POLYTECH ANNECY-CHAMBÉRY

INFO 631

TP3 Automates

Exercice 1- Télécharger le fichier Automate.pl

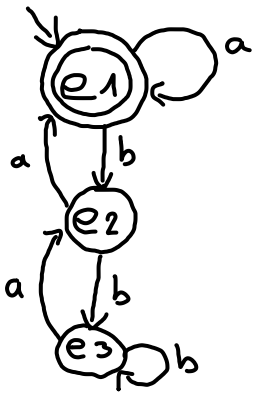
Un automate est caractérisé par un ensemble d'états et un ensemble de transitions étiquetées permettant de passer d'un état à un autre. Parmi les états possibles de l'automate, certains sont définis comme étant des états de départ autorisés (états initiaux). De même, certains états correspondent à des états d'arrivée autorisés (états finaux).

Les faits donnés ci-dessous définissent un automate à trois états (e1, e2 et e3) et à six transitions. Le fait transition(e1, b, e2) signifie que si l'automate est dans l'état e1 et reçoit le symbole b alors il passe dans l'état e2. L'état e1 est à la fois un état initial et un état final.

```
etat(e1).
etat(e2).
etat(e3).

initial(e1).
final(e1).

transition(e1, a, e1).
transition(e1, b, e2).
transition(e2, a, e1).
transition(e2, b, e3).
transition(e3, a, e2).
transition(e3, b, e3).
```



1.1 Donner une représentation graphique de l'automate défini ci-dessus où chaque transition est visualisée par une flèche étiquetée par le symbole qui permet de la franchir.

1.2 Le prédicat reconnu(Mot) réussit si la liste de symboles Mot est reconnue par l'automate, c'est-à-dire si la succession de transitions définies par Mot représente un parcours de l'état initial à un état final.

```
reconnu(Mot) :- initial(EI),
                parcours(Mot, EI, EF),
                final(EF).
```

Ecrire le prédicat parcours et tester le bon fonctionnement de reconnu.

1.3 Ecrire le prédicat parcours_ch(Mot, EI, EF, Chemin) qui fournit via l'argument Chemin la liste des états parcourus pour reconnaître Mot. Il sera utilisé par le prédicat reconnu_ch défini comme suit :

```
reconnu_ch(Mot, CH) :- initial(EI),
                       parcours_ch(Mot, EI, EF, CH),
                       final(EF).
```

Par exemple, à la question ?- reconnu_ch([b,a], Ch).
le programme fournira la réponse Ch = [e1, e2, e1].

De même, à la question ?- reconnu_ch(Mot, [e1, e2, e1]).
la réponse donnée est Mot = [b, a].

Quant à la question ?- reconnu_ch(Mot,[e2,e1]).
elle conduit à la réponse No.
du fait que l'état initial n'est pas correct.

Par contre, la question ?- parcours_ch(Mot, EI, EF, [e2,e1]).
fournit la réponse Mot = [a]
EI = e2
EF = e1

1.4 Qu'obtient-on comme réponse(s) aux questions ?- parcours_ch(Mot, e1, e3, Ch).
?- parcours_ch(Mot, e3, e1, Ch).

Expliquer pourquoi.

1.5 Ecrire un prédicat existe_parcours(ED, EA, LInterdit) qui réussit s'il existe un parcours de l'état de départ ED jusqu'à l'état d'arrivée EA ne passant pas par un état interdit, c'est-à-dire un état appartenant à la liste instanciée des états interdits LInterdit.

Pour éviter que le programme ne boucle indéfiniment, les états auxquels on a déjà accédé deviendront interdits de façon à ne pas y retourner.

Ainsi, le prédicat chemin(ED, EA) défini par :

```
chemin(ED, EA) :- existe_parcours(ED, EA, []).
```

devra réussir s'il est possible de rejoindre l'état EA en partant de l'état ED.

Tester vos prédicats sur les questions ?- chemin(e1, e3).
?- chemin(e3, e1).
?- existe_parcours(e3, e1, [e2]).

Enfin, ajouter deux nouveaux états e4 et e5 et les transitions suivantes :

```
transition(e3, c, e4).
transition(e4, a, e5).
transition(e4, b, e3).
transition(e5, a, e5).
```

et tester vos prédicats sur les questions ?- chemin(e1, e5).
?- chemin(e5, e1).
?- existe_parcours(e4, e2, [e5, e1]).

1.6 Le prédicat prédéfini setof(X, predicat(..., X, ...), LX) permet de regrouper dans la liste LX ordonnée toutes les valeurs distinctes de X rendant vrai le predicat spécifié.

Quelle est la réponse obtenue à la question ?- setof(X, chemin(e1,X), LX).
De même pour la question ?- setof(X, chemin(X,e1), LX).

Ecrire le prédicat est_accessible(E) qui est vrai si l'état E est accessible en partant de n'importe quel état de l'automate.

Ecrire le prédicat permet_acces(E) qui est vrai si en partant de E on peut accèder à tous les autres états de l'automate.

Exercice 2

Dans l'exercice **1**, la façon de définir un automate ne permet pas d'en gérer plusieurs simultanément. Pour palier cet inconvénient, on définit un prédicat automate d'arité 5, automate(NA, LE, LI, LT, LF), avec :

NA : nom de l'automate,
LE : liste des états,
LI : liste des états initiaux,
LT : liste des transitions,
LF : liste des états finaux.

Ainsi, les faits :

```
automate( aut1,
          [e1, e2, e3],
          [e1],
          [ [e1, a, e1], [e1, b, e2], [e2, a, e1], [e2, b, e3], [e3, a, e2], [e3, b ,e3] ],
          [e1] ).

automate( aut2,
          [e1, e2, e3, e4, e5],
          [e1, e4],
          [ [e1, a, e1], [e1, b, e2], [e2, a, e1], [e2, b, e3], [e3, a, e2], [e3, b ,e3],
            [e3, c, e4],  [e4, a, e5], [e4, b, e3], [e5, a, e5] ],
          [e1, e2] ).
```

définissent deux automates distincts. Le premier de nom aut1 est celui du début de l'exercice **1**. Le second de nom aut2 correspond à la version modifiée de aut1 introduite à la question **1.5** avec l'ajout de nouveaux états et de nouvelles transitions, étendue par l'ajout d'un nouvel état initial et final.

2.1 Definir les automates aut1 et aut2 dans votre programme et représenter graphiquement l'automate aut2.

2.2 Ecrire les prédicats etat, initial, transition et final qui permettent de gérer des listes d'états et de transitions. Ces prédicats sont les équivalents des prédicats de même nom de l'exercice **1** avec l'ajout d'une liste en 1^{ier} argument. Par exemple, le prédicat initial d'arité 2, initial (LI, EI), est satisfait pour tout état initial EI de la liste LI.

2.3 Ecrire les versions avec automate des prédicats reconnu et reconnu_ch, similaires à leurs homologues de **1.2** et **1.3**, mais appliqués à l'automate de nom Aut donné en 1^{ier} argument. A la question ?- reconnu_ch(aut1, [a, b, a], Ch), le programme fournira donc la solution *Ch* = [e1, e1, e2, e1] avant de répondre Yes. On prendra soin d'écrire les versions avec liste ou automate des prédicats qui le nécessitent.

2.4 Réécrire le prédicat chemin(ED, EA) de **1.5** qui devient chemin(Aut, ED, EA).

2.5 Réécrire les prédicats de **1.6**.