UNIVERSITÉ DE MONTPELLIER
-
FACULTÉ DES SCIENCES

# Examen d'Intelligence Artificielle
# HMIN107

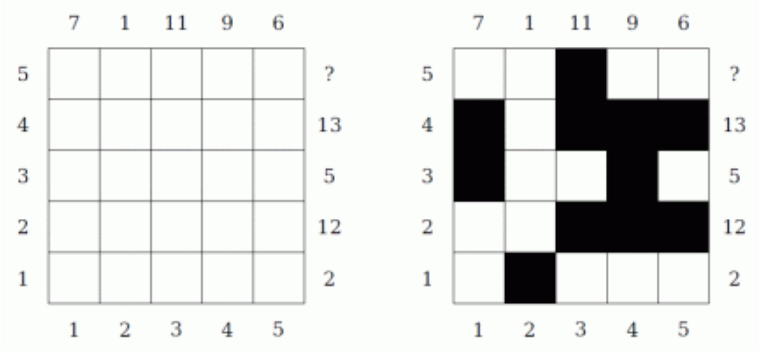
| | |
|---|---|
| Session : 1 | Durée de l'épreuve : 2 heures |
| Date : Janvier 2016 | Documents autorisés : tous |
| Master Informatique | |

## 1 CSP

Le jeu du Bokkosu consiste à noircir certaines cases d'une grille carrée $n \times n$ donnée. Les nombres à gauche et en bas de la grille permettent d'associer des valeurs verticales et horizontales à chaque case de la grille ; ces valeurs vont toujours de 1 à $n$. Les valeurs verticale et horizontale de la case en haut à gauche d'une grille sont donc toujours $n$ et 1 (sur l'exemple ci-dessous 5 et 1).

Les nombres en haut et à droite de la grille indiquent, respectivement, la somme des valeurs verticales des cases noircies de la colonne correspondante, et la somme des valeurs horizontales des cases noircies de la ligne correspondante. Sur la figure ci-dessous on donne une grille et sa solution. On peut, par exemple, voir que la somme des valeurs verticales $4+3$ des cases de la colonne 1 vaut bien 7 ; la somme des valeurs horizontales $1+3+4+5$ des cases de la ligne 4 vaut bien 13.

Un ? laisse le joueur libre d'avoir n'importe quelle somme sur la ligne ou colonne corrrespondante.




Proposer une modélisation de ce jeu comme un CSP.

## 2 Backtrack

Soit le CSP $P = (X, D, C)$ où :
$X = \{x_1, \ldots, x_9\}$
$D = \{dom_i\}$, $i : 1 \ldots 9$, avec $dom_i = \{1, 2, 3\}$ (on considère le même domaine pour chacune des variables)
$C = \{c_1 : allDiff^{1}(x_1, x_2, x_3), \quad c_2 : x_4 \neq x_5, \quad c_3 : x_4 \neq x_6,$
$c_4 : x_5 \neq x_6, \quad c_5 : x_7 \neq x_8, \quad c_6 : x_8 \neq x_9,$
$c_7 : x_7 \neq x_4, \quad c_8 : x_1 > x_8 + 1, \quad c_9 : x_5 \neq 2,$
$c_{10} : x_2 = x_9, \quad c_{11} : x_5 \neq x_3 \}$

1. $allDiff$ est une contrainte globale qui indique que toutes les variables sur lesquelles elle porte doivent recevoir des valeurs différentes.

1. Dessinez le graphe de contraintes correspondant à $P$.
2. Dans l'objectif d'appliquer l'algorithme de Backtrack, calculer l'ordre d'affectation des variables en appliquant l'heuristique suivante :
   - choisir la variable qui permettra de vérifier à l'étape courante le plus de contraintes,
   - en cas d'égalité, choisir la variable qui partage le plus grand nombre de contraintes avec les variables restant à assigner,
   - et toujours en cas d'égalité, choisir la variable $x_i$ ayant le plus petit $i$.

   Vous compléterez le tableau suivant en indiquant à chaque étape : la variable choisie, les contraintes vérifiées à cette étape, et pour chaque variable le nombre de contraintes que l'on pourrait vérifier si on sélectionnait cette variable à l'étape suivante ainsi que le nombre de contraintes qu'elle partage avec les variables qui restent à assigner (par exemple, à l'étape initiale 0, si on choisit $x_1$, on ne pourra vérifier aucune contrainte sur les 3 qui restent à vérifier pour $x_1$, par contre si on choisit $x_5$, on pourra vérifier 1 contrainte sur les 3 qui restent à vérifier pour $x_5$).

| Etape | V | C | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | 0 / 2 | 0 / 2 | 0 / 2 | 0 / 3 | 1 / 3 | 0 / 2 | 0 / 2 | 0 / 3 | 0 / 2 |
| 1 | $x_5$ | $c_9$ | 0 / 2 | 0 / 2 | 1 / 2 | 1 / 3 | - | 1 / 2 | 0 / 2 | 0 / 3 | 0 / 2 |
| 2 | $x_4$ | $c_2$ | | | | | | | | | |
| ... | | | | | | | | | | | |

3. Appliquez l'algorithme de Backtrack en utilisant l'ordre sur les variables calculé ci-dessus et en testant les valeurs dans l'ordre croissant. Vous représenterez l'arbre de recherche développé et indiquerez à chaque backtrack le numéro de la contrainte violée.
   Indiquez la première solution trouvée.

## 3 Calcul d'homomorphisme et CSP

On note les constantes par des lettres du début de l'alphabet ($a$, $b$, $c$, ...) et les variables par des lettres de la fin de l'alphabet ($t$, $u$, $v$, $w$, $x$, $y$, $z$, ...).

**Rappels.** *Le problème* CSP (satisfaction de contraintes) *prend en entrée un réseau de contraintes $P = (X, D, C)$ [où $X = \{x_1, \ldots, x_n\}$ est un ensemble de variables, $D = \{D_i | x_i \in X\}$ est l'ensemble des domaines des variables, $C$ est l'ensemble des contraintes portant sur les variables de $X$] et demande si ce réseau admet une solution. Le problème* HOM (homomorphisme) *prend en entrée un couple $(\mathcal{A}_1, \mathcal{A}_2)$ [où $\mathcal{A}_1$ et $\mathcal{A}_2$ sont des ensembles d'atomes représentant des formules existentielles conjonctives de la logique du premier ordre] et demande s'il existe un homomorphisme de $\mathcal{A}_1$ dans $\mathcal{A}_2$. On suppose ici que $\mathcal{A}_2$ ne contient aucune variable, autrement dit chacun de ses atomes ne porte que sur des constantes.*
*Une* réduction *$f$ d'un problème de décision $P_1$ à un problème de décision $P_2$ transforme toute instance $I$ de $P_1$ ("la donnée en entrée de $P_1$") en une instance $f(I)$ de $P_2$ de façon à ce que la réponse à $P_1$ pour $I$ soit oui si et seulement si la réponse à $P_2$ pour $f(I)$ est oui. Une telle réduction est* polynomiale *si pour tout $I$ elle se calcule en temps polynomial en la taille de $I$.*

Dans cet exercice, on considère des instances de CSP dans lesquelles toutes les contraintes sont données en *extension* (c'est-à-dire par énumération des tuples de valeurs compatibles), sauf la *contrainte d'égalité* qui peut être donnée en intension. Par contre, on n'a *pas d'égalité* dans l'instance de HOM, seulement des prédicats quelconques. On va s'intéresser à des réductions polynomiales entre HOM et CSP.

1. Soit l'instance de HOM suivante :
   $\mathcal{A}_1 = \{\ p(x,y),\ q(y,z,a),\ p(a,w),\ q(w,x,w)\ \}$
   $\mathcal{A}_2 = \{\ p(a,a),\ p(a,b),\ p(b,c),\ q(c,c,a),\ q(a,b,a)\ \}$.
   Quels sont tous les homomorphismes de $\mathcal{A}_1$ dans $\mathcal{A}_2$ ? On ne vous demande pas d'expliquer comment vous faites le calcul.
2. On considère d'abord le cas particulier de HOM dans lequel chaque atome de $\mathcal{A}_1$ ne comporte que des *variables distinctes* deux à deux.
   Dans le cadre de cette restriction, définir une réduction polynomiale $f_1$ de HOM à CSP et une réduction polynomiale $f_2$ de CSP à HOM, de façon à satisfaire la propriété suivante : pour tout couple $(\mathcal{A}_1, \mathcal{A}_2)$ instance de HOM, $f_2(f_1((\mathcal{A}_1, \mathcal{A}_2))) = (\mathcal{A}_1, \mathcal{A}_2)$ (autrement dit lorsqu'on transforme nos deux ensembles d'atomes en un réseau de contraintes et qu'on effectue la transformation inverse on retrouve exactement les ensembles d'atomes initiaux).
   Illustrer votre transformation avec :
   $\mathcal{A}'_1 = \{\ p(x,y),\ q(y,z,u),\ p(u,t),\ q(t,x,v)\ \}$
   $\mathcal{A}_2 = \{\ p(a,a),\ p(a,b),\ p(b,c),\ q(c,c,a),\ q(a,b,a)\ \}$.
3. Étendre les transformations données à la question précédente de façon à pouvoir traiter un ensemble $\mathcal{A}_1$ sans restriction, tout en conservant la propriété $f_2 \circ f_1 =$ *identité*.
   Illustrer votre transformation avec le couple $(\mathcal{A}_1, \mathcal{A}_2)$ de la question 1.

# 4 Chaînage avant avec des règles positives d'ordre 1

On considère la base de connaissances ci-dessous.

- Base de règles (où $a$ et $b$ sont des constantes) :
  $R_1 : p(x) \wedge q(x,y) \rightarrow p(y)$
  $R_2 : q(x,y) \wedge q(y,z) \rightarrow q(x,a)$
  $R_3 : q(x,b) \rightarrow r(x)$
  $R_4 : q(x,x) \rightarrow r(x)$
  $R_5 : r(x) \wedge q(x,y) \rightarrow p(x)$

- Base de faits (où tous les termes sont des constantes)
  $\{\ p(b),\ q(a,e),\ q(b,c),\ q(c,d),\ q(e,d)\ \}$

Saturer la base de faits avec les règles, en procédant en largeur (à chaque étape, calcul de tous les nouveaux homomorphismes des hypothèses de règles dans la base de faits courante, puis application des règles selon ces homomorphismes avant de passer à l'étape suivante). Présenter les résultats selon le format suivant :

| Etape | Règle applicable | Homomorphisme | fait produit | utile ? |
|---|---|---|---|---|
| *n° étape* | *n° règle* | | | *oui/non* |
| ... | | | | |

# 5 Graphe de dépendance des règles

On rappelle qu'une règle $R_j$ *dépend* d'une règle $R_i$ (où $R_i$ et $R_j$ peuvent être la même règle) si et seulement s'il existe une base de faits telle qu'une application de $R_i$ peut susciter une nouvelle application de $R_j$ (c'est-à-dire avec un nouvel homomorphisme de l'hypothèse de $R_j$).

1. (Question préalable) Etant données deux règles $R_1$ et $R_2$, comment déterminer en pratique si $R_2$ dépend de $R_1$ ?
2. On considère la base de règles suivante (où $a$ et $b$ sont des constantes) :
   $R_1 : p(x) \wedge q(x, y) \rightarrow p(y)$
   $R_2 : q(x, y) \wedge q(y, z) \rightarrow q(x, a)$
   $R_3 : q(x, b) \rightarrow r(x)$
   $R_4 : q(x, x) \rightarrow r(x)$
   $R_5 : r(x) \wedge q(x, y) \rightarrow p(x)$
   Dessiner le graphe de dépendance des règles associé à cette base de règles, avec la convention suivante : un arc de $R_i$ vers $R_j$ signifie que $R_j$ dépend de $R_i$ ("$R_i$ peut déclencher $R_j$").
3. Il peut arriver qu'une règle $R_j$ dépende d'une règle $R_i$, mais que, quelle que soit la base de faits, toute nouvelle application de $R_i$ suscitée par $R_i$ ne puisse produire qu'un fait inutile (c'est-à-dire déjà présent dans la base de faits).
   - Donner un exemple illlustrant cela.
   - Comment pourrait-on affiner le test de dépendance de façon à exclure ces dépendances inintéressantes ?