جامعة العلوم و التكنولوجيا و الطب
Université des Sciences de Technologie et de Médecine
USTM

Min Projet : INFO S3 2017

Date : 11 janvier 2017

Par D.Sidi EHMETY

Réalisé par:

-Abdallah Mohamed lemine Haiballa C09801

-Mohamed Ahid Issa C08555

-Mohamed Boullah Mohamed C09751

Sujet :💬

Ce mini projet s’intéresse à la représentation des automates fini en Ocaml.

AUTOMATE

DEFINITION:

Un automate fini ou automate avec un nombre fini d'états

est un modèle mathématique de calcul,

utilisé dans de nombreuses circonstances, allant de la conception de programmes informatiques et de circuits en logique séquentielle aux applications dans des protocoles de communication, le contrôle processus.

pour représenter les automates finis en Ocaml, on a fait la structure suivante :

```
type automate = {
   etat_initial : int ;
   ensemble_des_etats : int list;
   alphabets : char list;
   transitions : (int*char*int) list;
   etats_finaux : int list;
};;
```

SAISIR UNE AUTOMATE

```
let ei = let ()= print_string "l'etat_initial de l'automate\n"
in let () = print_string "donnez l'etat initial: " in let
n=read_int ()
in n;;

let ne =let ()= print_string "les etats de l'automate \n" in
let ()= print_string "nombre des etats: " in let n=read_int ()
in let rec r a = match a with
0 -> []
|_ -> let () = print_string "donnez un etat: " in let
t=read_int () in t::r (a-1) in
r n;;

let ef = let ()= print_string "les etat_finaux de
l'automate\n" in
let ()= print_string "nombre d'etats_finaux: " in let
n=read_int () in
let rec r a = match a with
0 -> []
|_ -> let () = print_string "donnez un etat_final: " in let
t=read_int () in t::r (a-1) in
r n;;
```

```
let al = let ()= print_string "les alphabets de l'automate \n"
in
let ()= print_string "nombre des alphabets: " in let
n=read_int () in
let rec r a = match a with
0 -> []
|_ -> let () = print_string ("donnez un alphabets: ") in let
t=read_line() in let k= String.get t 0 in k::r (a-1) in
r n;;

let tr =
 let () = print_string "les transitions de l'automate \n" in
 let () = print_string "nombre des transitions: " in
 let n=read_int () in
let rec g a=
match a with
0 -> []
|_ -> let ()= print_string "\ntransition: \n"
in let ()=print_string "Donnez l'Etat de départ: "
in let e= read_int()
in let ()= print_string "Donnez l'alphabet: "
in let w=read_line() in let k= String.get w 0
in  let ()= print_string "Donnez l'Etat drivée: " in
let d=read_int () in (e,k,d)::(g (a-1)); in g n;;
```

SAISIR UNE AUTOMATE

```
let a1={
        etat_initial =ei;
        ensemble_des_etats =ne ;
        alphabets =al;
        transitions =tr;
        etats_finaux =ef;
};;
```

AUTOMATE VALIDER

Valide:

C'est la fonction qui dit est ce que l'automate est accepté ou non.

```
let valide auto=
let rec separer ts =
match ts with
[] -> ([],[],[])
| (x,y,z)::r -> let (xs, ys, zs) = (separer r) in (x::xs, y::ys, z::zs) ;in
let (dom, sym, img) = separer auto.transitions ;in
let rec member_element x l = match l with
[] -> false
|(t::r) -> if x=t then true else member_element x r ;in
let rec membre_liste l l' =
match (l, l') with
([], _) -> true
| (x::xs, _) ->
(member_element x l') && (membre_liste xs l') ;in
 if (membre_liste dom auto.ensemble_des_etats =true &&
membre_liste sym auto.alphabets = true &&
membre_liste img auto.ensemble_des_etats = true &&
membre_liste auto.etats_finaux auto.ensemble_des_etats
=true && member_element auto.etat_initial
auto.ensemble_des_etats=true) then true else false;;
```

les fonctions

1) Fonction « complet »

```
let complet auto =
let rec img e a t=
  match t with
     [] -> []
    |x::r -> let (i,j,k) = x  in
         if (i=e && j=a) then k::(img e a r) else img e a r; in
let rec nodef e w t =
  match w with
     [] -> []
    |a::r -> if (img e a t) = [] then (e,a)::(nodef e r t) else
nodef e r t;in
let rec complet_cond e w t =
  match e with
     [] -> []
    |e1::en -> (nodef e1 w t)@(complet_cond en w t); in
if ((valide auto=true)&&((complet_cond
auto.ensemble_des_etats auto.alphabets
auto.transitions)=[])) then true else false;;
```

2) Fonction « deterministe »

```
let deterministe auto=
let h a =let rec membre a l = match l with
[]-> false
|x::r -> a=x||membre a r; in
let rec g a = match a with
[] -> false
|x::r ->(membre x r)||(g r); in
let rec f a = match a with
[] -> []
|(x,y,z)::r -> (x,y)::(f r); in
let l = (f a) in (g l) ;in
let a=auto.transitions in
let rec s l= let rec m (x,y,z) l= match l with
[]->false
|(a,b,c)::r -> ((x,y,z)=(a,b,c))||(m (x,y,z) r); in match l with
[] -> []
|(x,y,z)::r -> if (m (x,y,z) r) then (s r) else (x,y,z)::s r; in
let v =s a in if h v then false else true;;
```

les fonctions

3) Fonction « dessiner_automate»

```
let dessiner_automate auto=
let cet auto =
let rec coor l = match l with
[] -> []
|x::r -> let n=List.length l in if (n mod 2=0)then let a=((n*100)+100) and  b=200 in let c=(x,(a,b)) in c::(coor r)
                                        else let a=((n*100)+100) and  b=100 in let c=(x,(a,b)) in c::(coor r);in
coor auto.ensemble_des_etats; in
let ctr auto=
let rec bring x l = match l with
[] -> (0,(0,0))
|(c,(a,b))::r -> if c=x then (c,(a,b)) else bring x r; in
let rec cooetrans coreta transitions=match transitions with
[] -> []
|(x,y,z)::r -> let a=bring x coreta and b=bring z coreta in (a,y,b)::cooetrans coreta r ;in
cooetrans (cet auto) auto.transitions;
```

les fonctions

3) Fonction « dessiner_automate» suite

```
in

let etat (c,(x,y))=draw_circle x y 10 ;let a=(x-5) and b=(y-5) in moveto a b ;let s=string_of_int c in draw_string s;in
let etati (c,(x,y))=draw_circle x y 15 ;draw_circle x y 10 ;let a=(x-5) and b=(y-5) in moveto a b ;let s=string_of_int c in
draw_string s;
in
let flech (c,(x,y))=let a= (x-8) and b=(y+30) in moveto a b;
let d=x and h=(y+15) in lineto d h;
let r=(x+2) and z=(y+15) in lineto r z;
let s=(x+8) and w=(y+30) in lineto s w;let a=(x-8) and b=(y+30) in lineto a b;
in
let bon ((i,(xi,yi)),p,(j,(xj,yj)))=let xc=xi and yc=yi+44 in moveto xc yc; draw_char p;
let xa=xi and ya=yi+20 in draw_arc xa ya 17 25 320 220;
let xv=xi-20 and yv=yi+5 in moveto xv yv ;draw_char 'V';
in
```

les fonctions

3) Fonction « dessiner_automate» suite

```
let arrow ((i,(xi,yi)),p,(j,(xj,yj))) =
if ((xi<xj)&&(yi=yj)) then(let x1=xi+15 and y1=yi+5 in moveto x1 y1;let x2=xj-15 and y2=yj+5 in lineto x2 y2;let a=(x2-8) and
b=(y2-8) in lineto a b;let c=(y2+8) in moveto a c;lineto x2 y2;let i=((x1+x2)/2) and j=((y1+y2+5)/2) in moveto i j ;draw_char
p;)
else if ((xi>xj)&&(yi=yj)) then (let x1=xi-15 and y1=yi-5 in moveto x1 y1;let x2=xj+15 and y2=(yj-5) in lineto x2 y2;let
a=(x2+8) and b=(y2+8) in lineto a b;let c=(y2-8) in moveto a c;lineto x2 y2;let i=((x1+x2)/2) and j=((y1+y2-22)/2) in
moveto i j ;draw_char p;)
else if ((xi=xj)&&(yi<yj)) then (let x1=xi-5 and y1=(yi+15) in moveto x1 y1;let x2=xj-5 and y2=(yj-15) in lineto x2 y2;let
a=(x2+8) and b=(y2-8) in lineto a b;let c=(x2-8) and d=(y2-8) in moveto c d;lineto x2 y2;let i=((x1+x2-11)/2) and
j=((y1+y2)/2) in moveto i j ;draw_char p;)
else if ((xi=xj)&&(yi>yj)) then (let x1=xi+5 and y1=(yi-15) in moveto x1 y1;let x2=xj+5 and y2=(yj+15) in lineto x2 y2;let
a=(x2+8) and b=(y2+8) in lineto a b;let c=(x2-8) and d=(y2+8) in moveto c d;lineto x2 y2;let i=((x1+x2+11)/2) and
j=((y1+y2)/2) in moveto i j ;draw_char p;)
else if ((xi<xj)&&(yi<yj)) then (let x1=xi+5 and y1=(yi+15) in moveto x1 y1;let x2=xj-15 and y2=(yj-5) in lineto x2 y2;let
a=(y2-8) in lineto x2 a;let c=(x2-8) in moveto c y2;lineto x2 y2;let i=((x1+x2-5)/2) and j=((y1+y2+5)/2) in moveto i j
;draw_char p;)
else if ((xi>xj)&&(yi>yj)) then (let x1=xi-5 and y1=(yi-15) in moveto x1 y1;let x2=xj+15 and y2=(yj+5) in lineto x2 y2;let
a=(x2+8) in lineto a y2;let c=(y2+8) in moveto x2 c;lineto x2 y2;let i=((x1+x2+5)/2) and j=((y1+y2-5)/2) in moveto i j
;draw_char p;)
else if ((xi<xj)&&(yi>yj)) then (let x1=xi+15 and y1=(yi-5) in moveto x1 y1;let x2=xj-5 and y2=(yj+15) in lineto x2 y2;let
a=(x2-8) in lineto a y2;let c=(y2+8) in moveto x2 c;lineto x2 y2;let i=((x1+x2+5)/2) and j=((y1+y2+5)/2) in moveto i j
;draw_char p;)
else (let x1=xi-15 and y1=(yi+5) in moveto x1 y1;let x2=xj+5 and y2=(yj-15) in lineto x2 y2;let a=(x2+8) in lineto a y2;let
c=(y2-8) in moveto x2 c;lineto x2 y2;let i=((x1+x2-5)/2) and j=((y1+y2-5)/2) in moveto i j ;draw_char p;);in
```

les fonctions

3) Fonction « dessiner_automate» suite

```
let fin l (c,(a,b))=let rec membre a l =match l with
| [] -> false
| x::rl -> x=a || membre a rl;in if membre c l then true else false;
in
let fik i (c,(a,b))=if i=c then true else false;
in
let draw_atra auto =
let rec drawtr l k h= match l with
[] ->moveto 100 100;
|(x,y,z)::r -> if fin k x then etati x else etat x ;if fik h x then flech x else etat x;
 if x=z then bon (x,y,z) else arrow (x,y,z);if fin k z then etati z else etat z ;if fik h x then flech x else etat x;drawtr r k h;
 in
drawtr (ctr auto) auto.etats_finaux auto.etat_initial;in
draw_atra auto ;;
```

FIN