Université de Batna
Faculté des sciences de l'ingénieur
Département d'informatique
Module : Modèles du parallélisme et les langages FDT

Durée : 1h30 Le 14/03/2010

Contrôle final
5ème année ingénieur

Exercice 01 (4 pts)

1. Quelles sont les conditions pour que deux automates à états finis soient équivalents ? [1 pt]
2. Remplissez le vide dans le théorème suivant [1 pt] :
   Théorème (Rabin-Scott):
   Si le langage **L** est accepté par un automate fini ……………………………………..........................
   alors **L** est accepté par un ……………………………………………………..……….............................
3. Dans un automate à nombre fini d'états, est ce que :
   a. L'ensemble des états finaux E_f peut être plein $E_f = E$? [0.25 pt]
   b. L'ensemble des états finaux E_f peut être vide $E_f = \varnothing$? [0.25 pt]
   c. L'ensemble des états E peut être vide ? [0.25 pt]
   d. La fonction de transitions δ est définie pour tous les couples formés d'un état et d'un symbole ? [0.25 pt]
4. Quelle est la différence entre une transition vivante et une transition quasi-vivante ? [1 pt]

Exercice 02 (4 pts)

Soit un système de commande de feu tricolore. Modélisez ce système à l'aide d'un automate d'états fini (formellement et graphiquement), sachant que :
Les informations d'entrées sont : Voiture-Présente (VP), Voiture-Absente (VA) et les couleurs du feu : Vert(V), Orange(O) ou Rouge(R).
Le système se comporte comme suit :

- Au départ le feu est rouge.
- Si le feu est rouge :
  - si une voiture est présente, le feu passe au vert ;
  - sinon le feu reste rouge.
- Si le feu est orange le feu passe au rouge.
- Si le feu est vert :
  - si une voiture est présente, le feu reste au vert ;
  - si une voiture est absente le feu passe à l'orange.
- Si le feu est orange :
  - si une voiture est présente, le feu passe au rouge;
  - si une voiture est absente, le feu passe au rouge.

Pour rendre l'automate fini, nous supposons que le système peut s'arrêté quand le feu passe au rouge.

Exercice 03 (12 pts)

Dans un protocole de communication, un ensemble de trames peuvent être transférées sur un réseau. Une ou plusieurs trames peuvent être encapsulées dans une autre trame.

Nous voulons modéliser ce protocole à l'aide d'un RdP prédicats-transitions. Pour cela nous définissons l'ensemble de clauses (de règles) suivants :

r_1 : contient(a, b) $\leftarrow$
r_2 : contient(b, c) $\leftarrow$
r_3 : fait_partie_de(y, x) $\leftarrow$ contient (x, y)
r_4 : fait_partie_de (z, x) $\leftarrow$ contient (x, y), fait_partie_de (z, y)
r_5 : $\leftarrow$ fait_partie_de (c, x)

tel que : la règle r_1 signifie que « ***a*** encapsule ***b*** »
r_2 signifie que « ***b*** encapsule ***c*** »
r_3 signifie que « si ***x*** encapsule ***y*** alors ***y*** fait partie de ***x*** »
r_4 signifie que « ***x*** encapsule ***y*** et ***z*** fait partie de ***y*** alors ***z*** fait partie de ***x*** »
r_5 signifie que « ***c*** fait partie de ***x*** »

Donnez un RdP prédicats-transitions qui modélise ce protocole (définition formelle et graphique).

Bonne chance…

NB: Le corrigé type vous le trouverez sur le site: Iguezouli.110mb.com

Contrôle final
$5^{ème}$ année ingénieur

Exercice 01 (4 pts)

5. Quelles sont les conditions pour que deux automates à états finis soient équivalents ? [1 pt]
6. Remplissez le vide dans le théorème suivant [1 pt] :
   Théorème (Rabin-Scott):
   Si le langage **L** est accepté par un automate fini ……………………………………..
   alors **L** est accepté par un ……………………………………………………..………..
7. Dans un automate à nombre fini d'états, est ce que :
   e. L'ensemble des états finaux E_f peut être plein $E_f = E$? [0.25 pt]
   f. L'ensemble des états finaux E_f peut être vide $E_f = \varnothing$? [0.25 pt]
   g. L'ensemble des états E peut être vide ? [0.25 pt]
   h. La fonction de transitions δ est définie pour tous les couples formés d'un état et d'un symbole ? [0.25 pt]
8. Quelle est la différence entre une transition vivante et une transition quasi-vivante ? [1 pt]

Correction :

1. Quelles sont les conditions pour que deux automates à états finis soient équivalents ? [1 pt]
   Deux automates A_1 et A_2 sont équivalents s'ils acceptent le même langage, c'est-à-dire si $L(A_1) = L(A_2)$.
2. Remplissez le vide dans le théorème suivant [1 pt] :
   Théorème (Rabin-Scott):
   Si le langage L est accepté par un automate fini non déterministe alors L est accepté par un automate fini déterministe.
3. Dans un automate à nombre fini d'états, est ce que :
   a. Oui E_f peut être plein $E_f = E$ [1 pt] [0.25 pt]
   b. Oui E_f peut être vide $E_f = \varnothing$ [0.25 pt]
   c. Non, E n'est jamais vide car il existe toujours l'état initial [0.25 pt]
   d. Non, δ est une fonction partielle parce qu'elle peut ne pas être définie pour tous les couples formés d'un état et d'un symbole [0.25 pt]
4. Quelle est la différence entre une transition vivante et une transition quasi-vivante ? [1 pt]
   Une transition est dite quasi-vivante pour un marquage M s'il existe un marquage M' accessible à partir de M permettant de la franchir, alors qu'une transition est dite vivante si elle est quasi-vivante pour tout marquage accessible à partir de M_0.

Exercice 02 (4 pts)

Soit un système de commande de feu tricolore. Modélisez ce système à l'aide d'un automate d'états finis (formellement et graphiquement), sachant que :
Les informations d'entrées sont : voiture-présente (VP), voiture-absente (VA) et les couleurs du feu : Vert(V), Orange(O) ou Rouge(R).
Le système se comporte comme suit :

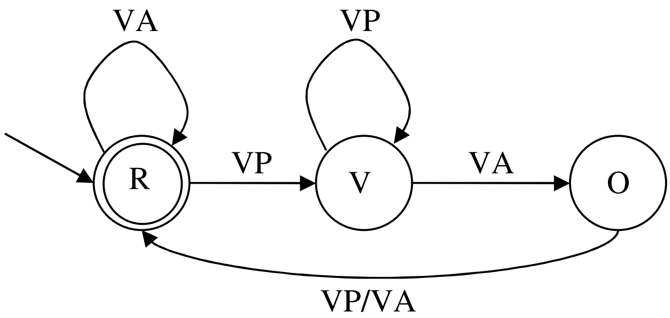
- Au départ le feu est rouge.
- Si le feu est rouge :
  - si une voiture est présente, le feu passe au vert ;

    - sinon le feu reste rouge.
- Si le feu est orange le feu passe au rouge.
- Si le feu est vert :
    - si une voiture est présente, le feu reste au vert ;
    - si une voiture est absente le feu passe à l'orange.
- Si le feu est orange :
    - si une voiture est présente, le feu passe au rouge;
    - si une voiture est absente, le feu passe au rouge.

Pour rendre l'automate fini, nous supposons que le système peut s'arrêté quand le feu passe au rouge.

Correction

A = <E, ∑ , e_0, E_f, δ > [0.5 pt]
Tel que :
E = {V, R, O} [0.5 pt]
∑ = {VA, VP} [0.5 pt]
e_0 = R [0.5 pt]
E_f = {R} [0.5 pt]



[1.5 pt]

Exercice 03 (12 pts)

Dans un protocole de communication, un ensemble de trames peuvent être transférées sur un réseau. Une ou plusieurs trames peuvent être encapsulées dans une autre trame.

Nous voulons modéliser ce protocole à l'aide d'un RdP prédicats-transitions. Pour cela nous définissons l'ensemble de clauses (de règles) suivants :

r_1 : contient(a, b) ←
r_2 : contient(b, c) ←
r_3 : fait_partie_de(y, x) ← contient (x, y)
r_4 : fait_partie_de (z, x) ← contient (x, y), fait_partie_de (z, y)
r_5 : ← fait_partie_de (c, x)

La règle r_1 signifie que « ***a*** encapsule ***b*** »
r_2 signifie que « ***b*** encapsule ***c*** »
r_3 signifie que « si ***x*** encapsule ***y*** alors ***y*** fait partie de ***x*** »
r_4 signifie que « ***x*** encapsule ***y*** et ***z*** fait partie de ***y*** alors ***z*** fait partie de ***x*** »
r_5 signifie que « ***c*** fait partie de ***x*** »

Donnez un RdP prédicats-transitions qui modélise ce protocole (définition formelle et graphique).

Correction

$R_{pt} = <R, A_n, M_0>$ [0.5 pt]

Tel que :

$R = <P, T, Pré, Post>$ Réseau de Petri ordinaire sous-jacent de R_{pt}. [0.5 pt]

P = {contient, fait_partie_de}

$T = \{t_1, t_2, t_3, t_4, t_5\}$

A_n: L'annotation de R, $A_n = <C_{onst}, V, A_{tc}, A_{ta}, A_c>$ [0.5 pt]

$C_{onst}= \{a, b, c\}$ [0.5 pt]

$V = \{x, y, z\}$ [0.5 pt]

On ajoute aux transitions d'entrée et de sortie du réseau t_1, t_2 et t_5 les conditions suivantes:

$A_{tc}(t_1) = (\boldsymbol{x}==\boldsymbol{a}) \wedge (\boldsymbol{y}==\boldsymbol{b})$ [1 pt]

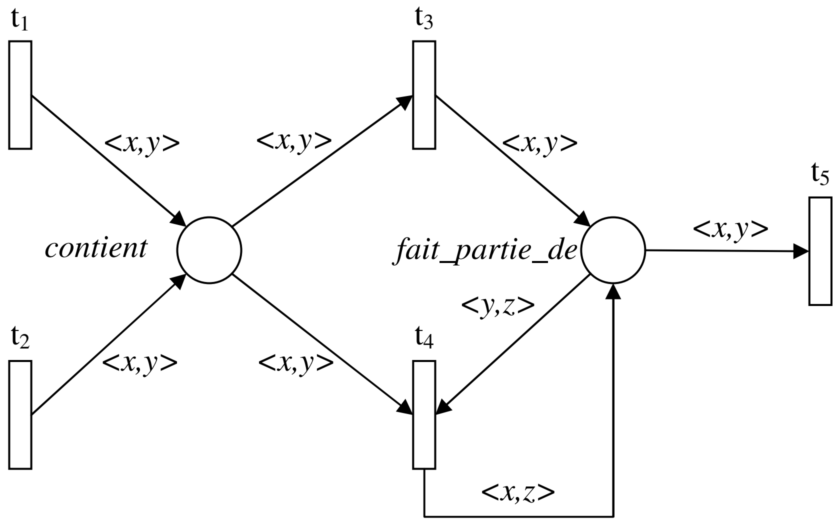
$A_{tc}(t_2) = (\boldsymbol{x}==\boldsymbol{b}) \wedge (\boldsymbol{y}==\boldsymbol{c})$ [1 pt]

$A_{tc}(t_5) = (\boldsymbol{y}==\boldsymbol{c})$ [1 pt]

On n'a pas besoin d'actions (A_{ta})

$$A_c = \begin{array}{r} \\ contient \\ fait\_partie\_de \end{array} \begin{array}{c} \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \end{array} \\ \begin{bmatrix} <x,y> & <x,y> & -<x,y> & -<x,y> & 0 \\ 0 & 0 & <x,y> & <x,z> - <y,z> & -<x,y> \end{bmatrix} \end{array}$$ [2.5 pt]

M_0: est le marquage initial.



[4 pt]