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

Année 2015-2016

Examen de rattrapage

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 du sujet.
Ce sujet (annexes comprises) comporte 4 pages.

Exercice 1 Quatre programmes en ASP – 4 points
Donner tous les ensembles-réponse de chacun des quatre programmes suivants en détaillant et en justifiant tous les calculs.

$$\Pi_0 \left\{ \begin{array}{c} p(a) \leftarrow \textbf{not } q(a). \\ p(b) \leftarrow \textbf{not } q(b). \\ p(a). \end{array} \right. \quad \Pi_1 \left\{ \begin{array}{c} p(a) \leftarrow \textbf{not } \neg p(a). \\ q(b) \leftarrow \textbf{not } \neg q(b). \\ \neg p(a). \end{array} \right. \quad \Pi_2 \left\{ \begin{array}{c} \{p, s, q\}2 \\ r \leftarrow p. \\ p \leftarrow q. \end{array} \right. \quad \Pi_3 \left\{ \begin{array}{c} p(X) \leftarrow r(X), \textbf{not } q(X). \\ q(X) \leftarrow r(X), \textbf{not } p(X). \\ r(a) \leftarrow . \\ r(b) \leftarrow . \\ \leftarrow p(a). \end{array} \right.$$

Exercice 2 Formalisme des "ensembles-réponse" (*Answer Sets*) – 4 points
Soit les 7 les propositions suivantes :

a) Le rugby, le criquet et le football sont des sports anglais.

b) L'escrime et la pétanque sont des sports.

c) Le rugby n'est pas ennuyeux.

d) Les sports anglais sont des sports.

e) Amusant et ennuyeux sont contradictoires.

f) En général, les sports sont amusants. *Autrement dit, les sports qui ne sont pas ennuyeux sont amusants.*

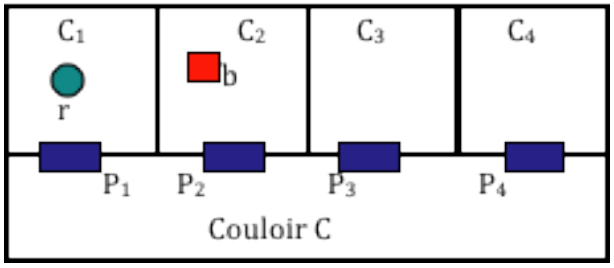
g) En général, les sports anglais sont ennuyeux. *Autrement dit, les sports anglais qui ne sont pas amusants sont ennuyeux.*

1. Représenter ces propositions dans le formalisme des ensembles-réponse (ASP) en faisant appel aux quatre prédicats suivants : $sport/1$, $sport\_anglais/1$, $amusant/1$, $ennuyeux/1$.
2. Donner tous les ensembles-réponse de ce programme.

Exercice 3 Jeux 5 points
La figure qui se trouve sur la page 4 de l'énoncé représente un arbre de jeux (arbre de Kühn) pour un jeu à somme nulle. On pourra répondre à certaines questions de cet exercice directement sur cette page que l'on joindra à la copie.

1. **Min-Max**
   (a) Sachant que les nombres associés aux feuilles de l'arbre indiquent les gains de A, c'est-à-dire les pertes de B, calculer la valeur des différents nœuds internes de l'arbre en faisant appel au principe du *Min-Max*.
   (b) Quel coup doit jouer le joueur A pour espérer minimiser ses pertes ?
1. **Alpha-beta**
   (a) Rappeler le principe de l'*alpha-beta*.
   (b) Appliquer l'algorithme *alpha-beta* à l'arbre de jeu précédent en procédant de gauche à droite.
   (c) Appliquer l'algorithme *alpha-beta* à l'arbre de jeu précédent en procédant de droite à gauche et comparer avec le résultat de la question précédente.

Exercice 4 'Planification – 7 points
Considérer le diagramme ci-après :



1. *Représenter*, dans le formalisme STRIPS, l'état du monde donné dans cette figure en faisant appel aux quatre prédicats suivants :
   a) $dans(x, y)$ qui indique qu'un objet x se trouve dans une pièce y,
   b) $connecte(P, X, Y)$ qui dit que la porte P fait communiquer la pièce X avec la pièce Y,
   c) $robot(X)$ qui précise que X est un robot et
   d) $boite(B)$ qui précise que B est une boite.
2. *Opérateurs* : écrire, en STRIPS, les opérateurs
   (a) $passer(R, Ca, Cb)$ qui fait passer le robot R de la pièce C_a à la pièce C_b s'il y a une porte entre les deux.
   (b) $pousser(R, B, Ca, Cb)$ grâce auquel le robot R pousse la boite B de C_a vers C_b si une porte connecte les deux pièces.
3. *Condition terminale* : donner la condition terminale qui spécifie que le robot r et la boite b se retrouvent tous deux dans la pièce C_4
4. Donner un *plan* qui permette de passer de l'état initial à un état qui satisfait la condition terminale.
   Remarque : on souhaite ici uniquement un plan, sans décrire l'ensemble des actions du planificateur qui permettent de l'atteindre.
5. Simuler le *fonctionnement du planificateur* STRIPS sur ce problème en décrivant précisément l'évolution de la *pile de buts* et de l'*état courant* au cours des quatre premières étapes.
   Remarque 1 : on demande les quatre premières étapes du planificateur et non les quatre premiers déclenchements de règles.
   Remarque 2 : comme il est dit dans l'énoncé, il faut décrire explicitement l'évolution de la *pile de buts* et de l'*état courant*.

Annexes

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

Algorithme de Smodels

L'algorithme utilisé dans `smodels`, un des premiers solveurs ASP performants, est un algorithme de recherche arborescente d'une structure similaire à DPLL (Déduction, Analyse, Choix). L'algorithme `clasp` qui a été utilisé en TME possède un mécanisme de propagation similaire, mais se rapproche plus d'une structure CDCL (avec analyse des conflits et retour arrière non synchrone).

L'algorithme de `smodels` est toutefois plus facile pour rechercher "manuellement" les answer sets d'un programme. Il comporte trois phases.

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.

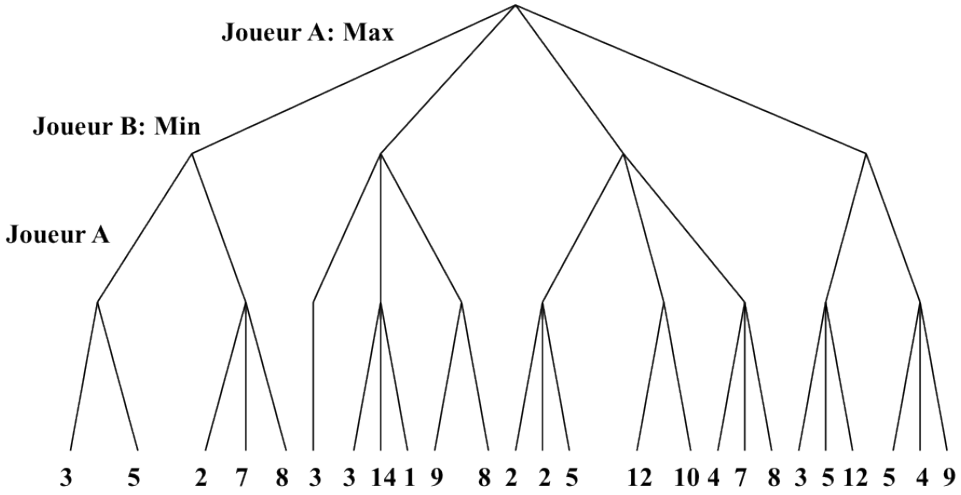
1. **Déduction.** On procède en deux temps.
   (a) Littéraux positifs. On ajoute à M_{pos} tous les atomes découlant de faits [1], soit $M_{pos} := M_{pos} \cup \{a \in Lit \setminus M_{pos} | 'a.' \in P'\}$. Pour chacun des atomes a ainsi rajouté à M_{pos}, on procède à la *mise à jour par a* du programme courant P', par suppression ou modification de règles, en traitant chaque règle $r \in P'$ de la façon suivante :
      — Si a est la tête de r, on peut supprimer la règle (elle est satisfaite car sa tête est satisfaite).
      — 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 (soit *not a* est dans le corps de r), on supprime la règle (elle est satisfaite car son corps n'est pas satisfait).
   (b) Littéraux négatifs. On ajoute à M_{not} tous les atomes qui ne sont plus présents dans aucune tête de règle de P', soit $M_{not} := M_{not} \cup \{a \in Lit \setminus M_{not} | \forall r \in P' a \notin head(r)\}$. Pour chacun des atomes a ainsi rajouté à M_{not}, on procède à la *mise à jour par not a* du programme courant P' en traitant chaque règle $r \in P$de la façon suivante :
      — Si a est dans le corps positif de r, on supprime la règle (elle est satisfaite car son corps n'est pas satisfait).
      — Si a est dans le corps négatif de r (soit *not a* est dans le corps 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 backtrack (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 backtrack 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*.

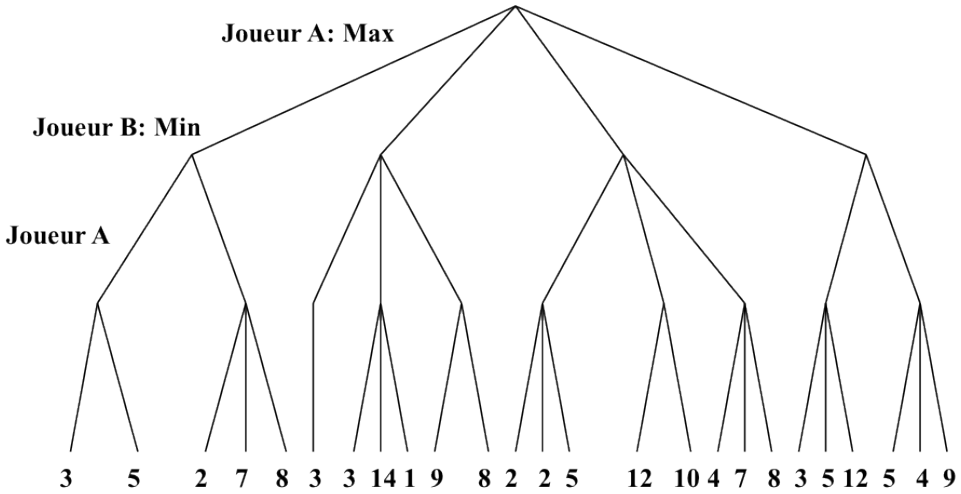
   On reprend alors à l'étape de déduction avec ce nouveau P'.

1. Cette phase est similaire à une propagation unitaire, les faits étant de même nature que des clauses unitaires.

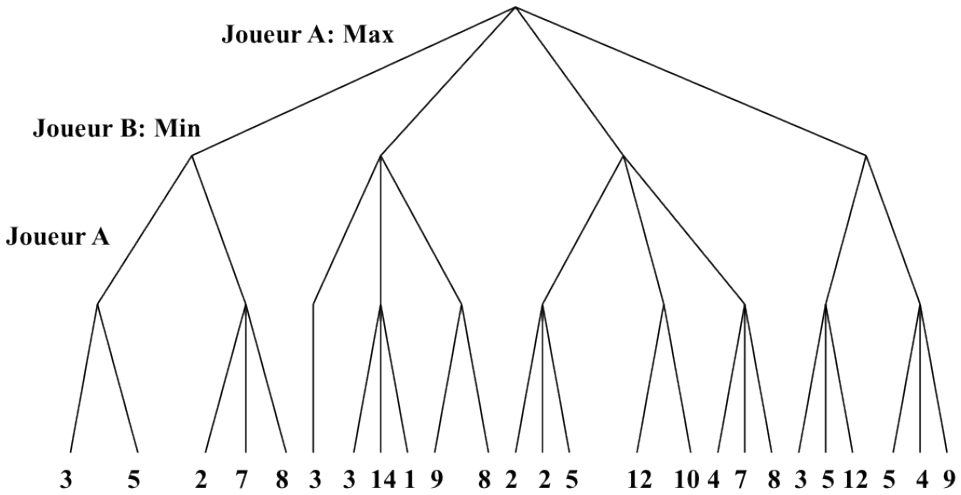
NOM : Prénom :



Valeur des différents nœuds



Alpha-Beta **de gauche à droite** $\longrightarrow$



Alpha-Beta **de droite à gauche** $\longleftarrow$