Programmation Logique et Par Contraintes Avancee
Cours 6 { Propagateurs

Ralf Treinen

Université Paris Diderot
UFR Informatique
Laboratoire Preuves, Programmes et Systèmes
treinen@pps.univ-paris-diderot.fr

Rappel du cours precedent

- *domaine D* : associe à des variables des ensembles finis de

Propagateurs de bornes

- On parle de l'infimum et du suprémum du domaine d'une variable $(inf(x), sup(x))$
- Propagateur de bornes : ne fait que croître l'infimum et décroître le suprémum d'un domaine : transforme toujours un intervalle en un domaine qui est un intervalle, mais peut aussi être appliquée à un domaine qui n'est pas un intervalle.

Propagateurs de bornes

$$inf(X_i) := max\left(inf(X_i), \left\lceil \frac{b - \Sigma_{(1 \leq j \leq n; j \neq i)} a_j \, sup(X_j)}{a_i} \right\rceil\right)$$

$$sup(X_i) := min\left(sup(X_i), \left\lfloor \frac{b - \Sigma_{(1 \leq j \leq n; j \neq i)} \, a_j inf(X_j)}{a_i} \right\rfloor\right)$$

On obtient donc $2n$ propagateurs différents !

Propagateur de bornes pour des equations lineaires

Donnée une équation linéaire :

$$\Sigma_{i=1}^{n} a_i X_i = b$$

où les X_i sont des variables à domaine fini, et les a_i et b des constantes entières, le propagateur fait la chose suivante :

Un peu de theorie

- Donné un ensemble F de variables de domaine fini.
- Domaine : fonction $D : F \rightarrow 2^{\mathbb{N}}$.
- Soit $\mathcal{D}$ l'ensemble de tous les domaines.
- On a un ordre partiel sur $\mathcal{D}$:

$$D_1 \sqsubseteq D_2 \Leftrightarrow \forall x \in F : D_1(x) \subseteq D_2(x)$$

- Propagateur : Fonction $p : \mathcal{D} \rightarrow \mathcal{D}$ qui est *monotone* et *decroissant* (voir le transparent suivant).

Proprietes de propagateurs

Un propagateur doit être

- *decroissant* : envoie toujours un domaine vers un domaine plus fort (plus restrictif) : $\forall D \in \mathcal{D} : p(D) \sqsubseteq D$
- *monotone* : $\forall D_1, D_2 \in \mathcal{D}$: Si $D_1 \sqsubseteq D_2$ alors $p(D_1) \sqsubseteq p(D_2)$

Un propagateur n'est *pas necessairement*

- *idempotent* : $p(p(D)) = p(D)$
- *complet*

Correction d'un propagateur par rapport a une contrainte

Le propagateur p est correct par rapport à la contrainte c si pour tout $D \in \mathcal{D}$:

- Si $\alpha \models c$
- et $\forall x \in F : \alpha(x) \in D(x)$
- alors $\forall x \in F : \alpha(x) \in p(D)(x)$

Autrement dit : p ne perd pas de solutions à la contrainte c.

Completude d'un ensemble de propagateurs par rapport a une contrainte

L'ensemble P de propagateurs est complèt par rapport à la contrainte c si pour tout $D \in \mathcal{D}$:

- Si on a atteint un point fixe : $p(D) = D$ pour tout $p \in P$
- et α est une affectation admise par D : $\forall x \in F : \alpha(x) \in p(D)(x)$
- alors $\alpha \models c$

Autrement dit : on élimine toutes les non-solutions seulement par propagation.
(c'est très rarement le cas !)

Exemple de non-completude

```
declare X Y Z [X Y Z]:::1#10 {Browse [X Y Z]}

X*Y =: Z

Z=10

X>:1

Y>:1
```

Un algorithme naïve de propagation de contraintes

```
while ∃ propagateur p with p(D) ≠ D do
  D := p(D)
end
```

Remarques

- L'algorithme termine (quand F est fini), car tous les domaines sont finis et tous les propagateurs sont décroissants.
- L'algorithme est non-déterministe.
- Est-ce que le non-déterminisme donne lieu à des résultats non-déterministes ?
- L'algorithme est naïve car on cherche à chaque itération un propagateur dans l'ensemble de tous les propagateurs disponibles.

Independance du resultat de la strategie

- Peut importe la stratégie utilisée dans l'algorithme, à la fin on obtient un domaine D qui est
  - un *point fixe* D avec $p(D) = D$ pour tous les propagateurs.
  - plus petit que le domaine initiale $D \sqsubseteq D_0$
- Nous allons montrer :
  - Si l'algorithme donne D_i dans la i-ème itération
  - et si D est un point fixe plus petit que D_0
  - Alors $D \sqsubseteq D_i$.

Demonstration

Par induction sur i !

- $i = 0$: On a $D \sqsubseteq D_0$ par hypothèse
- $i \to i + 1$: Hypothèse : $D \sqsubseteq D_i$. À montrer : $D \sqsubseteq D_{i+1}$
  - $p_i(D_i) = D_{i+1}$, où p_i propagateur de la i-ème étape.
  - $p_i(D) \sqsubseteq p_i(D_i)$ car p_i est monotone
  - $D = p_i(D)$ car D est un point fixe.
  - Donc : $D \sqsubseteq D_{i+1}$!

Pour conclure

- Soient D_1 et D_2 obtenues par l'algorithme selon deux stratégies différentes.
- Donc D_1 est D_2 sont plus petits que D_0, et des points fixes.
- Donc $D_1 \sqsubseteq D_2$ et $D_2 \sqsubseteq D_1$.
- Donc : $D_1 = D_2$

```
declare
fun {MapColoring Data}
   Countries = {Map Data fun {$ C#_} C end} in % list of countries
   proc {$ Color}
      NbColors  = 4 % number of colors
   in
      Color = {FD.record color Countries 1#NbColors}
      {ForAll Data
       proc {$ A#Bs}
          {ForAll Bs proc {$ B} Color.A \=: Color.B end}
       end}
      {FD.distribute ff Color}
   end
end

{Browse {SearchOne {MapColoring Europe}}}
```

Exemple : colorer la carte de l'Europe

On suppose donné une carte sous forme d'une liste d'association qui associe à chaque pays la liste de ses voisins, comme

```
Europe = [ austria     # [italy switzerland germany]
           belgium     # [france netherlands germany luxembu
           france      # [spain luxemburg italy]
           germany     # [austria france luxemburg netherlan
           italy       # nil
           luxemburg   # nil
           netherlands # nil
           portugal    # nil
           spain       # [portugal]
           switzerland # [italy france germany austria] ]
```

Colorer cette carte avec 4 couleurs (1,2,3,4).