GNU Prolog

- `http://gnu-prolog.inria.fr/`
- Prolog avec solutionneur de contraintes sur un domaine fini
- Dans les contraintes toutes les variables sont des entiers entre 0 et `fd_max_integer`.
- Solutionneur basé sur arc-consistance, bornes-consistance
- Deux représentations : intervals, représentation “clairsemée” (Attention: marche uniquement jusqu'à `vector_max`)

Prédicats de base

- `fd_domain(?Vars, +Integer1, +Integer2)`
  définit le domaine d'une variable `Vars` ou d'une liste de variables d'être entre `Integer1` et `Integer2`.
- `fd_domain(?Vars,+ListeValeurs)`
  pareil avec une liste de valeurs
- `fd_all_different(?ListeVars)`
  Ce prédicat décrit la contrainte qui impose que toutes les variables de la liste `ListeVars` prennent des valeurs différentes.
- minimiser, maximiser: voir manuel

Contraintes arithmétiques

- Les contraintes arithmétiques s'écrivent en utilisant les fonctions habituelles et les prédicats suivant: `#=`, `#\=`, `#<`, `#>`, `#<`, `#=<`, `#>=`
- On peut aussi utiliser `#=#`, `#\=#`, `#<#`, `#>#`, `#<#`, `#=<#`, `#>=#`
- `#pred` indique qu'on utilise uniquement la bornes-consistances.
- `#pred#` indique qu'on utilise l'(hyper)-arc-consistance

Exemple

```
| ?- fd_domain(X,1,8), fd_domain(Y,2,7), X #= 2*Y.

X = _#3(4..8)
Y = _#25(2..4)

yes
| ?- fd_domain(X,1,8), fd_domain(Y,2,7), X #=# 2*Y.

X = _#3(4:6:8)
Y = _#25(2..4)

yes
```

Résoudre les contraintes

- `fd_labeling(?Vars)`
  Ce prédicat est utilisé pour rechercher des solutions de contraintes sur les variables `Vars` avec des options par défaut.

- `fd_labeling(?Vars, ?Options)`
  Ce prédicat est utilisé pour rechercher des solutions des contraintes sur les variables `Vars` avec une liste d'options `Options`.
  - par exemple: `[variable_method(most_constrained)]` a comme effet que la variable choisie pendant la résolution est celle qui est la plus contrainte.

- plus de détails dans le manuel.

Programme en GNU Prolog

- Un programme pour résoudre une contrainte s'écrit en trois parties:
  - définir les domaines des variables
  - décrire la contrainte
  - résoudre

Exemple

```
probleme([X,Y,Z]) :- fd_domain(X,0,5),
                     fd_domain([Y,Z],3,7),
                     X+Y #< 2*Z,
                     fd_labeling([X,Y,Z],[]).
```

Ensuite:

```
| ?- probleme(L).
L = [0,3,3] ? ;
L = [0,3,4] ? ;
L = [0,3,5] ? ;
etc.
```


Contraintes numériques linéaires sur les réelles

- Une expression linéaire est de la forme $a_1x_1 + a_2x_2 + \cdots a_nx_n$ où chaque a_i est une constante et chaque x_i une variable.

- Une équation linéaire est de la forme $e_1 = e_2$ où e_1 et e_2 sont des expressions linéaires.

- Une inéquation est de la forme $e_1 \leq e_2, e_1 < e_2, e_1 \geq e_2$ ou $e_1 > e_2$.

- Une contrainte linéaire est une conjonction d'équations et inéquations linéaires.

Cas spécial : équations linéaires

- Solutionneur de contraintes : élimination Gauss-Jordan classique
- Idée : Réécrire une contrainte successivement pour la mettre en forme résolue. À chaque étape on remplace la contrainte par une contrainte équivalente.
- Une conjonction d'équations linéaires est en forme résolue, si elle a la forme

$$x_1 = e_1 \ \wedge \ x_2 = e_2 \ \wedge \ \cdots \ \wedge \ x_n = e_n$$

où les variables $x_1, \ldots, x_n$ (les non-paramètres) sont distinctes et n'apparaissent pas dans les expression $e_1, \ldots, e_n$.

- Pour une contrainte en forme résolue, on peut choisir librement une valeur pour les variables paramètres et les valeurs des variables non paramètres en découlent.

Exemple

$$\begin{aligned} 1 + X &= 2Y + Z \ \wedge \\ Z - X &= 3 \ \wedge \\ X + Y &= 5 + Z \end{aligned}$$

On choisit la 1er équation, on la réécrit $X = 2Y + Z - 1$ et on remplace X :

$$\begin{aligned} X &= 2Y + Z - 1 \ \wedge \\ Z - 2Y - Z + 1 &= 3 \ \wedge \\ 2Y + Z - 1 + Y &= 5 + Z \end{aligned}$$

On choisit la deuxième équation et on obtient

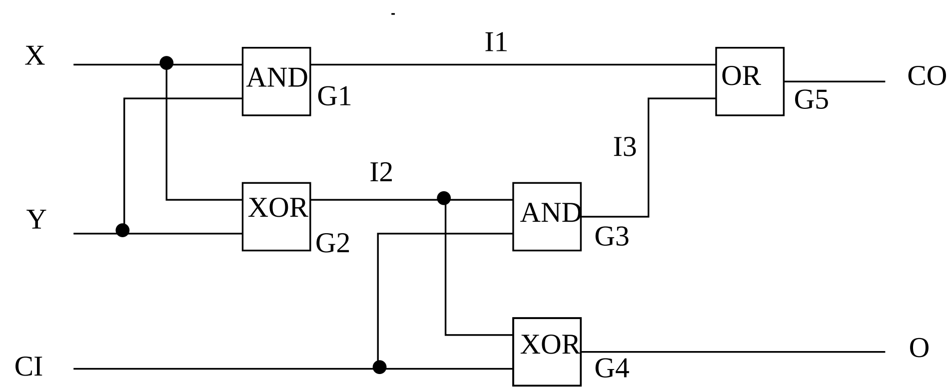
$$\begin{aligned} X &= 2Y + Z - 1 \ \wedge \\ Y &= -1 \ \wedge \\ -2 + Z - 1 - 1 &= 5 + Z \end{aligned}$$

La dernière équation est $-4 = 5$ ce qui n'est pas satisfaisable.

Contraintes d'arbres

- Exemple : $list(a, list(b, Y)) = list(a, L)$
- Une équation de termes et de la forme $s = t$ où s et t sont des termes.
- Une affection θ est une solution de $s = t$, si $\theta(s)$ et $\theta(t)$ sont syntaxiquement identiques.
- Exemple : $\theta = \{L \leftarrow list(b, Y)\}$ est une solution de $list(a, list(b, Y)) = list(a, L)$.
- Contrainte d'arbre : Conjonction d'équations de termes.
- Solutionneur de contraintes d'arbres : Algorithme d'unification

Contraintes booléennes



$$I1 \leftrightarrow X \& Y \ \wedge \ I2 \leftrightarrow X \oplus Y \ \wedge \ I3 \leftrightarrow I2 \& CI \ \wedge \ O \leftrightarrow I2 \oplus CI \ \wedge \ CO \leftrightarrow I1 \vee I3$$

Solutionneur de contraintes booléennes

- Problème NP-complet
- Solutionneur probabiliste :
  - Entrée : C une formule booléenne, ϵ entre 0 et 1
  - Soit m le nombre de contraintes simples dans C.
  - $n := \lceil ln(\epsilon)/ln(1-(1-1/m)^m)\rceil$
  - Pour $i := 1$ à n faire:
    * Générer aléatoirement une affectation θ pour toutes les variables de C
    * Si θ satisfait X alors retourner $vrai$
  - retourner $inconnu$

Solutionneur de contraintes

- Un solutionneur de contraintes est une fonction sol qui prend une contrainte C et qui retourne $vrai$, $faux$ ou $inconnu$
  - Si $sol(C) = vrai$ alors C est satisfaisable
  - Si $sol(C) = faux$ alors C est insatisfaisable
- Un solutionneur de contraintes se comporte bien, s'il est
  - basé sur les ensembles : si $ensemble(C_1) = ensemble(C_2)$ alors $sol(C_1) = sol(C_2)$
  - monotone : si $sol(C_1) = faux$ alors $sol(C_1 \wedge C_2) = faux$
  - indépendant des noms des variables : Si C_1 est une variante de C_2 alors $sol(C_1) = sol(C_2)$ (Une variante de C est la contrainte obtenu à partir de C en rénommant ses variables).
- Un solutionneur de contraintes est complet, s'il répond toujours soit $vrai$ soit $faux$ (jamais $inconnu$).

Variables déterminées et propagation locale

- Solutionneur général
- Idée : Détecter des variables dont les valeurs sont fixées par la contrainte et les éliminer
- Une variable x est déterminé par C d'avoir la valeur e (une expression sans variables), si toute solution de C est aussi une solution de $x = e$.
- Une contrainte est en forme résolue détérminée, si elle a la forme

  $$x_1 = e_1 \wedge \cdots \wedge x_n = e_n$$

  où les variables $x_1, \ldots, x_n$ sont distinctes et les expressions $e_1, \ldots, e_n$ ne contiennent pas de variables.

Algorithme propagation locale

- Entrée: Une contrainte C
- On maintient une contrainte en forme normale résolu détérminée C_1 (initialement C_1 est la conjonction vide) et une contrainte C_2 qui n'est pas encore résolue (au début $C_2 = C$)
- Tant que C_2 n'est pas vide et C_1 ou C_2 ont changé
  - On choisit une contrainte simple c de C_2
  - Si c est insatisfaisable, on termine.
  - Si c est satisfaisable avec $variables(c) = \emptyset$ alors $C_2 := C_2$ sans c.
  - Sinon, on teste, si à partir de c (uniquement !) on peut déterminer la valeur de certaines variables. Si c'est possible, on ajoute ces valeurs dans C_1 et on change C_2 en remplacant les valeurs des variables nouvellement déterminées.

  Si C_2 est vide, on retourne $vrai$ sinon $inconnu$.

On a besoin d'un solutionneur pour des contraintes simples et d'une fonction qui étant donné une contrainte simple retourne une forme résolue détérminée si possible.

Simplification de contraintes

Deux contraintes équivalentes représente la même information, mais une peut être plus simple que l'autre

$X \geq 1 \wedge X \geq 3 \wedge 2 = Y + X$	
$X \geq 3 \wedge 2 = Y + X$	enlever des contraintes rédondantes,
$3 \leq X \wedge X = 2 - Y$	réécrire une contrainte simple,
$X = 2 - Y \wedge 3 \leq X$	changer l'ordre,
$X = 2 - Y \wedge 3 \leq 2 - Y$	substituer en utilisant une équation,
$X = 2 - Y \wedge Y \leq -1$	préservent l'équivalence

Contraintes redondantes

- Une contrainte C_1 implique une autre contrainte C_2, si les solutions de C_2 sont un sous-ensemble des solutions de C_1.
- Une contrainte C_2 est appelé redondante par rapport à C_1, si C_1 implique C_2.
- On écrit $C_1 \rightarrow C_2$.
- Par exemple, $X \geq 3 \rightarrow X \geq 1$
- $Y \leq X + 2 \wedge Y \geq 4 \rightarrow X \geq 1$
- $machin(X, X) = machin(Z, b) \rightarrow Z = b$
- Une contrainte $C = c_1 \wedge c_2 \wedge \cdots \wedge c_n$ est sans redondance, si aucun c_i est redondant par rapport aux autres contraintes simples (On n'a pas $\bigwedge_{j \neq i} c_j \rightarrow c_i$)

Contraintes redondantes

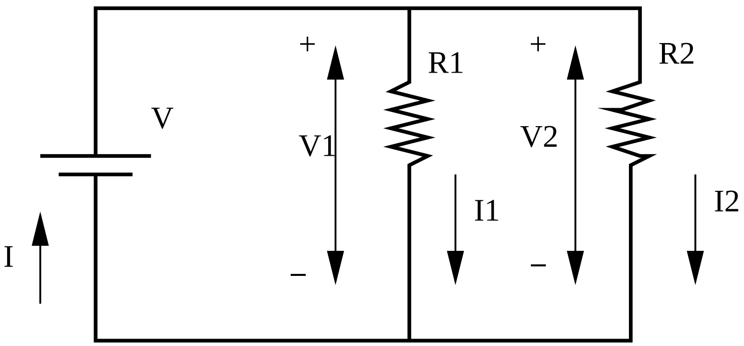
- On peut enlever une contrainte simple qui est redondante par rapport au reste de la contrainte.
- $X \geq 1 \wedge X \geq 3 \leftrightarrow X \geq 3$
- $Y \leq X + 2 \wedge X \geq 1 \wedge Y \geq 4 \leftrightarrow Y \leq X + 2 \wedge Y \geq 4$
- $machin(X, X) = machin(Z, b) \wedge Z = b$
  $\leftrightarrow machin(X, X) = machin(Z, b)$
- On obtient une contrainte plus simple.

Solutionneur qui renvoie une forme résolue

- Exemples: Algorithme d'unification, Gauss-Jordan
- Il obtient une contrainte en forme résolue équivalente à la contrainte initiale
- Il peut être donc un simplificateur
- $machin(X, X) = machin(Z, b) \wedge Y = truc(X) \wedge truc(Z) = Y \wedge Z = b$
  $\leftrightarrow X = b \wedge Z = b \wedge Y = truc(b)$
- $X = 2 + Y \wedge 2Y + X - T = Z \wedge X + Y = 4 \wedge Z + T = 5$
  $\leftrightarrow X = 3 \wedge Y = 1 \wedge Z = 5 - T$

Projection

Il est très important de pouvoir simplifier, si on est seulement intéressé par quelques variables de la contrainte.



$V1 = I1 * R1 \ \wedge\ V2 = I2 * R2 \ \wedge\ V - V1 = 0 \ \wedge\ V - V2 = 0 \ \wedge$
$V1 - V2 = 0 \ \wedge\ I - I1 - I2 = 0 \ \wedge\ -I + I1 + I2 = 0 \ \wedge\ R1 = 5 \ \wedge\ R2 = 10$

Simplifié par rapport à $\{V, I\}$: $V = (10/3) * I$

Projection

- La projection d'une contrainte C_1 sur un ensemble de variables V est une contrainte C_2 tel que
  - C_2 ne contient que les variables V
  - Chaque solution de C_1 est une solution de C_2
  - Chaque solution de C_2 peut être étendue vers une solution de C_1
- Exemple:

$X \geq Y \ \wedge\ Y \geq Z \ \wedge\ Z \geq 0$	$X \geq 0$
$\{X \leftarrow 0, Y \leftarrow 0, Z \leftarrow 0\}$	$\{X \leftarrow 0\}$
$\{X \leftarrow 3, Y \leftarrow 2, Z \leftarrow 1\}$	$\{X \leftarrow 3\}$

Algorithme de Fourier

- Permet de projeter une contrainte C contenant des inéquations
- Élimination d'une variable y :
  - Écrire chaque inéquation avec y dans la forme :
    $t_1 \geq y$ ou $y \geq t_2$
  - Pour chaque pair $t_1 \geq y$ $y \geq t_2$ on met une nouvelle inéquations $t_1 \geq t_2$
  - Résultat : Toutes les nouvelles inéquations + ceux de C ne contenant pas y
- Exemple : $X - 1 \leq Y \ \wedge\ -1 - X \leq Y \ \wedge\ Y \leq 1 - X \ \wedge\ Y \leq 1 + X$

  $X - 1 \leq Y$ et $Y \leq 1 - X$ donne $X \leq 1$
  $X - 1 \leq Y$ et $Y \leq 1 + X$ donne $0 \leq 2$
  $-1 - X \leq Y$ et $Y \leq 1 - X$ donne $0 \leq 2$
  $-1 - X \leq Y$ et $Y \leq 1 + X$ donne $-1 \leq X$

  donne $X \leq 1 \ \wedge\ -1 \leq X$

Projeter des contraintes de termes

- On peut aussi projeter des contraintes de termes
- Exemple : $machin(Y, Y) = machin(X, Z) \ \wedge\ truc(Z) = truc(T)$ projeté sur $\{X, Z\}$ donne $X = Z$
- Qu'est-ce que $X = machin(Y, Z)$ projeté sur $\{X\}$?

Simplificateur de contraintes

- Deux contraintes C_1 et C_2 sont équivalentes par rapport à un ensemble de variables V si
  - on prend une solution d'une des contraintes et on la restraint sur les variables V, cette solution restrainte peut être étendue vers une solution de l'autre contrainte.
- Exemple: $X = truc(Y)$ et $X = truc(Z)$ sont équivalents par rapport à $\{X\}$

$X = truc(Y)$	$\{X\}$	$X = truc(Z)$
$\{X \leftarrow truc(a), Y \leftarrow a\}$	$\{X \leftarrow truc(a)\}$	$\{X \leftarrow truc(a), Z \leftarrow a\}$
$\{X \leftarrow truc(b), Y \leftarrow b\}$	$\{X \leftarrow truc(b)\}$	$\{X \leftarrow truc(b), Z \leftarrow b\}$

Simplificateur de contraintes

- Un simplificateur de contraintes est une fonction simpl qui prend une contraintes C et un ensemble de variables V et retourne une contrainte C_{simpl} qui est équivalente à C par rapport à V.
- Par exemple, on peut faire un simplificateur pour des inéquations en utilisant l'algorithme de Fourier
- Simplificateur pour les contraintes de termes :
  - Appliquer le solutionneur de contraintes de termes sur C. On obtient C_1.
  - Pour chaque équation $x = t$ dans C_1 faire
    - si x est dans V alors
      - si t est une variable qui n'est pas dans V, substituer x pour t partout dans C_1 et dans le $résultat$.
      - sinon ajouter $x = t$ dans le $résultat$.
  - retourner $résultat$

Exemple de simplifaction de contraintes de termes

- On veut simplifier $h(f(X,Y), Z, g(T)) = h(f(g(T), X), f(X,X), g(U))$ par rapport à $\{Y, T\}$
- Le solutionneur de contraintes donne une contrainte équivalente:
  $Z = f(g(U), g(U)) \ \wedge \ X = g(U) \ \wedge \ Y = g(U) \ \wedge \ T = U$
- On enlève les deux premières équations, on garde la troisième et on utilise la dernière pour substituer U par T
- Résultat: $Y = g(T)$

Autre exemple: $X = f(X_1, X_1) \wedge X_1 = f(X_2, X_2) \wedge \cdots \wedge X_{n-1} = f(X_n, X_n)$ projeté sur $\{X\}$ donne ?

Propriétés de simplificateur

Un simplificateur est

- projetant, si $variables(simpl(C, V)) = V$
- projetant faible, si pour toute contrainte C_2 équivalente à C_1 par rapport à V, on a $|variables(simpl(C_1, V)) - V| \leq |variables(C_2) - V|$ (c.-à-d. un simplificateur projetant faible n'utilise jamais plus de variables que nécéssaire)
- sans redondance, si $simpl(C, V)$ est sans redondance.
- Un simplificateur projetant faible peut être utilisé comme solutionneur. Comment ?

Implication et équivalence

D'autres opérations importantes sont

- Implication: Tester si C_1 implique C_2
  - $impl(C_1, C_2)$ répond $vrai$, $faux$ ou $inconnu$
- Équivalence: Tester si C_1 et C_2 sont équivalents
  - $equiv(C_1, C_2)$ répond $vrai$, $faux$ ou $inconnu$

Simplificateur canonique

- Un simplificateur est canonique: Si C_1 et C_2 sont équivalentes par rapport à V alors $simpl(C_1, V) \equiv simpl(C_2, V)$ (c.-à-d. C_1 et C_2 sont syntaxiquement identiques)
- La forme canonique d'une contrainte C est $simpl(C, variables(C))$