Calcul du flot des données en présence de contraintes non-affines

Denis Barthou

PR*i*SM

Université de Versailles - Saint Quentin

23 février 1998

Plan

- Parallélisation et dépendances
  Enjeux du calcul des dépendances de flot.
- Analyse exacte du flot des données
  Modèle. Méthode classique. Algorithme simplifié.
- Analyse approchée du flot des données
  Modèle. Description de l'approximation. Optimalité.
- Applications des analyses
  Vérification. Expansion mémoire.

Parallélisation et dépendances

Parallélisation d'un code séquentiel en deux étapes :

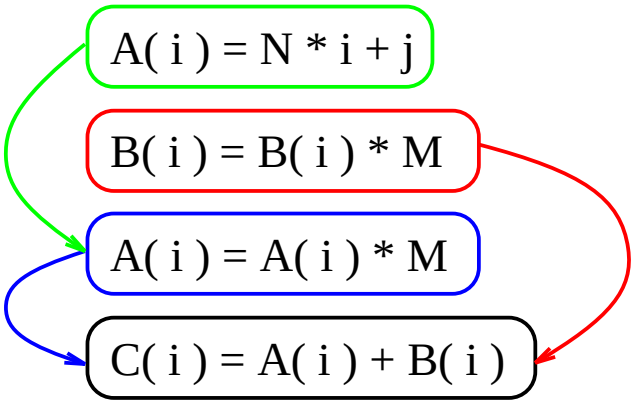
1. Trouver un sous-ordre de l'ordre séquentiel préservant la sémantique
   $\Rightarrow$ *Calcul des dépendances*

2. Exploiter le parallélisme entre opérations non comparables
   $\Rightarrow$ *Transformations de code*

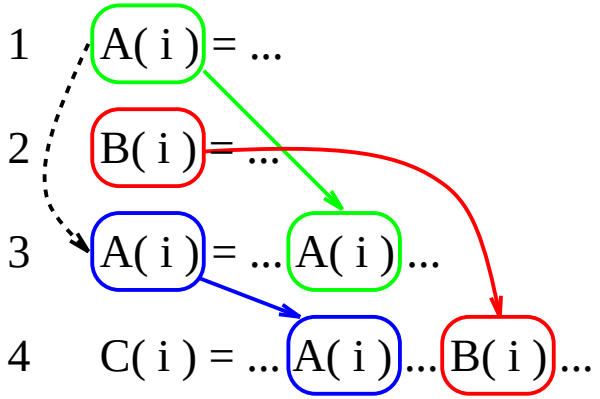
Plus les dépendances sont décrites finement,
plus les transformations peuvent être puissantes.

Le flot des données

- Caractéristique de l'algorithme
- Construit par analyse des utilisations/définitions



Flot des données



Dépendances de flot directes

Utilité du flot des données

- Expression du parallélisme par transformations de code :
  - expansion mémoire
  - ordonnancement
  - détection des récurrences
- Optimisation mémoire
- Débogage, vérification de code

Conditions de dépendance de flot directe

Entre 2 *opérations* (exécutions d'instructions) :

1. l'une lit, l'autre écrit la même cellule
2. les 2 opérations sont exécutées
3. l'écriture a lieu avant la lecture
4. c'est la dernière écriture en date, pour cette cellule

Analyse exacte du flot des données

Hypothèse : *programmes à contrôle statique*

- structures de données : tableaux, scalaires
- structures de contrôle :
  séquence, boucles `DO`, conditionnelles
  $\Rightarrow$ *Opérations indexées par les compte-tours*
- linéarité en fonction des compte-tours :
  des bornes, des indices des tableaux, des tests

Calcul exact du flot des données

Principe (Feautrier) : on considère une lecture

- Définition des opérations

  1. écrivant dans la même cellule
  2. étant exécutées
  3. précédant la lecture

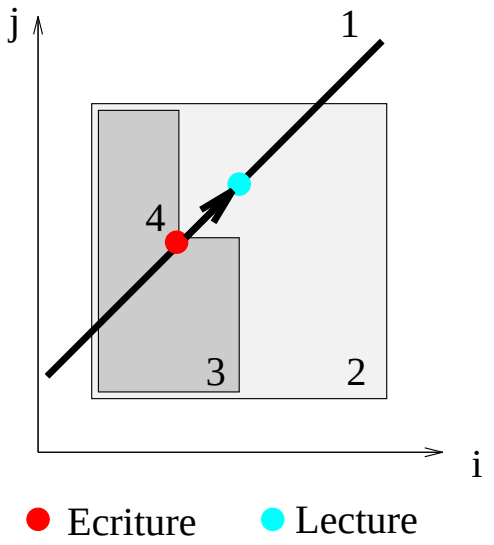
  $\Rightarrow$ *Ensemble défini par des contraintes affines*

- Calcul de la dernière opération d'écriture
  $\Rightarrow$ *Calcul d'un maximum*

Exemple de calcul

Quelle est la source d'une lecture de A(N+i-j) ?

```
DO  i = 1 , N
    DO  j = 1 , N
        A( N + i - j ) = ... A( N + i - j )
    ENDDO
ENDDO
```



Technique de calcul

Calcul du maximum par une méthode générale (simplexe entier)

- Coûteux
- Peu adapté aux contraintes usuelles :
  - prédominance de coefficients unitaires
  - système de contraintes peu dense

Algorithme simplifié de calcul

Calcul du maximum basé sur :

- l'élimination de Fourier-Motzkin (maximum)
- le lemme du reste chinois (résultat entier)

Complexité : au pire cubique en le nombre de boucles englobant l'écriture.

Domaine : sous-ensemble des programmes à contrôle statique

Performances de l'algorithme simplifié

Implémenté dans le projet PAF

- gain expérimental d'un facteur 5 à 40
- meilleure complexité que d'autres techniques polynômiales (Maydan *et al.*, Heckler et Thiele)
- meilleure complexité que des méthodes générales recourant à Fourier-Motzkin (Pugh, Maslov).

Extensions du modèle

Plus que les programmes à contrôle statique ?

Extensions :

- constantes symboliques comme coefficients de compte-tours (algorithme simplifié)
- certaines contraintes polynômiales (Maslov)
- certains appels de procédure (Leservot)

$\Rightarrow$ *Limites d'une analyse exacte.*

Analyse approchée du flot des données

Hypothèses :

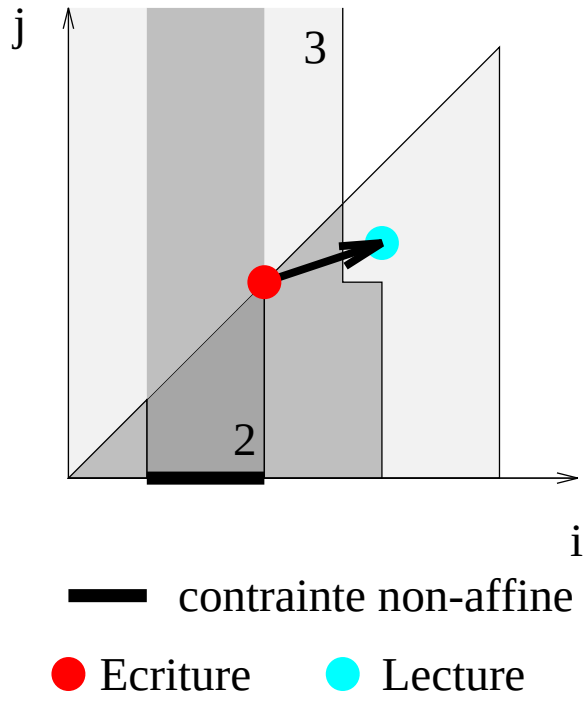
- structures de données : tableaux, scalaires
- structures de contrôle :
  séquence, boucles `DO` ou `WHILE`, conditionnelles
  $\Rightarrow$ *Opérations indexées par les compte-tours*

$\Rightarrow$ *Problème indécidable. Approximation nécessaire*

Exemple non-affine

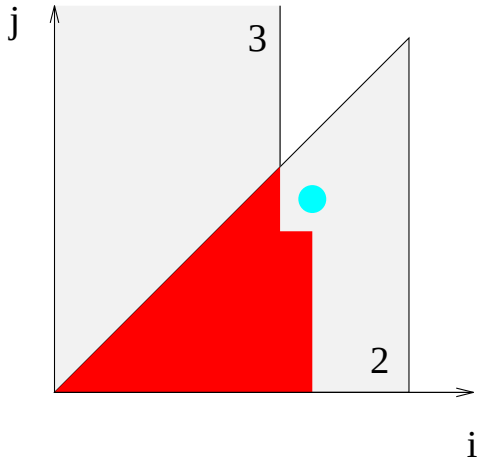
Un test de conditionnelle est non-affine :

```
DO  i = 1 , N
    DO  j = 1 , i
        ... = x
        IF   cos ( i * PI / 12 ) < .5   THEN
            x = ...
        ENDIF
    ENDDO
ENDDO
```



Quelle approximation ?

Première solution : ignorer les contraintes non-affines



Ensemble d'écritures possibles

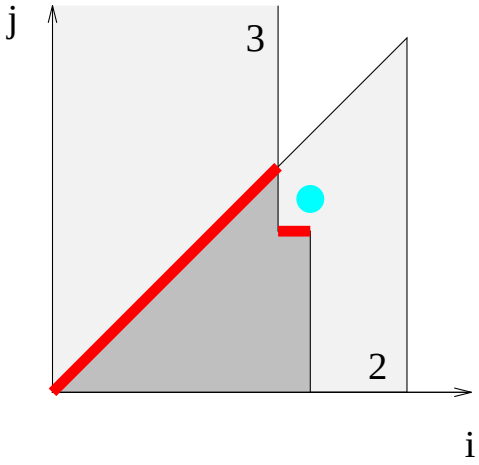
Lecture

⇒ Pas de calcul de maximum possible

Quelle approximation ?

Deuxième solution :
ensemble des solutions pour toutes les contraintes possibles.

$\cos(i\pi/12) \leq 0.5$ remplacé par f(i), f quelconque.



Ensemble d'écritures possibles

Lecture

$\Rightarrow$ *Description de l'ensemble à l'aide d'un paramètre*

Paramétrage

Principe :

1. substituer toutes les contraintes non-affines par une égalité entre compte-tours et un paramètre

2. calculer la dernière opération en fonction de ce paramètre
   $\Rightarrow$ *Solution paramétrée*
   $\Rightarrow$ *Réutilisation des techniques de calcul exact*

3. solution approchée :
   ensemble des opérations pour toutes valeurs possibles du paramètre

Amélioration de l'approximation

On a remplacé $\cos(i\pi/12) \leq 0.5$ par f(i), f quelconque.

Peut-on mieux caractériser f?

- f(i)=faux lorsque i=0
- f(i)=vrai lorsque $4 \leq$ i ≤ 8

$\Rightarrow$ *Analyses complémentaires sur les contraintes non-affines*

Analyses de contraintes non-affines

Méthodes existantes :

- propriétés d'une fonction non-affine (Maslov, Dumay)
- relations entre variables, entre contraintes. Basées sur
  - interprétation abstraite (Cousot et Halbwachs)
  - substitution/reconnaissance de motifs (Tu et Padua)
  - analyse de flot (Barthou, Collard et Feautrier)
    $\Rightarrow$ *Analyse itérative*

$\Rightarrow$ *Toutes ces propriétés s'écrivent comme des formules du* 1^{er} *ordre*

Analyse structurelle

Objectif : exprimer les propriétés des `while` et `if..then..else`

```
   DO  i = 1 , n
1     IF  P( i )  THEN  x = ...
2     ELSE  x = ...
      ENDIF
   ENDDO
   ... = x
```

Contraintes : $f_1(i) = P(i),\ f_2(i) = \neg P(i)$

Propriétés : $\forall i, f_1(i) \vee f_2(i)$ et $\forall i, \neg f_1(i) \vee \neg f_2(i)$

Analyse itérative

But : comparer deux contraintes non-affines

Principe : elles ont même valeur si

- les fonctions sont égales
- les variables utilisées ont même source, donc même valeur
  $\Rightarrow$ *Analyse de flot de données pour ces variables*

Exemple d'analyse itérative

Analyse du flot pour A(i) :

```
DO   i = 1 , N
1    A( B ( i ) ) = ...
2    B( i + 1 ) = ... A( i )
3    A( B( i + 1 ) ) = ...
ENDDO
```

Comparaison des contraintes B(i')=i et B(i"+1)=i :

- fonctions $x \rightarrow x = \text{i}$ et $x \rightarrow x = \text{i}$ égales
- B(i') et B(i"+1) ont même source si i'>1 et i"=i'-1

$\Rightarrow$ *Égalité des contraintes si* i"=i'-1 *et* i'>1.

Propriétés sur les paramètres

Objectif : trouver des propriétés sur les paramètres *impliquées* par les propriétés sur les contraintes non-affines

Méthode : méthode de résolution sur prédicats du 1er ordre
$\Rightarrow$ *Similaire à un démonstrateur de théorèmes*

Par exemple :

$$\begin{aligned} \forall i, 4 \leq i \leq 8 \quad &\Rightarrow \quad f(i) \\ \forall i, \big(1 \leq i \leq N\big) \wedge f(i) \quad &\Rightarrow \quad \big(i \leq \alpha\big) \end{aligned}$$

implique : $\forall i, 4 \leq i \leq \max(8, N) \Rightarrow i \leq \alpha$.

Optimalité de l'approximation

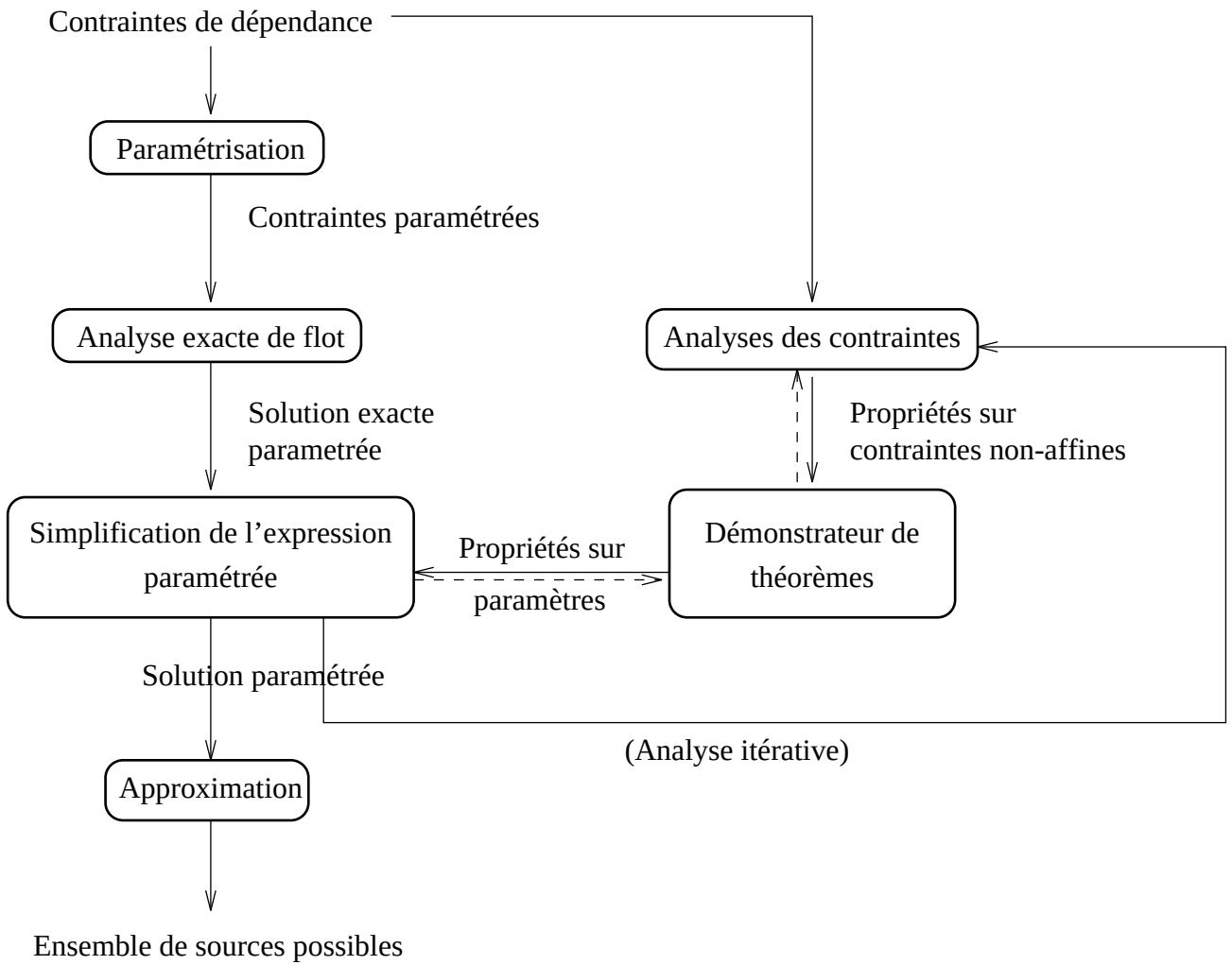
Plus petit ensemble possible par rapport aux propriétés sur les contraintes non-affines

Condition : déduire toutes les propriétés sur les paramètres
$\Rightarrow$ *Calcul non-fini*

Optimalité en temps fini :

- pour les relations affines entre variables (Cousot, Halbwachs)
- pour propriétés structurelles (`while` et `if..then..else`)

Schéma de l'analyse approchée



Implémentation

Implémentation en Objective Caml dans CARAVAN.

Prototype :

- analyseur syntaxique de Fortran
- analyse approchée du flot des données

Limitations actuelles :

- démonstrateur de théorèmes très simple
  $\Rightarrow$ basé sur une version précédente du formalisme
- seules les propriétés structurelles des `if..then..else` sont traitées

Analyses dans Caravan

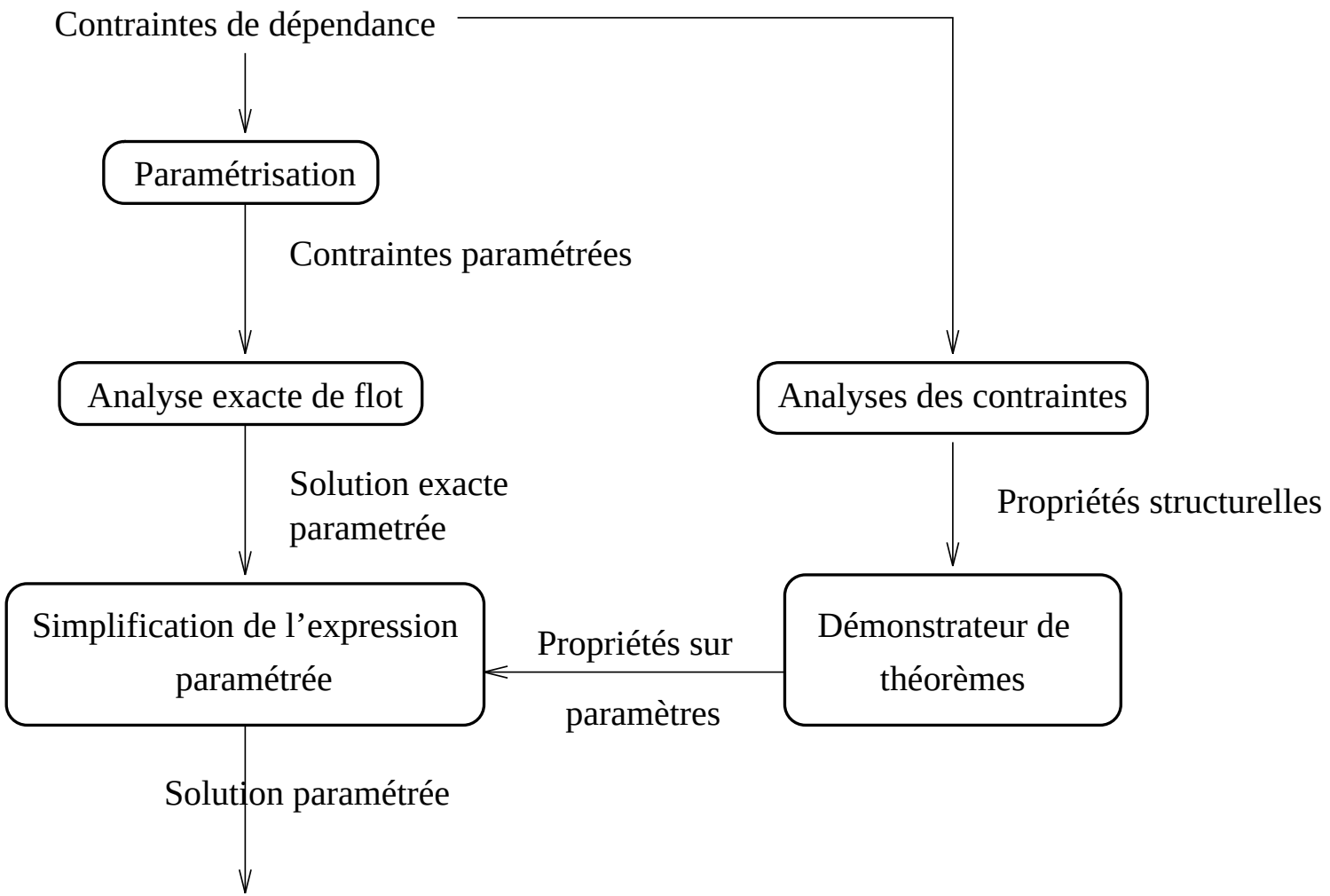
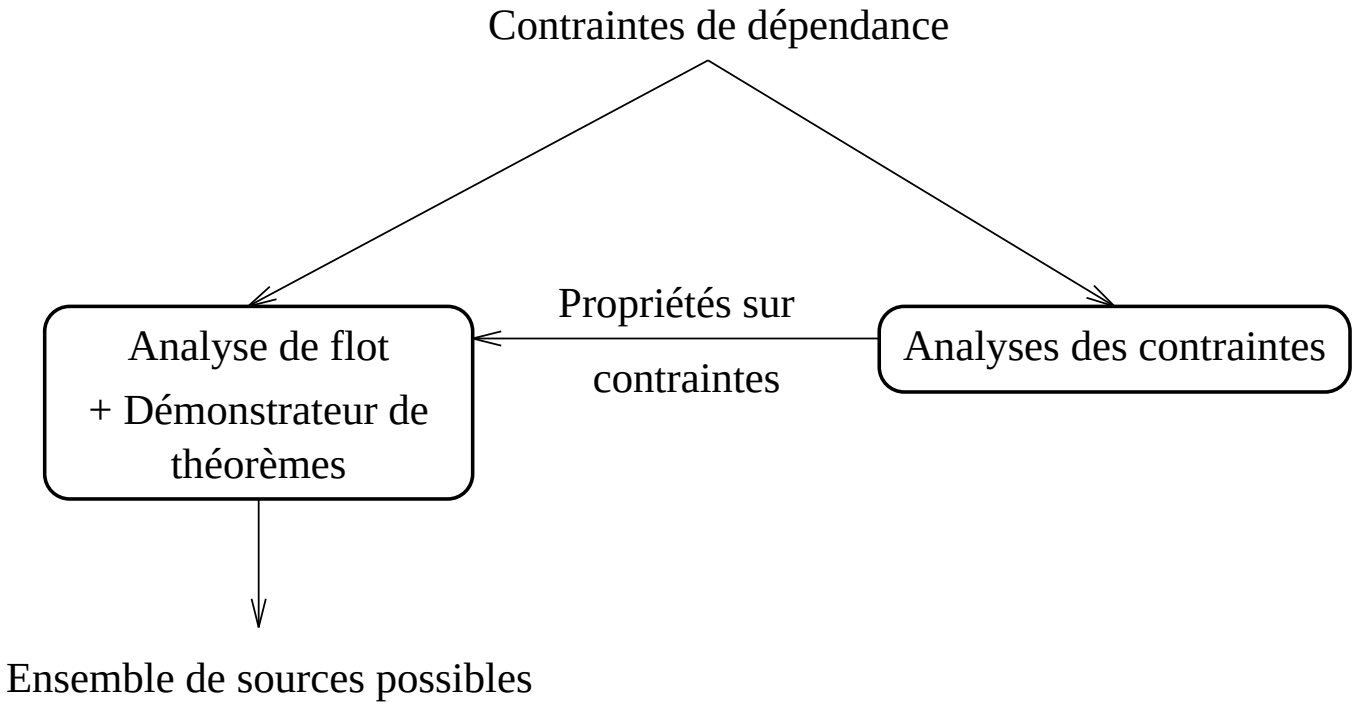


Schéma d'une analyse approchée sans paramètre



Approche choisie par Pugh et Wonnacott

⇒ Pas de solution paramétrée, pas de rétroaction possible

Applications de l'analyse approchée

Consiste à adapter les applications de l'analyse exacte.

Très simple pour :

- ordonnancement
- élimination de code mort
- transformations de boucles
- détection de récurrences

Vérification de code

Objectif : vérifier que toutes les variables sont définies avant d'être lues.

Principe : à partir de la solution paramétrée,

1. trouver les conditions sur les paramètres pour une non-initialisation
2. en déduire les conditions sur les contraintes non-affines
3. vérifier ces conditions (utilisateur, démonstrateur de théorèmes)

Vérification de code

À partir de l'exemple :

```
DO  i = 1 , N
   A( f( i ) ) = ...
ENDDO
DO  i = 1 , N
   ... = A( i )
ENDDO
```

on produit automatiquement la condition :

$$\forall i \text{ tq. } 1 \leq i \leq N, \exists i' \text{ tq. } 1 \leq i' \leq N, f(i') = i.$$

$\Rightarrow$ *f est une permutation*

Expansion de tableaux

Objectif : ne conserver que le flot des données

- Expansion limitée par la précision de l'analyse de flot
- Expansion totale et restauration dynamique du flot
  $\Rightarrow$ *Risque de surcoût, difficulté de mise en œuvre*

Principe de l'expansion statique maximale :

- utilisation du résultat approché de l'analyse de flot
- n'expanser un élément que pour des opérations dans des ensembles différents de sources possibles

Contributions

- Algorithme polynômial pour l'analyse exacte
- Approche générale pour traiter des contraintes non-affines
  - solution paramétrée : recule le moment de l'approximation
  - intègre les résultats de très nombreuses analyses
  - analyse itérative
  - méthode indépendante de la représentation
  - optimalité pour de nombreux cas courants
- Adaptation des applications :
  - Vérification : retrouve des propriétés sur les contraintes
  - Expansion statique maximale

Conclusion et perspectives

- Utiliser davantage la rétroaction entre solution paramétrée et propriétés sur contraintes non-affines
  - choisir des analyses complémentaires plus pointues
  - choisir la précision de l'analyse en fonction de l'endroit du programme
- Étendre le modèle aux appels de procédures
- Unifier l'approche sémantique et celle, plus syntaxique, des dépendances