LA PROGRAMMATION LOGIQUE PAR CONTRAINTES
MOTIVATIONS

De la PL vers la PLC

1) Introduction de la PLC : motivations, exemples

2) Principes de la programmation par contraintes (définition, résolution)

3) La programmation logique avec contraintes (méta-interprète et machine abstraites)

4) Limites

LIMITES DE LA PROGRAMMATION EN LOGIQUE

`?- 1 + X = 3` ⇒ FAIL

Deux solutions en Prolog :

- **Utilisation des axiomes de Peano**

```
plus(0,Y,Y).
plus(s(X),Y,s(Z)) :- plus(X,Y,Z).
```

et ?- 1 + X = 3 **devient** ?- **plus(s(0),Y,s(s(s(0))))**.
>Y = s(s(0))

- **Retardement de l'évaluation**

```
plus(X,Y,Z) :-
        freeze(X,freeze(Y,Z is X+Y)), freeze(Y,freeze(Z,X is Z-Y)), freeze(X,freeze(X,Y is X-Z)).
```

```
?- sort(X1,X2,X3) :-  X1 ≥ X2, X2 ≥ X3, ...
```

Ne permet pas de définir une relation d'ordre sur des variables

`?- sort(X1,X2,X3).` ⇒ FAIL

APPROCHE DE LA PROGRAMMATION PAR CONTRAINTES

N-REINES

COMMENT ANTICIPER LES IMPASSES ?

SEND + MORE = MONEY

$$\begin{array}{cccccc} & S & E & N & D \\ + & M & O & R & E \\ \hline = & M & O & N & E & Y \end{array}$$

COMMENT TROUVER L'INFORMATION PERTINENTE

- KNIGHT TOUR (CHEMIN HAMILTONIEN): **COMMENT TOUVER LES BONNES HEURISTIQUES ?**

PRINCIPES DE LA PROGRAMMATION AVEC CONTRAINTES

Contrainte

Une ***contrainte*** est une relation logique (une propriété qui doit être vérifiée) entre différentes inconnues. Elle restreint les valeurs que peuvent prendre simultanément les variables.

Exemple : "*x + 3*y = 12*" restreint les valeurs que l'on peut affecter simultanément aux variables *x* et *y*.

Les ***inconnues***, appelées variables, prennent leurs valeurs dans un ensemble donné, appelé ***domaine***.

Une contrainte est ***relationnelle, déclarative*** et peut être définie en ***extension*** ou en ***intension*** :

- Énumération des tuples de valeurs de la relation : "*(x=0 et y=1) ou (x=0 et y=2) ou (x=1 et y=2)*"
- Définition des propriétés mathématiques connues: "*x < y*" ou encore "*A et B => non(C)*"

Types de contraintes :

- Les contraintes numériques, entières ou réelles (linéaires ou non linéaires)
- Les contraintes ensemblistes, contraintes booléennes, univers de Herbrandt

PRINCIPES DE LA PROGRAMMATION AVEC CONTRAINTES
(suite)

Définition d'un CSP

Un CSP (Problème de Satisfaction de Contraintes) est un triplet *(X,D,C)* tel que :

- ***X*** *= { X1, X2, ..., Xn}* est l'ensemble des variables (les inconnues) du problème ;
- ***D*** : fonction qui associe à chaque variable *Xi* son domaine *D(Xi)*, l'ensemble des valeurs que peut prendre *Xi*
- ***C*** *= {C1, C2, ..., Ck}* est l'ensemble des contraintes.

Exemple:

- $X = \{a,b,c,d\}$
- $D(a) = D(b) = D(c) = D(d) = \{0,1\}$
- $C = \{ a \neq b, c \neq d, a+c < b \}$

Solution d'un CSP : ***affectation*** des variables, de telle sorte que toutes les contraintes soient satisfaites

Notion de CSP sur-contraint ou sous-contraint

- Un CSP qui n'a pas de solution est *sur-contraint* → *max-CSP* (préférences entre les contraintes)
- Un CSP qui admet beaucoup de solutions différentes est *sous-contraint*

PRINCIPES DE LA PROGRAMMATION AVEC CONTRAINTES
(suite)

Modélisation sous la forme d'un CSP

- identifier l'ensemble des variables *X* (les inconnues du problème) et leurs domaines
- identifier les contraintes *C* entre les variables.

On ne se soucie pas de savoir comment résoudre le problème : on cherche simplement à le spécifier formellement

PRINCIPES DE LA PROGRAMMATION AVEC CONTRAINTES
(suite)

Exemple : modélisation des n-reines

Les *inconnues* : les positions des reines sur l'échiquier

En numérotant les lignes et les colonnes de l'échiquier :

	1	2	3	4
1				
2				
3				
4				

-> association à chaque reine *i* de deux variables *Li* et *Ci* correspondant respectivement à la ligne et la colonne sur laquelle placer la reine

….. ou une variable Xi par colonne dont la valeur sera la position de la reine (numéro de ligne)

PRINCIPES DE LA PROGRAMMATION AVEC CONTRAINTES
(suite)

Exemple : modélisation des n-reines (suite)

Les *contraintes* :

- les reines doivent être sur des lignes différentes *:*

  {Xi ≠Xj pour i dans {1,2,3,4}

- les reines doivent être sur des colonnes différentes :
  aucune contrainte avec la deuxième modélisation !

- les reines doivent être sur des diagonales différentes :

  {Xi ≠Xj +/- (i-j) pour i,j dans {1,2,3,4} et i≠j

TECHNIQUES DE RESOLUTION POUR LA PROGRAMMATION AVEC CONTRAINTES

L'algorithme "génère et teste" … prolog !

Principe : énumérer toutes les affectations totales possibles jusqu'à en trouver une qui satisfasse toutes les contraintes

L'algorithme "simple retour-arrière" : backtrack chronologique

Principe : pour améliorer l'algorithme "génère et teste" on va tester au fur et à mesure de la construction de l'affectation partielle sa consistance : dès lors qu'une affectation partielle est inconsistante, on "backtrack" jusqu'à la plus récente instanciation partielle consistante que l'on peut étendre en affectant une autre valeur à la dernière variable affectée

Filtrage par consistance locale et utilisation d'heuristiques

Principes :

- ***Élimination a priori*** des valeurs qui ne peuvent être dans aucune solution (travail local à chaque contrainte)
- ***Choix* des variables et valeurs** (lors de l'énumération) qui devraient permettre de découvrir rapidement les ***impasses***

Backtracking vs Generate & Test : exemple

Contraintes: X = Y, X 6= Z, Y > Z

Domaines : $D_X = D_y = D_z = \{1, 2\}$,

Generate & Test:

X	Y	Z	Test
1	1	1	fail
1	1	2	fail
1	2	1	fail
1	2	2	fail
2	1	1	fail
1	1	1	fail
2	1	2	fail
2	2	1	succes

Backtracking:

X	Y	Z	Test
1	1	1	fail
		2	fail
	2		fail
2	1		fail
	2	1	succes

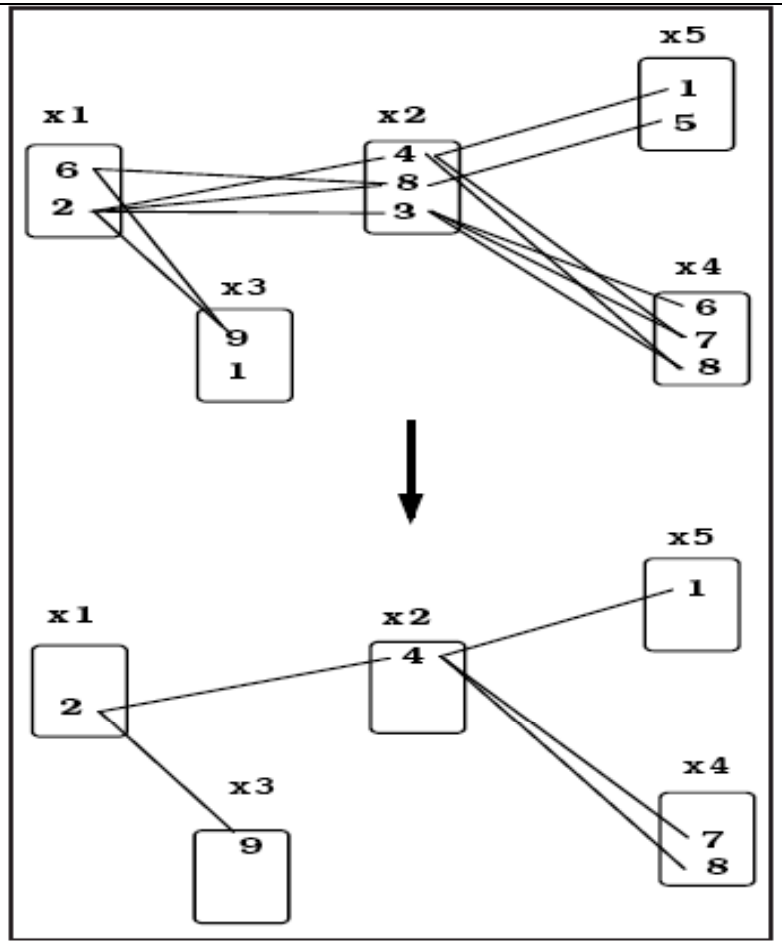
Filtrage par consistance d'arc : exemple

Contraintes :

x1 < x2, x5 + 3 = x2, x3 > x1, x4 > x2 + 3

Domaines :

Dx1 = {2, 6},
Dx2 = {3, 4, 8},
Dx3 = {1, 9},
Dx4 = {6, 7, 8},
Dx5 = {1, 5}



LA PROGRAMMATION LOGIQUE PAR CONTRAINTES : DEFINITION

→ THEORIE DU PREMIER ORDRE **qui préserve les PROPRIETES DE CALCUL associées aux clauses de Horn**

→ **Classes de** LP langages **dont les variables peuvent prendre des valeurs sur des** DOMAINES **variés**

EXEMPLE :

PROLOG

```
fib(0,1).
fib(1,1).
fib(N,R):-
        N ≥2,
        N1 is N-1, fib(N1,R1),
        N2 is N-2, fib(N2,R2),
        R is R1+R2.
?-fib(10,X)   ⇒  89
?-fib(X,89)   ⇒  ERREUR
```

CLP

```
fib(0,1).
fib(1,1).
fib(N,R1+R2):-
        N ≥2,
        fib(N-1,R1),
        fib(N-2,R2).

?-fib(10,X)  ⇒  89
?-fib(X,89)  ⇒  X=10
```

LA PROGRAMMATION LOGIQUE PAR CONTRAINTES : META-INTERPRETE

PROLOG

```
solve([]).
solve([Goal|RestGoal]):-
        solve(Goal),
        solve(RestGoal).
solve(Goal):-
        clause(Goal,Body),
        solve(Body).
```

LA PROGRAMMATION LOGIQUE PAR CONTRAINTES : META-INTERPRETE (suite)

CLP (Noyau)

```
solve([],C,C).
solve([Goal|Rest_Goal],Previous_C,New_C):-
      solve(Goal,Previous_C,Temp_C),
      solve(Rest_Goal,Temp_C,New_C).
solve(Goal,Previous_C,New_C):-
      clause(Goal,Body,Current_C),
      merge_c(Previous_C,Current_C,Temp_C),
      solve(Body,Temp_C,New_C).
```

merge_c(Previous_C,Current_C,Temp_C) :

Si ***Previous_C & Current_C*** **n'a pas de solution** → Fail

Sinon → • simplification **de *Temp_C***

• instanciation **des variables suffisamment contraintes**

LA PROGRAMMATION LOGIQUE PAR CONTRAINTES : MACHINE ABSTRAITE

(1) $\sigma = (W, t_0 t_1 \ldots t_n, S)$ (2) $s_0 \rightarrow s_1 \ldots s_m,\ R\ (m \geq 0)$

(3) $\sigma' = (W, s_1 \ldots s_m t_1 \ldots t_n, S \cup R \cup (s_0 = t_0))$

(1) **Représente** l'état σ de la machine **à un instant donné**

- W : **liste des variables de la question initiale,**
- $t_0 t_1 \ldots t_n$: **liste de termes (buts) à effacer,**
- S : **liste des contraintes courantes (contraintes satisfiables)**

(2) Représente la règle d'inférence **(i.e. règle du programme utilisée pour changer d'état)**

- $s_1 \ldots s_m$: **une suite de termes**
- R : **les contraintes de la règle**

(3) **Représente le** nouvel état σ' **de la machine après application de la règle (2) si**

$S \cup R \cup (s_0 = t_0)$ **est** satisfiable

Solution **du système :** $\sigma_f = (W, \varepsilon, \text{Contraintes\_Finales})$

LA PROGRAMMATION LOGIQUE PAR CONTRAINTES : LIMITES

Exemple : ChatsOiseaux

? NOMBRE D'OISEAUX ET DE CHATS QUI TOTALISENT 12 TETES ET 34 PATTES

```
ChatsOiseaux(C,O,P,T) :- P #=4C+2O, T#=C+O.
```

> ChatsOiseaux(C,O,14,5) . → *{C = 2, O =3}*

> ChatsOiseaux(1,O,P,5) . → *{O = 4, P =12}*

> ChatsOiseaux(1,1,2,4) . → *fail*

> ChatsOiseaux(C,O,6,4) . → *{C = -1, O =5}*

```
ChatsOiseaux(C,O,P,T) :- P #=4C+2O, T#=C+O, C#>=0,O#>=0,P#>=0,T#>=0.
```

> ChatsOiseaux(C,O,6,4) . → *fail*

> ChatsOiseaux(C,O,7,3) . → *{C = 1/2, O =5/2}*

⇒Ajouter *enum(C), enum(O)*

LA PROGRAMMATION LOGIQUE PAR CONTRAINTES : LIMITES (suite)

Solution de :

$a^n + b^n = c^n$

avec a,b,c et n : **entiers**

... et **n > 2**