Université de Sherbrooke
Département d'informatique

IFT615 – Intelligence artificielle
Examen final
Été 2010

Le mercredi 4 août 2010, 9 h à 12 h, au D4-2021

Chargé de cours

Éric Beaudry

Instructions

L'examen dure trois heures. L'examen comporte 7 questions pour un total de 40 points. Le questionnaire contient 9 pages incluant celle-ci. Vous devez répondre directement sur le questionnaire aux endroits indiqués. Une feuille de brouillon est incluse à la fin du questionnaire. Ne détachez aucune feuille du questionnaire à l'exception de l'annexe 1 et de la feuille de brouillon.

Matériel permis

- Livre de référence du cours (*Artificial Intelligence: A Modern Approach*).
- Deux feuilles recto-verso de notes personnelles.
- Calculatrice non programmable.
- Tout autre appareil électronique est strictement interdit.

Solutions

Q1 /5	Q2 /8	Q3 /4	Q4 /10	Q5 /5	Q6 /4	Q7 /4	TOTAL /40

Question 1 – Unification et composition de substitutions (5 points)

Dans les expressions qui suivent, ?u, ?x, ?y, ?z sont des variables; a, b, c sont des constances; f et g sont des symboles de fonction; p et q sont des symboles de prédicat.

a) Calculez un unificateur le plus général (upg) pour les expressions suivantes. S'il n'existe pas d'unificateur, écrivez «aucun». (3 points)

Prédicats	UPG	Prédicats	UPG
p(?x, a) p(f(?x), ?y)	Aucun	p(?x, ?y, ?z) p(a, ?x, ?z)	{?x=a, ?y=a}
p(?x, f(?x)) p(?z, f(a))	{?x=a, ?z=a}	p(?x) q(?x)	Aucun
p(f(?x, g(a, ?y)), ?z) p(?z, ?u)	{?z=f(?x, g(a, ?y)), ?u=f(?x, g(a, ?y)) }	p(?x) p(?x)	{}

b) Donnez la composition des substitutions suivantes. (2 points)

{?x=y, ?z=f(?y)} {?y=?x}	{?z=f(?x), ?y=x}
{?x=f(?y), ?z=?y} {?x=a, ?y=b, ?z=c}	{?x=f(b), ?z=b, ?y=b}
{?x=?y, ?u=b}{?y=?z}{?z=?x, ?y=a}	{?u=b, ?y=x, ?z=x}

Question 2 – Logique du premier ordre et résolution (8 points)

Soit le texte suivant :

« *Le ePhone est un téléphone mobile. Ali a toujours son ePhone avec lui. Pour fonctionner, un téléphone mobile doit se situer à l'intérieur d'une zone couverte par un réseau sans-fil. Ali se trouve à l'UdeS. L'UdeS est une université. Toutes les universités sont couvertes par un réseau sans-fil.* »

a) Exprimez ce texte en logique du premier ordre. (3 points)

1. téléphone(ePhone)

2. ∀x setrouve(Ali, x) → setrouve(ePhone, x)

3. ∀x,y téléphone(x) ∧ couverte(y) ∧ setrouve(x, y) → fonctionne(x)

4. setrouve(Ali, UdeS)

5. université(UdeS)

6. ∀x université(x) → converte(x)

b) Convertissez les formules obtenues en a) sous forme de clauses. Ne donnez pas les étapes de conversion. Numérotez les clauses afin d'y référer à la sous-question suivante. (3 points)

1. téléphone(ePhone)

2. ¬setrouve(Ali, x1) ∨ setrouve(ePhone, x1)

3. ¬téléphone(x2) ∨ ¬couverte(x3) ∨ ¬setrouve(x2, x3) ∨ fonctionne(x2)

4. setrouve(Ali, UdeS)

5. université(UdeS)

6. ¬université(x4) ∨ converte(x4)

c) Utilisez la preuve par résolution pour montrer que l'ePhone d'Ali fonctionne. (2 points)

Ajout de la négation de l'affirmation à prouver :

7. ¬fonctionne(ePhone)

Résolution

8. setrouve(ePhone, UdeS) // 2 et 4 {x1=UdeS}

9. couverte(UdeS) // 5 et 6 {x4=UdeS}

10. ¬couverte(x3) ∨ ¬setrouve(ePhone, x3) ∨ fonctionne(ePhone) // 1 et 3 {x2=ePhone}

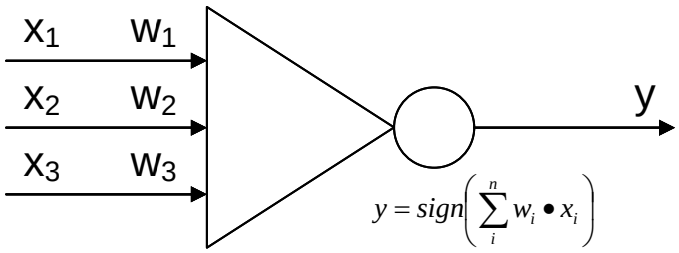
11. ¬setrouve(ePhone, UdeS) ∨ fonctionne(ePhone) // 9 et 10 {x3=UdeS}

12. fonctionne(ePhone) // 11 et 8 {}

13. Clause vide // {12 et 7} → On peut conclure que le téléphone d'Ali fonctionne.

Question 3 – Réseaux de neurones (4 points)

Soit, le neurone artificiel suivant et les données d'entraînement suivantes.



x_1	x_2	x_3	y
8	1	3	-1
8	4	7	1
5	4	7	-1

Simulez une itération complète de l'algorithme d'apprentissage du perceptron en utilisant les trois données d'entraînement. Le pas d'apprentissage est fixé à 0.1 et les poids initiaux *sont* w_1*=1,* w_2*=2* et w_3*=1*.

Data 1 : y=sign(1*8 + 2*1 + 1*3) = +1 → Δw = 0.1(-1-1)[8 1 3] = [-1.6 -0.2 -0.6] → w = [-0.6 1.8 0.4]

Data 2 : y=sign(-1.6*8+1.8*4+-0.4*7) = -1 → Δw = 0.1(1--1)[8 4 7] = [+1.6 +0.8 +1.4] → w = [+1 +2.6 +1.8]

Data 3 : y=sign(1*5+1.6*4+1.8*7) = +1 → Δw = 0.1(-1-1)[5 4 7] = [-1.0 -0.8 -1.4] → w = [0 1.8 0.4]

w_1=0, w_2=+1.8* et *w_3=+0.4

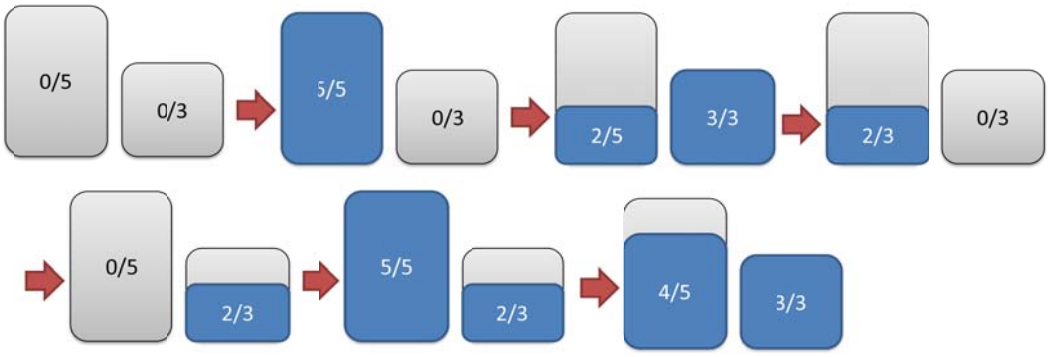
Question 4 – Recherche dans un espace d'états (10 points)

Soit le problème suivant : à l'aide de deux contenants de 3 et 5 litres, comment pouvez-vous mesurer très exactement 4 litres? Les trois types d'actions permises sont :

1. **remplir** un contenant à l'aide d'une source d'eau (ex : robinet);
2. **transvider** un contenant dans un autre jusqu'à temps que ce dernier soit plein ou que le contenant source soit vide;
3. **vider** (jeter) le contenu d'un contenant.

Une solution à ce problème classique est de procéder avec les étapes suivantes :

1. **remplir** le contenant de 5 litres;
2. **transvider** le contenant de 5 litres vers celui de 3 litres;
3. **vider** le contenant de 3 litres.
4. **transvider** le contenant de 5 litres vers celui de 3 litres;
5. **remplir** le contenant de 5 litres;
6. **transvider** le contenant de 5 litres vers celui de 3 litres;



Ainsi, le contenant de 5 litres contient très exactement 4 litres. Cette solution nécessite 10 litres d'eau provenant du robinet (étapes #1 et #5). Il est à noter que l'eau vidée à l'étape 3 est perdue puisqu'il n'existe pas un troisième contenant pouvant être utilisé pour stocker l'eau temporairement.

Ce problème peut être généralisé à **n** contenants ayant des capacités de C_1, …, C_n litres, et dont le but est de mesurer exactement **X** litres. Pour résoudre ce problème, on vous demande d'implémenter un résolveur basé sur un algorithme de recherche dans un espace d'états. Le pseudocode suivant est fourni.

```
Nombre n, X, C[1..n]; // constantes définies plus haut.
classe État {
   Nombre N[1..n]; // tableau exprimant le niveau actuel des n contenants.
}
classe Noeud {
   État état;
   Nombre f, g, h;
   Noeud parent;
   String action; // action qui a été exécuté depuis le noeud parent.
}
// Retourne un vecteur d'action.
Vecteur<String> A_Étoile(État e);
// Retourne vrai si le but est atteint.
Booléen But(État e);
// Estime le coût restant pour atteindre le but.
Nombre h(État e);
// Retourne un ensemble de noeuds sucesseurs.
Ensemble<Noeud> GénérerSucesseurs(Noeud n);
```

a) Complétez le pseudocode de la fonction suivante. (3 points) Note : les solutions acceptées peuvent varier.

```
Ensemble<Noeud> GénérerSucesseurs(Noeud n0){
  S ← {}
  Pour i = 1, ..., n
    m ← n0
    m.état.N[i] ← C[i]
    m.parent ← n0   // non requis: cela peut être fait par A_Étoile
    m.action ← « Remplir » + i
    m.g ← n0.g + C[i] - n0.état.N[i]
// Note 1 : si onveut minimiser la quantité, c'est le seul endroit où
// modifier g. Sinon, g=g+1 partout pour minimiser le nombre d'actions.
// Note 2 : f et h seront modifiés par A_Étoile
    ajouter m à S.

  Pour i = 1, ..., n
    m ← n0;
    m.état.N[i] ← 0
    m.parent ← n0   // non requis: cela peut être fait par A_Étoile
    m.action ← « Vider » + i
    ajouter m à S.

  Pour i = 1, ..., n
     Pour j = 1, ..., n
       Si i≠ j
          m ← n0;
          m.parent ← n0 // non requis: cela peut être fait par A_Étoile
          m.action ← « Transvider » + i « à » + j
          v ← min(C[j]-n0.état.N[j], n0.état.N[i])
          m.état.N[i] ← n0.état.N[i] - v
          m.état.N[j] ← n0.état.N[j] + v
          ajouter m à S.

  retourner S;
}
```

b) Complétez la fonction but(État). (1 point)

```
Booléen but(État e) {
 Pour i = 1, ..., n
   Si e.N[i]=X Alors Retourner VRAI
 Retourner FAUX
}
```

c) Supposons que les débits de la source d'eau, ainsi que du transvidage et du vidage des contenants, soient tous limités à un litre par minute. En raison du débit limité, on s'intéresse maintenant à trouver un plan qui minimise la durée du plan plutôt que de la quantité d'eau utilisée. Donnez les changements requis à la fonction successeur pour résoudre ce nouveau problème. (2 points)

Pour chaque noeud successeur généré, la valeur de l'attribut g doit être calculé en fonction du volume d'eau impliqué.

```
Cas #1 : m.g ← n0.g + (C[i] - n0.état.N[i]) * 1

Cas #2 : m.g ← n0.état.N[i] * 1

Cas #3 : m.g ← v * 1
```

d) Proposez une fonction heuristique admissible la plus efficace possible pour le problème en c). (2 points)

```
Nombre h(État e) {
H ← +infini
Pour i = 1, ..., n
   H ← min(H, abs(e.N[i] – X )
Retourner H * 1
}
```

e) Supposons qu'il soit maintenant possible d'effectuer plusieurs actions simultanément. Décrivez la principale modification requise afin de générer des plans pouvant contenir des actions simultanées. (2 points)

Pour planifier des actions simultanées, il faut simuler les effets projetés dans le temps (« delayed effects »). Pour ce faire, il faut modifier la classe nœud pour ajouter :

- un attribut current-time (ou heure, etc.)

- une file d'événements à venir.

L'algorithme de recherche applique les actions sans faire progresser le temps. Une action spéciale incrémente le temps.

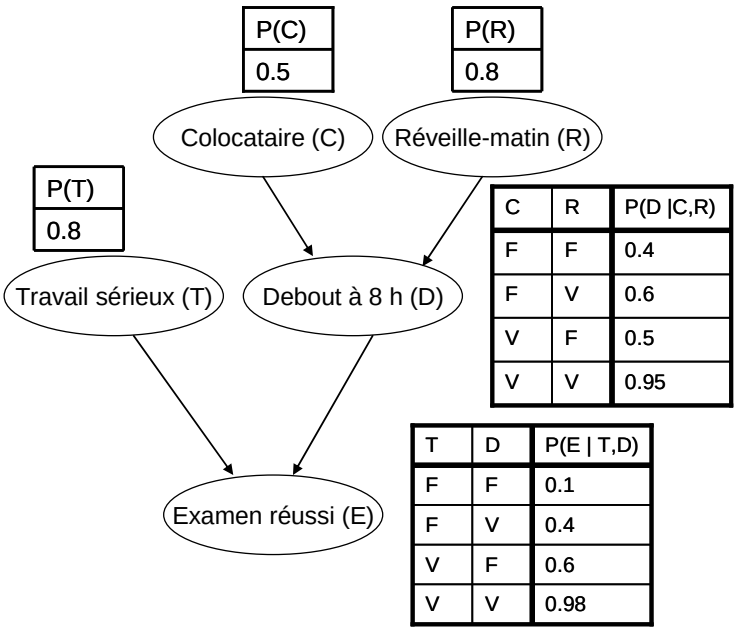
Question 5 – Planification et langage PDDL (5 points)

Écrivez le code PDDL pour résoudre le problème à la question 4c). Vous trouverez, en annexe à la fin du questionnaire, sur une feuille que vous pouvez détacher, un exemple de fichier PDDL pour la livraison de colis. Vous pouvez supposer l'existence des fonctions dont vous avez besoin, et ce, même si elles ne sont pas nécessairement dans le langage PDDL. La syntaxe exacte de PDDL ne sera pas évaluée.

```
(define (domain (MesurerXLitres)
  (:types contenant robinet) )
  (:fonctions
    (capacité ?c – contenant)
    (niveau ?c – contenant))

(:durative-action remplir
   (:parameters (?c – contenant))
   (:effects (= (niveau ?c) (capacité ?c)))
   (:duration (- (capacité ?c) (niveau ?c)))
)

(:durative-action vider
   (:parameters (?c – contenant))
   (:effects (= (niveau ?c) 0))
   (:duration (niveau ?c))
)

(:durative-action transvider
   (:parameters (?a – contenant   ?b - contenant) )
   (:effects (increase (niveau ?b)
       (min  (- (capacité ?b) (niveau ?b))  (niveau ?a))
   (:duration (min  (- (capacité ?b) (niveau ?b))  (niveau ?a))
)
```

Question 6 – Réseaux bayésiens (4 points)

Soit le réseau bayésien suivant dans lequel les nœuds représentent des variables aléatoires booléennes. Le réseau bayésien détaille des relations causales dans le contexte de la réussite d'un examen. La réussite d'un examen (E) dépend de l'effort de l'étudiant (T) et du fait qu'il puisse se lever à 8 h le jour de l'examen (D). Pour se lever tôt, l'étudiant a demandé à son colocataire de le réveiller (C). Comme ce dernier a tendance à faire souvent la fête, l'étudiant compte aussi sur son réveille-matin (R).



C	R	P(D |C,R)
F	F	0.4
F	V	0.6
V	F	0.5
V	V	0.95

T	D	P(E | T,D)
F	F	0.1
F	V	0.4
V	F	0.6
V	V	0.98

a) Écrivez le pseudocode d'un algorithme d'inférence approximatif, basé sur une méthode d'échantillonnage (Monte Carlo), pour **estimer** la probabilité P(E|C=true). Vous devez utiliser la fonction *random*() qui retourne un nombre aléatoire uniformément distribué dans l'intervalle [0, 1[. Le code demandé doit être spécifique au réseau bayésien ci-haut. (2 points)

```
réel Estimer_P(E|C=true) {

compte←0
Répéter N fois
  R ← random() < 0.8
  T ← random() < 0.8
  D ← random() < P(D|C=true, R)
  E ← random() < P(E|T,D)
  Si E Alors compte ← compte + 1
Retourner compte/N
}
```

b) Écrivez le pseudocode d'un algorithme approximatif pour estimer la probabilité P(T|E=false). (2 points)

```
réel Estimer_P(T|E=false) {

M ← 0
Compte ← 0
Répéter N fois
  C ← random() < 0.5
  R ← random() < 0.8
  T ← random() < 0.8
  D ← random() < P(D|C=true, R)
  E ← random() < P(E|T,D)
  Si non E Alors
    M ← M + 1;
    Si T Alors  compte ← compte + 1

Retourner compte/M
}
```

Annexe 1 – Exemple de langage PDDL

```
 (define (domain SimTransport)
  (:types location – object  robot – object   box - object)
  (:predicates
    (robot-at ?r - robot ?l - location)
    (box-at ?b - box ?l - location)
    (box-on ?b - box ?r - robot)
    (link ?x - location ?y - location) )
  (:functions
    (distance ?l1 - location ?l2 - location)
    (speed ?r - robot)
  )
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  (:durative-action Goto
    :parameters(?r - robot ?from - location ?to - location)
    :duration (= ?duration (/ (distance ?from ?to) (speed ?r)))
    :condition(and
          (at start (robot-at ?r ?from))
          (over all (link ?from ?to))
       )
    :effect(and
              (at start (not (robot-at ?r ?from)))
              (at end (robot-at ?r ?to))
     )
  )
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  (:durative-action Load
    :parameters(?r - robot ?loc - location ?b - box)
    :duration (= ?duration 60)
    :condition(and
          (over all (robot-at ?r ?loc))
          (at start (box-at ?b ?loc))
       )
    :effect(and
          (at start (not (box-at ?b ?loc)))
          (at end   (box-on ?b ?r))
     )
  )
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  (:durative-action Unload
    :parameters(?r - robot ?loc - location ?b - box)
    :duration (= ?duration 60)
    :condition(and
        (over all (robot-at ?r ?loc))
        (at start (box-on ?b ?r))
       )
    :effect(and
        (at end   (box-at ?b ?loc))
        (at start (not (box-on ?b ?r)))
     )
  )
)
```