Université Montpellier II – Master Informatique Janvier 2014
Intelligence Artificielle (GMIN101)

Examen

Durée : 2 heures
Documents autorisés

Exercice 1. Raisonnements en logique des propositions (4pts)

Soit la base de connaissances BC1 = (BF1, BR1) où **BF1** = {Z, E} et BR1 contient les règles suivantes :

R1 : $Z \rightarrow B$
R2 : $Z \wedge R \rightarrow A$
R3 : $B \wedge E \rightarrow R$
R4 : $E \wedge C \rightarrow D$

1. Quelle est la base de faits obtenue en saturant BF1 avec BR1? On ne demande pas de détailler les applications de règles.

2. On passe à des règles et faits avec négation : une règle a pour hypothèse une conjonction de littéraux et pour conclusion un littéral (rappelons qu'un littéral est un atome ou la négation d'un atome) ; un fait est un littéral. On considère la base de connaissances BC2 = (BF2, BR2) où BF2 = {A} et BR2 contient les règles suivantes :

   R1 : $A \wedge \neg C \rightarrow E$
   R2 : $A \wedge B \rightarrow C$
   R3 : $A \wedge \neg B \rightarrow C$
   R4 : $A \wedge C \rightarrow D$

   Quelle est la base de faits obtenue en saturant BF1 avec BR1?

3. On considère à nouveau BC2. Représenter la question « D se déduit-il de BC2 ? » par une instance du problème SAT (ou UNSAT). Dérouler l'algorithme DPLL sur cette instance et indiquer quelle est la réponse à la question.

4. Le chaînage avant est-il complet sur des règles avec négation ? Justifier votre réponse.

Exercice 2. Chaînage avant avec des règles d'ordre 1 (4pts)

On considère la base de connaissances suivante :

- Règles (les quantificateurs universels sont implicites)

  R1 : $r(x1,y1,z1) \wedge p(x1,y1) \rightarrow p(y1,z1)$
  R2 : $p(x2,y2) \wedge p(y2, z2) \rightarrow p(x2,z2)$

- Faits r(a,b,c) r(b,b,c) r(b,c,d)
  p(a,b) p(c,d) p(e,f) p(e,e)

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	Application utile ?	Fait ajouté (si application utile)
n° étape	n° règle	…	oui/non	…
…	…	…	oui/non	…

Exercice 3. Algorithme de recherche d'homomorphisme (4pts)

1. Dessiner l'arbre construit par l'algorithme de backtrack du cours recherchant s'il existe un homomorphisme de $\mathcal{A}1$ dans $\mathcal{A}2$ avec :

   $\mathcal{A}1$ = {p(x,y), p(y,z), q(z,x)}, où x, y et z sont des variables

   $\mathcal{A}2$ = {p(a,b), p(b,a), q(b,b), q(a,c), q(b,c)} où a, b et c sont des constantes

   Les variables de $\mathcal{A}1$ sont considérées dans l'ordre x y z et les constantes de $\mathcal{A}2$ dans l'ordre a b c.
   - Indiquer à chaque niveau de l'arbre quels atomes de $\mathcal{A}1$ sont considérés pour le test de solution partielle.
   - Marquer par X une feuille échec (la condition de solution partielle n'est pas satisfaite) et par O une solution, s'il en existe une.

2. La façon la plus simple de calculer l'ensemble des candidats pour une variable *x* de $\mathcal{A}1$ consiste à retourner l'ensemble des termes de $\mathcal{A}2$. Comment calculer de façon plus fine l'ensemble des images possibles d'une variable en phase de prétraitement ? Illustrer sur l'exemple.

Exercice 4. Modélisation en contraintes (4pts)

Une firme automobile élabore un nouveau modèle de voiture fabriquée dans toute l'Europe :
- les portières et le capot sont fabriqués à Lille où le constructeur ne dispose que de peinture rouge, jaune et noire ;
- la carrosserie est faite à Hambourg où l'on a de la peinture blanche, jaune, rouge et noire ;
- les pare-chocs, réalisés à Palerme, sont toujours blancs ;
- la bâche du toit ouvrant, faite à Madrid, ne peut être que blanche, jaune ou rouge ;
- les enjoliveurs sont fabriqués à Athènes où l'on a de la peinture rouge et jaune.

Le constructeur de la voiture a les exigences suivantes :
- la carrosserie doit être de la même couleur que les portières, qui doivent être de la même couleur que le capot ;
- les enjoliveurs, les pare-chocs et la bâche du toit ouvrant doivent être (strictement) plus clairs que la carrosserie (on considère que jaune est plus clair que rouge ; blanc et noir étant les deux extrêmes).

1. Représenter ce problème de configuration par un problème de satisfaction de contraintes (CSP) binaire utilisant des contraintes données en extension (c'est-à-dire par énumération des couples de valeurs compatibles).

2. Développer l'algorithme de Backtrack avec Forward Checking en choisissant à chaque étape la variable la plus contrainte et la valeur la moins contraignante. Vous expliquerez bien l'évolution des domaines de chaque variable après chaque étape de Forward Checking et fournirez la première solution trouvée.

3. Combien y a-t-il de configurations possibles ?

Exercice 5. De "CSP" à "SAT" (4pts)

Le problème *CSP (satisfaction de contraintes)* prend en entrée un réseau de contraintes et demande si ce réseau admet une solution. Un réseau de contraintes est composé d'un ensemble X = $\{x_1, \ldots, x_n\}$ de variables, d'un domaine D_i pour chaque variable x_i, et d'un ensemble C de contraintes portant sur les variables de X. On se restreint ici à des contraintes données en extension (c'est-à-dire par énumération des tuples de valeurs compatibles). Voir l'exemple ci-dessous.

Le problème *SAT* prend en entrée une conjonction de clauses *S* (ou ensemble de clauses) en logique propositionnelle et demande si *S* est satisfiable.

On veut construire une réduction de CSP à SAT, c'est-à-dire une transformation *t* de tout réseau de contraintes R en un ensemble de clauses *S* de façon à ce que R admette une solution si et seulement si S est satisfiable.

1. On cherche d'abord à traduire en logique des propositions le fait que "toute variable x de X doit avoir une valeur prise dans son domaine", sans chercher à traduire la notion de contrainte pour l'instant. On note "xv" le symbole propositionnel (ou "variable propositionnelle") exprimant que "la variable x prend la valeur v", où x est une variable de X et v une valeur de son domaine. Avec cette notation, exprimer sous forme de clauses les conditions suivantes :

   a. Toute variable x de X prend une valeur de son domaine ;
   b. Toute variable ne peut prendre qu'une valeur.

   Illustrer ceci sur l'exemple.

2. Ajouter à la transformation la représentation des contraintes.
   Indication : énumérer les tuples de valeurs interdits par les contraintes plutôt que les tuples autorisés.

   Illustrer ceci sur l'exemple.

<u>Exemple</u>

X = {x1, x2, x3, x4}

D1 = {a,b,c}
D2 = **D3** = {a, b}
D4 = {b, c}

C1

x1	x2
a	a
b	b
c	b

C2

x2	x3	x4
a	a	b
a	a	c
b	b	b
b	a	c