Programmation par contraintes

Ecole Jeunes Chercheurs en Programmation

Rencontre du 24 juin 2011

François Fages
Projet Contraintes
INRIA Rocquencourt

`http://contraintes.inria.fr`

Programmation par contraintes?

Calcul sur structures d'information partielle.

Modèle machine classique

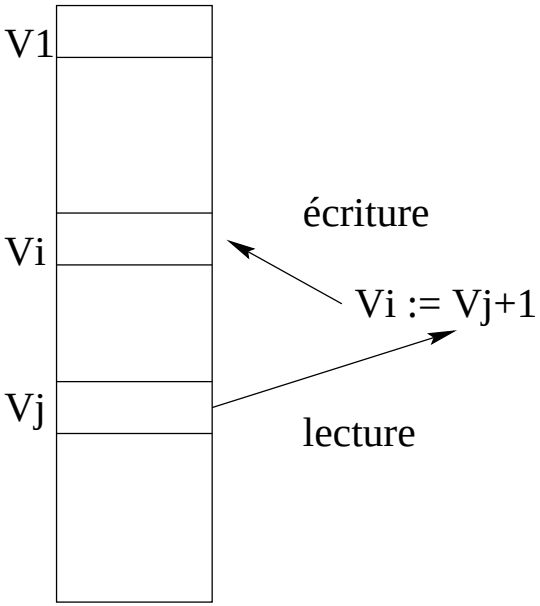
mémoire de valeurs

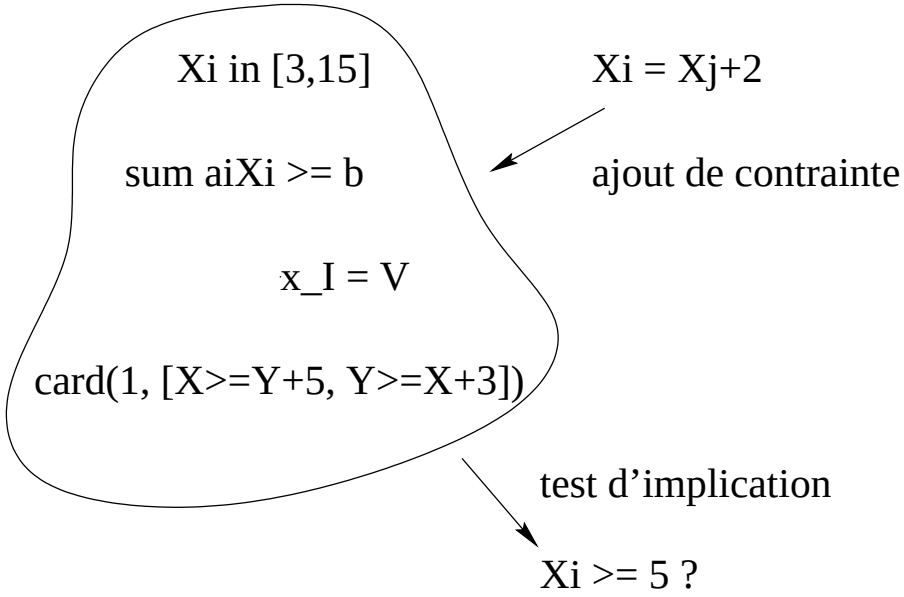
variables informatiques

Modèle machine de contraintes

mémoire de contraintes

variables mathématiques





Paradigme de la Programmation Logique (avec Contraintes)

Programme = Formule logique

Axiomatisation:

"Domaine du discours" $\mathcal{X}$,

Modélisation du problème P.

Exécution = Recherche de preuves

Satisfiabilité des contraintes,

Principe de résolution logique.

Classe de langages PLC($\mathcal{X}$) paramétrée par $\mathcal{X}$:

- Contraintes primitives sur $\mathcal{X}$

$$U = R * I$$

- Relations définies par des formules

$$\forall x, y \; path(x, y) \Leftrightarrow edge(x, y) \vee \exists z(edge(x, z) \wedge path(z, y))$$

Langages de définition de nouvelles relations

- Calcul des prédicats du premier ordre

$$\forall x, y\ path(x, y) \Leftrightarrow edge(x, y) \vee \exists z(edge(x, z) \wedge path(z, y))$$

- Clauses Prolog-PLC($\mathcal{X}$)

```
path(X,Y):- edge(X,Y).
path(X,Y):- edge(X,Z), path(Z,Y).
```

- Langages concurrents avec contraintes CC($\mathcal{X}$)

  Processus $A = c\ |p(x) \mid (A \parallel A) \mid A + A \mid ask(c) \rightarrow A \mid \exists x A$

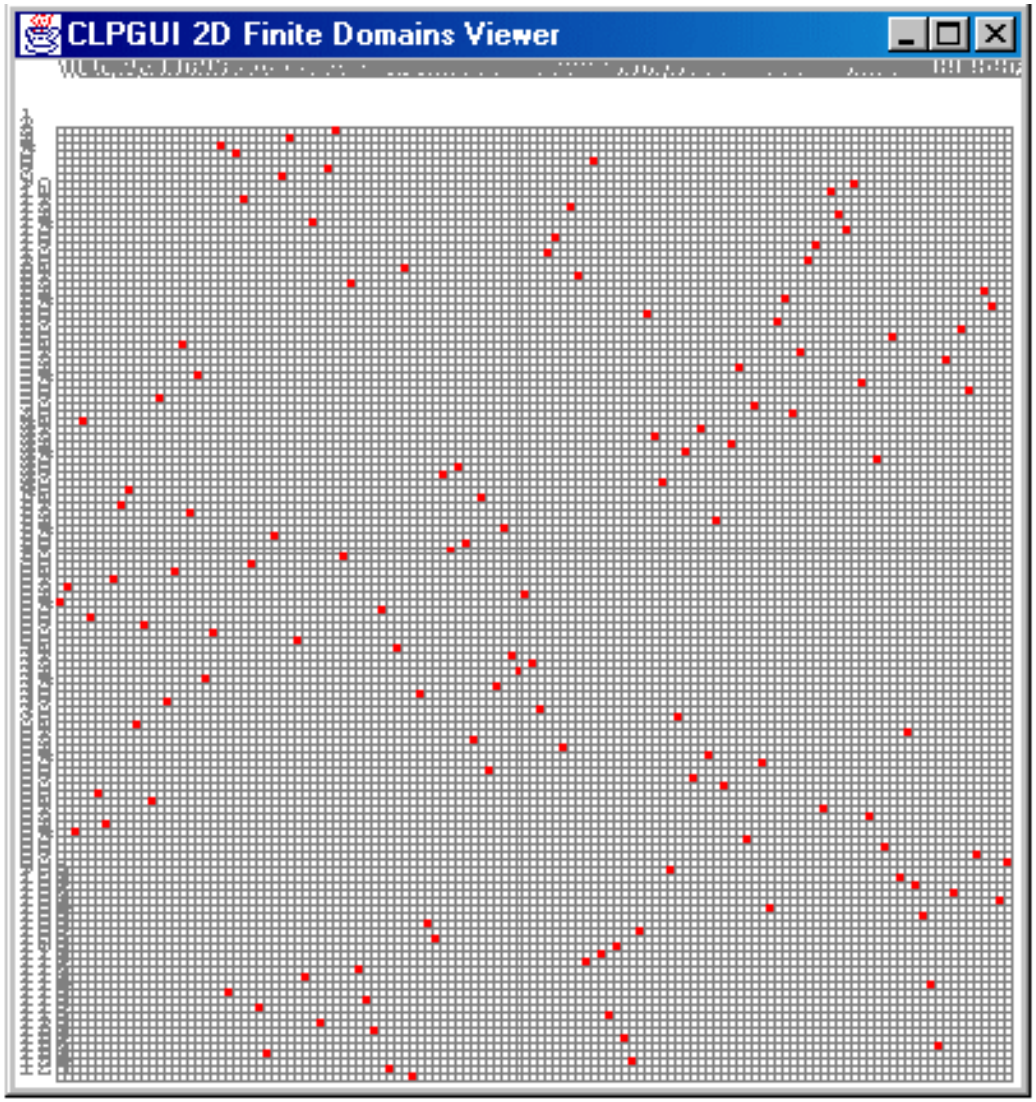
$$path(X, Y) :: edge(X, Y) + \exists Z(edge(X, Z)||path(Z, Y))$$

- Bibliothèques de contraintes dans langages à objets, langages fonctionnels, langages impératifs, ...

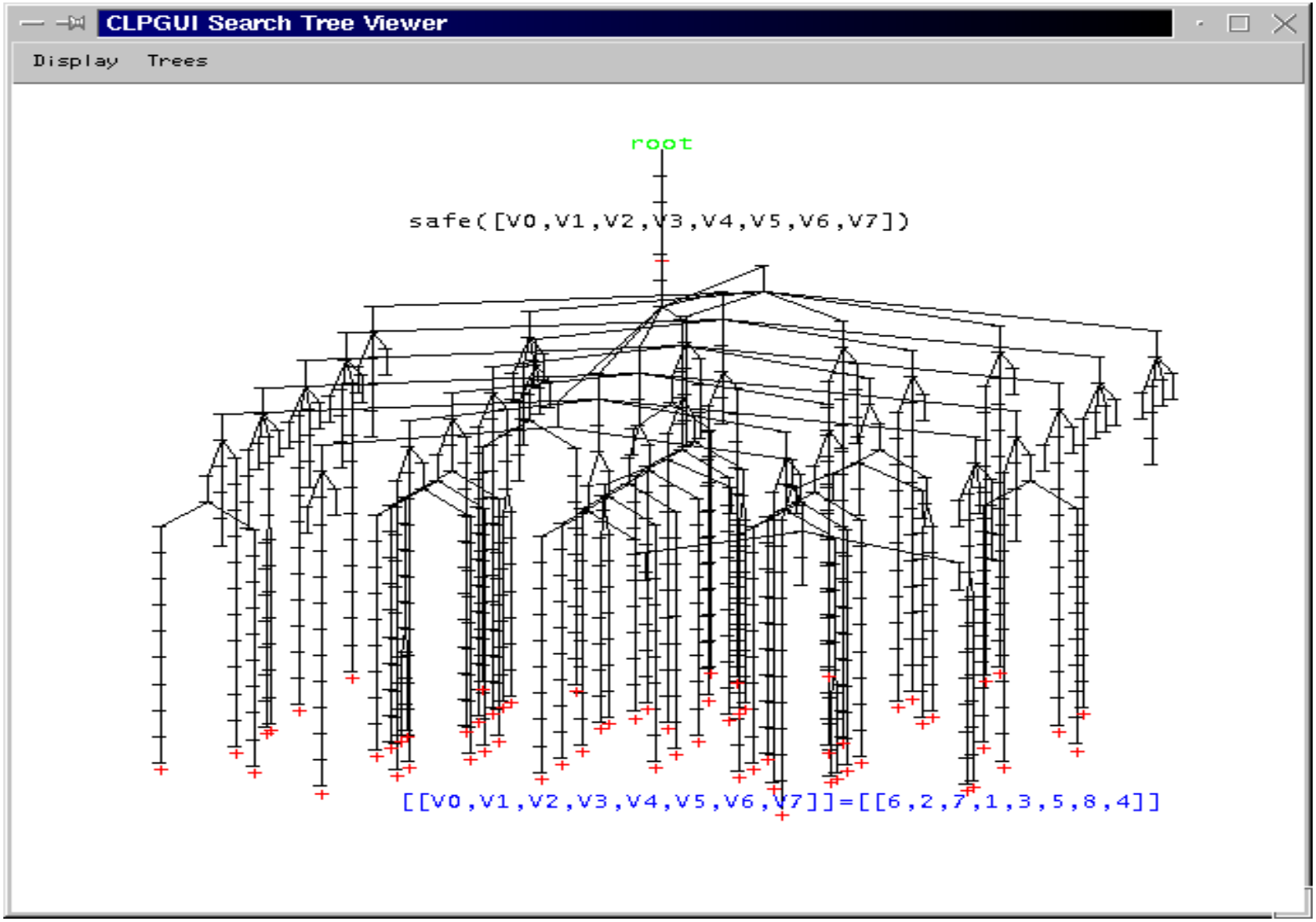
CLP(FD) N-queens Problem

GNU-Prolog program:

```
queens(N,L):-
    length(L,N),
    fd_domain(L,1,N),
    safe(L),
    fd_labeling(L,first_fail).
safe([]).
safe([X|L]):-
    noattack(L,X,1),
    safe(L).
noattack([],_,_).
noattack([Y|L],X,I):-
    X#\=Y,
    X#\=Y+I,
    X+I#\=Y,
    I1 is I+1,
    noattack(L,X,I1).
```



Search space of all solutions



Succés en résolution de problèmes combinatoires

Coloriage de graphes, allocation de ressources, ordonnancement de tâches...

- Problèmes de décision: existence d'une solution (de coût donné)
  dans classe P: si algorithme de complexité en temps polynomial
  dans NP: algorithme *non-déterministe* de complexité polynomiale.
  NP-complet si encodage polynomial de tout probleme dans NP.
- Problèmes d'optimisation: calcul d'une solution de coût optimal.
  NP-difficile si problème de décision associé NP-complet.
- Complexité indépendante de la taille de l'espace de recherche:
  Tri de n elements en $O(n \log n)$, espace de recherche en $!n$...
  SAT sur n variables booléennes en $O(2^n)$, espace de recherche en 2^n.
- Comparaison PPC / Programmation Linéaire, RO:
  + contraintes variées, + combinaison de techniques de résolution,
  + programmation de stratégies complexes, − difficulté d'utilisation.

Plan du Cours

1. Introduction, démonstrations GNU-Prolog+CLPGUI
2. Rappels de Logique
   théories complètes et décidabilité des langages de contraintes
3. Algorithmes de résolution de contraintes sur $\mathcal{H}$, sur $\mathcal{R}$,
   algorithme de propagation de contraintes sur FD,
4. Programmes Logiques avec Contraintes
   sémantique opérationnelle et exemples $PLC(\mathcal{H}, B, FD, \mathcal{R})$
5. Sémantiques de Point Fixe et Logique
   interprétation abstraite, model checking avec contraintes
6. Langages concurrents avec contraintes
7. Conclusion: panorama d'outils existants, sujets de recherche d'actualité.