correspondre à n'importe quelle commande complexe
- les parties structurées et non-structurées se contiennent récursivement
- on utilise une variante de l'algorithme de Bourdoncle pour n'avoir à analyser que des boucles WHILE

Algorithme : approximation de la fermeture transitive d'un *transformer*

1. $T(x^{k-1}, x^k)$
2. $T \wedge dx = x^k - x^{k-1}$
3. $T'(dx)$
4. $T'(dx) = \{dx \,|Adx = b \wedge A'dx \leq b'\}$
5. postcondition :

$$P^* = \{x \,|\exists k \in [0, \infty[\quad \begin{array}{rcl} Ax & = & Ax^0 + kb \\ \wedge A'x & \leq & A'x^0 + kb' \end{array}\}$$

Exemple de la voiture

```
s = 0, t = 0, d = 0
do while(s.le.2.and.t.le.3)
   read *, x
   if(x.gt.0.) then
      t = t + 1
      s = 0
   else
      d = d + 1
      s = s + 1
   endif
enddo
if(d.le.10) then
   print *, "healthy"
else
```

Exemple de la voiture (suite)

1. T(t,s)={s'=0, t'=t+1, d'=d, s<=2, t<=3}
2. T(d,s)={d'=d+1, s'=s+1, s'<=3, t=t'<=3}
3. T(d,s,t)={d'+t'=d+t+1,s+3t+1<=s'+3t'<=3t+3, t<=t',t'<=t+1,t<=4}
4. T'(dd,ds,dt){dd+dt=1, 1<=ds+3dt, 0<=dt<=1}
5. dd<=1ds+2dt

Deuxième partie
Extensions de l'algorithme

Assignations dans les boucles

```
read *, x
while(x.gt.0) {
  read *, x
  m = 10
}
```

Comportements périodiques

```
real x(0:1,0:9)
old = 0
new = 1
do i = 0, 9
  x(old,i) = f(i)
enddo
do t = 1, 1000
  do i = 0, 9
    x(new,i) = g(x(old,i))
  enddo
  old = new
  new = 1 - old
enddo
```

Utilisation de T^k

- $$\bar{T_k}^* = \bigsqcup_{i=0,k-1} T^i \circ \left(\bar{T}^k\right)^* \quad (1)$$
- $$\bar{T_k}^+ = \bigsqcup_{i=1,k} T^i \circ \left(\bar{T}^k\right)^* \quad (2)$$
- $$\bar{T}^* = \bigcap_{i\in[0,k[} \bar{T_k}^* \quad (3)$$
- non implanté

Comportements exponentiels

```
      n = 2**m
      nu = n
      ku = 2
      kn = ku + nu
      ln = 1
      do 110 j = 1, m
         do 100 i = 0, ln-1
            print *, i+kn
100      continue
         ku = ku + ln
         kn = ku + nu
         ln = 2 * ln
110   continue
```

Différences d'ordre supérieur et polynôme

- Jamais vu dans un code :

```
i = 0
j = 2
do k = 1, 10
  j = j - 1
  i = i + j
enddo
print *, i, j
```

Analyses de monotonicité

- tous les cas de la littérature ne sont pas traités
- les cas qui ne sont pas traités n'apparaissent pas dans les codes
- expériences à poursuivre

Troisième partie
Extensions de l'analyse des transformers

Adaptation de l'approche de bas en haut

```
do ix = 1, nx2
   ij = ix
   do iy = 1, ny2
         a(ij) = 0.
         ij = ij + nx2
   enddo
enddo
```

Recalcul des transformers pour les préconditions

- Les transformers sont recalculés pour propager les préconditions
- implanté

Recalcul des transformers avec les préconditions

- Les transformers sont recalculés avec les préconditions déjà calculées
- implanté

Calcul des transformers dans le contexte (intraprocédural)

- $T(S_1; S_2) = T(S_1) \circ T(S_2)$
- $T(S_1; S_2) = T(S_1) \circ T_{S_1}(S_2)$
- implanté

Quatrième partie
Extensions de l'état et des points de contrôle

Extensions de l'état

```
DO I = 1, IA(N)
   X(I) = 0.
ENDDO

M = IA(N)
DO I = 1, M
   X(I) = 0.
ENDDO
```

Résumé de la trace

Si `i` et `j` se trouvent toujours dans l'intervalle $[0, 255]$:

```
C  T(I,J,K) {I#init<=I+253K, I<=I#init+2K, 2K<=I, I+253K<=255,
C    J#init+252K<=J+252, J+3K<=J#init+3, 3<=J+3K, J<=252K+3}
      if(x.gt.0.) then
        k = 1
        i = 2
      else
        k = 0
        j = 3
      endif
```

Clonage et expansion de procédure

```
subroutine motor(a,c,n)

if(a.eq.1.and.c.eq.n) stop "error"
if(a.eq.2.and.c.eq.1) stop "error"

if(a.eq.1) then
   a = 0
   c = c + 1
endif
if(a.eq.2) then
   a = 0
   c = c - 1
endif
```

Cinquième partie
Extensions de l'abstraction

Transformer pseudo-affine

- j = i % 2
- k = i+1 % 2
- i == 2 * l + j
- i+1 == 2*m + k

Encodage de valeurs non-entières

- booléens
- chaînes de caractères
- flottants

Sixième partie
Questions de la communauté interprétation abstraite

Élargissement sur P

- P^0, $T(P^0)$, $T^2(P^0)$, $T^3(P^0)$,... ?

Élargissement sur T

- Que donne l'élargissement de T, T^2, T^3,... ?

Comportement non-affine (1)

```
X = 0
DO I = 0, N
  X = X + 1 + X*(X-1)*(X-2)*(X-3)...
ENDDO
```

Comportement non-affine (2)

```
i = 0
do while(x.gt.0.)
   read *, y
   if(y.gt.0.) then
      i = 0
   endif
   read *, x
enddo
```

Enveloppe convexe retardée

```
J = 0
DO I = 0, N
  IF(I.EQ.1) THEN
    J = J + 3
  ELSEIF(I.EQ.2) THEN
    J = J - 1
  ELSEIF(I.EQ.3) THEN
    J = 5
  ELSEIF(I.EQ.4) THEN
    ...
  ENDIF
ENDDO
```

Septième partie
Comparaison expérimentale

Cas de l'ascenseur (Boigelot & al.)

- impact du choix des constantes sur la convexité : ajouter des dimensions ?
- suppression des cycles identité ?
- faut-il utiliser l'algorithme de Bourdoncle ?
- la minimisation du nombre de WHILE génère davantage d'enveloppes convexes de transformers

Huitième partie
Conclusion

Conclusion

- algorithme développé expérimentalement (Béatrice Creusillet, Nga Nguyen)
- vitesse et précision pour l'analyse des codes scientifiques
- des extensions nombreuses, implantées ou pas
- de nouveaux challenges avec les codes d'automates

Planning du projet

- réunion de lancement : 28 octobre 2004
- 6 mois :
  - interfaces génériques, avec support pour exactitude et robustesse
  - définition des benchmarks
- 12 mois :
  - bibliothèques C pour les domaines existants avec la nouvelle interface
  - nouveaux algorithmes pour élargissement, itération, accélération et partitionnement dynamique
- 18 mois :
  - bibliothèques C pour les nouveaux domaines abstraits
  - version multidomaine de NBAC (polyèdre, octogone, intervalle)
  - nouvel outil interprocédural, INBAC

Planning du projet (suite)

- 24 mois :
  - intégration de la nouvelle API dans les outils existants pour certains domaines
  - nouveaux algorithmes (slicing, élargissement, accélération, partitionnement)
  - algorithmes adaptatifs : partitionnements dynamiques, partitionnement de traces, partitionnement du contrôle, listes de polyèdres
- 30 mois :
  - INBAC and NBAC avec produits cartésiens et réduits
  - PIPS avec un nouveau domaine abstrait et des algorithmes adaptatifs partitionnement de traces, partitionnement du contrôle, listes de polyèdres
- 36 mois :
  - résultats expérimentaux avec les nouveaux domaines et les nouvelles strategies de resolution