M1 - DAC
IAMSI
Intelligence Artificielle et
Manipulation Symbolique de l'Information

Année 2015-2016

Examen réparti n° 2

UPMC SORBONNE UNIVERSITÉS

Rappels : **Aucun document n'est autorisé.** Le barème n'est donné qu'à titre indicatif. Une annexe contenant des rappels de cours est donnée en fin de sujet. **Ce sujet comporte 4 pages.**

Exercice 1 Programmation en ASP – 4 points

1. Construire, en ASP, un programme $fib(N, F)$ qui calcule la valeur F du $N^{ième}$ nombre de Fibonacci que l'on note fib_N.
   Remarque : rappelons que $fib_1 = 1$, $fib_2 = 1$ et $fib_N = fib_{N-1} + fib_{N-2}$.
2. Construire, toujours en ASP, un prédicat $maxFib(N, P)$ qui donne pour chaque entier N le plus grand entier P tel que $fib_P < N$.

Exercice 2 Interprétation de trois programmes en ASP – 3 points

Donner tous les ensembles réponses des trois programmes suivants en détaillant et en justifiant tous les calculs.

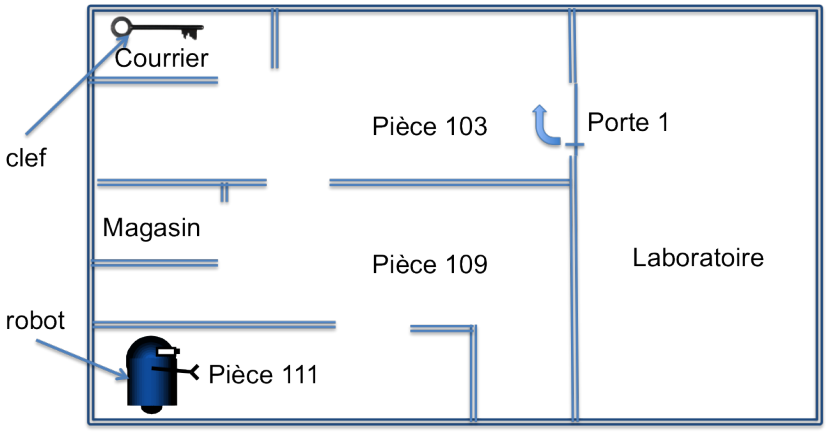
$$\Pi_1 \begin{cases} \neg a. \\ b \leftarrow \neg a, \ \textbf{not}\ c. \\ c \leftarrow \ \textbf{not}\ b. \end{cases} \qquad \Pi_2 \begin{cases} \{p(a),\ p(b),\ p(c)\}2. \\ p(a) \leftarrow p(X). \end{cases} \qquad \Pi_3 \begin{cases} 2\{p, s, q\}. \\ r \leftarrow p. \end{cases}$$

Exercice 3 "Robot de livraison autonome" – 5 points

Considérons les quatre actions suivantes,

- **déplacer(Ag, P1, P2)** : *L'agent Ag se déplace de la pièce P1 à la pièce P2*
  Remarque : l'agent peut aller d'une pièce $P1$ à une pièce $P2$ si elles sont adjacentes.
- **prendre(Ag, Obj, Pos)** : *L'agent Ag saisit l'objet Obj s'ils sont tous les deux à la position Pos*
- **lâcher(Ag, Obj, Pos)** : *L'agent Ag lâche l'objet Obj à la position Pos où il se trouve*
- **ouvrir-porte(Ag, Porte)** : *L'agent Ag ouvre la porte "Porte", ce qui suppose qu'il tienne la clef correspondante et qu'il se trouve devant la porte, c'est-à-dire en position "Porte".*

1. En supposant que l'on se donne les prédicats $ouvre(Clef, Porte)$, $adjacent(P1, P2)$, $position(Ag, Pos)$, $tient(Ag, Obj)$, $entre(Porte, P1, P2)$, $fermée(Porte)$, $ouverte(Porte)$, représenter, dans le formalisme STRIPS les quatre actions précédentes.
2. Représenter à l'aide du formalisme STRIPS le schéma suivant dans lequel la clef qui ouvre la Porte 1 se trouve au milieu du courrier et le robot dans la pièce 111.
   Remarque : deux pièces sont adjacentes si elles sont soit ouvertes l'une sur l'autre, soit séparées par une porte qui est ouverte.



3. On souhaite que le robot aille dans le laboratoire. Exprimer la condition terminale correspondante.

4. Donner un plan qui permette de passer de l'état initial décrit sur la figure à un état qui vérifie la condition terminale. *(On ne demande pas de simuler la génération de ce plan.)*
5. Montrer l'évolution de la pile des buts et de l'état pour les quatre premières étapes de la résolution.

Exercice 4 Ensembles fréquents – 3 points

Dans le tableau 1, les E_i correspondent à 8 exemples d'apprentissage, décrits par 5 attributs : *Couleur*, *Aboie beaucoup*, *Taille*, *Intelligence* et *Shetland*.

Exemple	Couleur	Aboie beaucoup	Taille	Intelligence	Shetland
E_1	*Noir*	*Oui*	*Petit*	*Élevée*	*Peut-être*
E_2	*Jaune*	*Non*	*Grand*	*Moyenne*	*Non*
E_3	*Jaune*	*Non*	*Grand*	*Faible*	*Non*
E_4	*Blanc*	*Oui*	*Moyen*	*Élevée*	*Peut-être*
E_5	*Noir*	*Non*	*Moyen*	*Élevée*	*Non*
E_6	*Brun*	*Oui*	*Petit*	*Élevée*	*Peut-être*
E_7	*Brun*	*Oui*	*Petit*	*Faible*	*Non*
E_8	*Noir*	*Non*	*Moyen*	*Moyenne*	*Non*

TABLE 1 – *Ensemble d'apprentissage E*

Ici, nous faisons appel à l'algorithme *Apriori* pour engendrer tous les ensembles fréquents de l'ensemble d'apprentissage E donné dans le tableau 1 avec un support minimal $minsup = 0.3$.

1. Rappeler les étapes de l'algorithme *Apriori*.
2. Engendrer tous les ensembles fréquents à l'aide de l'algorithme *Apriori*.

Exercice 5 Extraction de règles d'association – 2 points

1. Indiquer comment on extrait les règles d'association à partir d'un ensemble fréquent.
2. Engendrer trois règles d'association de confiance 100% à partir des ensembles fréquents trouvés à la question précédente.

Exercice 6 Construction d'un arbre de décision – 3 points

1. Rappeler comment on détermine le gain d'un attribut à partir de l'entropie pondérée par le nombre d'éléments et expliquer à quoi correspond cette formule.
2. Construire l'arbre de décision qui prédit la classe correspondant à l'attribut *Shetland* en fonction des autres attributs, pour les données de l'ensemble E, en faisant appel à la méthode de construction employée par l'algorithme ID3.

Remarque : le tableau 2, fourni en annexe, permet de traiter le problème sans calculette.

Annexe

Algorithme Smodels

On note Lit l'ensemble des littéraux du programme P.

On initialise le modèle en cours $M = M_{pos} \cup M_{not} \cup M_{Dpos} \cup M_{Dnot}$ à $\emptyset$ et le programme en cours à $P' = P$, toutes ces variables évoluent au cours de l'algorithme.

1. **Déduction**
   (a) Littéraux positifs : mise à jour de $M_{pos} := M_{pos} \cup \{a \in Lit \setminus M_{pos} | 'a.' \in P'\}$, puis, pour chaque atome a ainsi ajouté à M_{pos}, pour chaque règle $r \in P'$
   — Si a est la tête de r, on supprime la règle
   — Si a est dans le corps positif de r, on réécrit la règle r en supprimant a de son corps positif.
   — Si a est dans le corps négatif de r, on supprime la règle
   (b) Littéraux négatifs : mise à jour de $M_{not} := M_{not} \cup \{a \in Lit \setminus M_{not} | \forall r \in P' a \notin head(r)\}$, puis, pour chaque atome a ainsi ajouté à M_{not}, pour chaque règle $r \in P'$
   — Si a est dans le corps positif de r, on supprime la règle
   — Si a est dans le corps négatif de r, on réécrit la règle r en supprimant a de son corps négatif (i.e. en supprimant *not* a de son corps).

   Chacune de ces phases est répétée jusqu'à ce que M atteigne un point fixe.
2. **Analyse**
   — Si M est incohérent ($\bot \in M_{pos}$ ou $(M_{pos} \cup M_{Dpos}) \cap (M_{not} \cup M_{Dnot}) \neq \emptyset$) ou contradictoire ($\exists a \in Lit.\{a, \neg a\} \subseteq (M_pos \cup M_{Dpos})$), la branche ne produit aucun answer set. On backtracke (si possible) jusqu'au dernier point de décision (en remettant M et P' dans l'état où ils étaient au moment de la décision).
   — Si M est complet ($M_{pos} \cup M_{not} = Lit$), on a trouvé un answer set. On renvoie M_{pos}. Si on cherche plus d'answer set, on backtracke au point de décision précédent.
   — Sinon, on doit procéder à une décision.
3. **Décision** On choisit un littéral a dans $Lit \setminus M$ et on décide de le supposer positif ou négatif en le mettant respectivement dans M_{Dpos} ou M_{Dnot}.
   — Si on le rajoute dans M_{Dpos}, on fait une mise à jour de P' par a, mais **sans supprimer les règles qui ont a comme tête**.
   — Si on le rajoute dans M_{Dnot}, on fait une mise à jour de P' par *not* a.
   On reprend alors à l'étape de déduction avec ce nouveau P'.

Représentation des actions en STRIPS

`action(paramètres)`	
`precond`	conjonction de littéraux positifs sans symbole de fonction qui exprime ce qui doit être vrai pour que l'action puisse être exécutée
`delete`	conjonction de littéraux positifs sans symbole de fonction qui exprime ce qui n'est plus vrai après la réalisation de l'action
`add`	conjonction de littéraux positifs sans symbole de fonction qui exprime ce qui est rendu vrai par l'exécution de l'action

L'hypothèse cruciale suivante est faite : *toute formule non mentionnée dans la description de l'action reste inchangée.*

Algorithme APRIORI

APRIORI($B, minsup$)
Entrées : B ensemble de transactions
$minsup$ seuil de support
$C_1 \longleftarrow$ {singletons}
$k \longleftarrow 1$
tant que $C_k \neq \emptyset$ faire
pour chaque $c \in C_k$ faire
compte(c) = 0
pour chaque $t \in B$ faire
si $c \subset t$ alors compte(c) $\longleftarrow$ compte(c) +1
$F_k = \{c \in C_k \mid \frac{\text{compte}(c)}{|B|} \geq minsup\}$
$k \longleftarrow k + 1$
$C_k \longleftarrow$ APRIORI-GEN(F_{k-1})
retourner $\cup_k F_k$

APRIORI-GEN(F)
Entrée : F ensemble d'itemsets fréquents de longueur k
$C \longleftarrow \{c = f_1 \cup f_2 \in F \times F$ tels que $|c| = k + 1\}$
pour chaque $c \in C$ faire
pour chaque $i \in c$ faire
si $c \backslash \{i\} \notin F$ alors
$C \longleftarrow C \backslash \{c\}$
retourner C

Mesure d'information

— Entropie de Shannon de la classe C :

$$H_S(C) = -\sum_k p(c_k)\log(p(c_k))$$

— Forme conditionnelle de l'entropie de Shannon :

$$H_S(C|A) = -\sum_i p(v_i)\sum_k p(c_k|v_i)\log(p(c_k|v_i))$$

— Gain d'information de l'attribut A pour la classe C :

$$I_S(A,C) = H_S(C) - H_S(C|A)$$

Entropie pondérée (cas 2 classes)

On considère que la classe C peut prendre 2 valeurs $c_1 =' +'$ et $c_2 =' -'$.

Le tableau ci-dessous donne l'entropie pondérée $H_S(C)$ dans ce cas, pour un ensemble qui contient p éléments de la classe '+' et n éléments de la classe '−'.

$H_S(C)$ s'écrit alors : $-(n+p)\left(\frac{p}{n+p}\log\left(\frac{p}{n+p}\right)+\frac{n}{n+p}\log\left(\frac{n}{n+p}\right)\right)$

p/n	**1**	**2**	**3**	**4**	**5**
5	3.90	6.04	7.63	8.92	10.00
4	3.61	5.51	6.90	8.00	
3	3.25	4.85	6.00		
2	2.75	4.00			
1	2.00				

TABLE 2 – *Entropie pondérée*

Algorithme de construction d'arbre de décision (variante ID3 / CART)

— Entrée : $\mathcal{B}$, la base d'apprentissage
— Sortie : un arbre de décision

1. Pile $\mathcal{P} = \{\mathcal{B}\}$
2. Tant que $\mathcal{P}$ n'est pas vide :
   — soit $\mathcal{E}$ la tête de $\mathcal{P}$
   — si le critère d'arrêt est atteint pour $\mathcal{E}$ alors créer une feuille qui contient la classe majoritaire présente dans $\mathcal{E}$
   — sinon, avec $\mathcal{E}$:
   (a) calculer $H(C|A_j)$ pour tous les attributs A_j
   (b) choisir l'attribut A_j qui maximise $I_S(A_j, C)$
   (c) créer un nœud dans l'arbre de décision avec A_j
   (d) à l'aide de A_j, partitionner $\mathcal{E}$: créer autant de sous-ensembles d'exemples que de valeurs différentes de A_j
   (e) empiler les sous-ensembles obtenus dans $\mathcal{P}$