Université d'Orléans **Master Informatique**
Intelligence Artificielle Année 2010-2011

Contrôle final

Durée : 2h. Documents autorisés : notes de cours et de TD.

Exercice 1 : Planification On se penche sur le problème du départ en vacances. De nombreuses tâches préparatoires doivent être accomplies avant de tourner la clé de contact... Voici celles qui nous intéressent :

On remplit une bouteille d'eau au robinet, et on la charge dans la voiture. On charge les enfants, Jules et Jim, qui jouaient jusque là dans le jardin. Pour effectuer ces chargements, on gare la voiture devant la porte. Les niveaux doivent être contrôlés (quand le moteur est encore froid). Le plein d'essence doit être fait (attention, pour faire le plein la voiture doit être déplacée, donc le moteur n'est plus froid). La voiture est habituellement garée au garage. On n'oublie pas de couper l'eau avant de partir.

1. Représenter ce problème en langage STRIPS : état initial, buts, actions. On pourra avoir des prédicats et actions en propositionnel et en premier ordre.
2. Donner une planification en ordre partiel, sous forme graphique.
3. En déduire un ordre total (graphique)

Exercice 2 : Réseaux bayésiens Un vendeur de voitures a synthétisé son expérience sous forme d'un réseau bayésien. Ce réseau détermine les probabilités conjointes de l'âge du client, de l'âge de son véhicule actuel, du modèle et du type de moteur recherché, et enfin la probabilité d'achat. Ce réseau est représenté sur la figure 1.

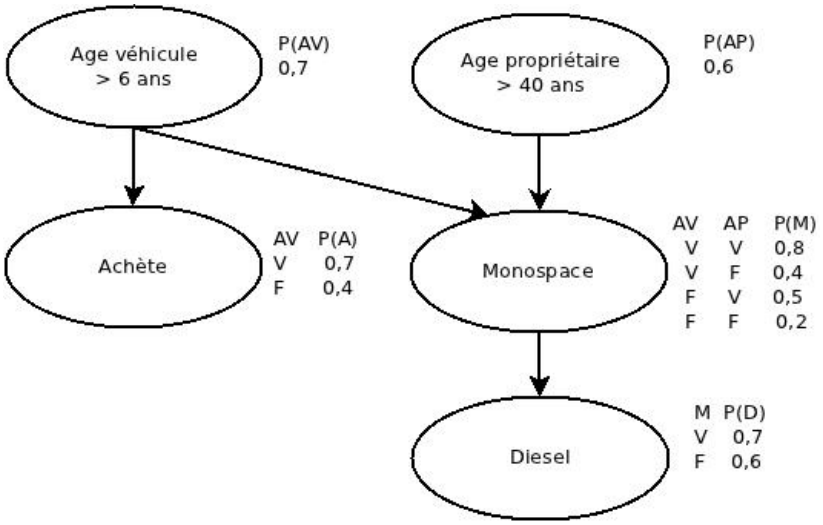

FIGURE 1 – Réseau bayésien du vendeur de voitures

1. Etablir la probabilité qu'un client de plus de 40 ans possèdant un véhicule âgé de plus de 6 ans s'intéresse à un monospace diesel mais ne l'achête pas.
2. Etablir la probabilité qu'un client qui s'intéresse à un monospace et qui a plus de 40 ans possède actuellement un véhicule de plus de 6 ans.
3. Le vendeur souhaite ajouter dans ce réseau bayésien le fait que le propriétaire ait des enfants mineurs. A quel endroit vous semble-t-il logique d'ajouter le noeud correspondant ?

Exercice 3 : Ensemble de clauses L'objectif est d'implanter un algorithme pour trouver une instanciation des variables pour satisfaire un ensemble de clauses. Chaque littéral est représenté par un terme `l(S,V)`, où est S est le signe p (positif) ou n (négatif) et V est le nom de la variable. Chaque clause est représentée par une liste de littéraux. L'ensemble de clauses est une liste de clauses. Par exemple pour représenter l'ensemble de clauses

$$\{a \vee \overline{b} \vee \overline{c},\ \overline{a} \vee b\}$$

nous avons la liste `[ [l(p,a), l(n,b), l(n,c)], [l(n,a), l(p,b)] ]`. Dans les questions suivantes, ajoutez des prédicats supplémentaires si besoin.

1. Ecrire un prédicat `eval_clause(+Var,+Val,+Clause,-NouvClause)`, qui prend en compte l'instanciation de la variable `Var` par la valeur `Val` (qui peut être `vrai` ou `faux`) pour calculer une nouvelle clause `NouvClause` à partir d'une clause `Clause`. La clause vide correspond à `faux` et la clause vraie est `[vrai]`. On remarquera que la nouvelle clause ne doit pas contenir la variable `Var`. Par exemple :

```
| ?- eval_clause(a,vrai,[l(n,a),l(p,b)],C).
C = [l(p,b)]
| ?- eval_clause(a,faux,[l(n,a),l(p,b)],C).
C = [vrai]
| ?- eval_clause(a,vrai,[l(n,a)],C).
C = []
```

2. Ecrire un prédicat `eval_ens(+Var,+Val,+Ens,-NouvEns)`, qui prend en compte l'instanciation de la variable `Var` par la valeur `Val` pour calculer le nouvel ensemble `NouvEns` à partir de l'ensemble de clauses `Ens`. Les ensembles ne contiennent pas de clause vide, car dès qu'une clause devient vide avec l'évaluation, elle est fausse, l'ensemble est donc insatisfiable. Ils ne contiennent pas non plus de clause `[vrai]`, car une telle clause sera supprimée de l'ensemble. L'ensemble vide correspond à `vrai`. Le prédicat rencontre un échec si l'ensemble `Ens` est insatisfiable. Par exemple :

```
| ?- eval_ens(a,vrai,[[l(n,a)],[l(p,b)]],E).
no
| ?- eval_ens(a,faux,[[l(n,a)],[l(p,b)]],E).
E = [[l(p,b)]]
| ?- eval_ens(a,vrai,[[l(p,b)],[l(n,a),l(n,c)]],E).
E = [[l(p,b)],[l(n,c)]]
```

3. Ecrire un prédicat `choisir_var(+Ens,-Var)` qui calcule une variable non instanciée de l'ensemble `Ens`. On se contente ici de prendre la première variable de la première clause de l'ensemble.

4. Ecrire un prédicat `algorithme(+Ens,-Solution)` qui prend en entrée un ensemble de clauses `Ens` et qui calcule pour `Solution` une liste d'instanciations de variables qui satisfait les clauses de `Ens`. Ce prédicat implantera l'algorithme suivant : on choisit une variable non instanciée et on l'instancie soit à vrai, soit à faux. L'instanciation de la variable permet d'évaluer l'ensemble `Ens` en un nouvel ensemble et l'algorithme continue avec le nouvel ensemble, jusqu'à ce que l'ensemble devienne vide ou que l'on rencontre un échec (l'ensemble est insatisfiable). Chaque instanciation est représentée par un terme `ins(Variable,Valeur)`. Par exemple :

```
| ?- algorithme([[l(p,a),l(n,b)], [l(n,a), l(p,b), l(p,c)]],L).
L = [ins(b,vrai),ins(a,vrai)] ? ;
L = [ins(c,vrai),ins(b,faux),ins(a,vrai)] ? ;
L = [ins(b,faux),ins(a,faux)] ? ;
no
| ?- algorithme([[l(p,a)], [l(n,a)]],L).
no
```

5. Modifier le prédicat `choisir_var` en un prédicat `choisir_var(+Ens,-Var,-Val)` pour implanter l'heuristique de clause unitaire. L'idée est que l'on cherche une instanciation d'une variable `Var` par une valeur `Val` pour satisfaire la première clause unitaire si une telle clause existe. Sinon le prédicat calcule une variable avec deux valeurs possibles. Par exemple :

```
| ?- choisir_var([[l(p,a),l(p,b)],[l(n,a)]],Var,Val).
Val = faux
Var = a
| ?- choisir_var([[l(n,a),l(p,b)]],Var,Val).
Val = vrai
Var = a ? ;

Val = faux
Var = a
```

   Modifier en conséquence le prédicat `algorithme` pour adapter au nouveau prédicat `choisir_var`.