

Première partie

Analyse de billet de concert

Exercice 1 : Traitement d'une commande de billets

Spécification de Concert.1.lex : reconnaissance des champs clefs

```
1  alpha  [a-zA-Z]
2  codeConcert  ^T[0-9]{2,6}
3  codeDossier  [0-9]{8}
4  date  [0-9][0-9]\\/[0-9]{1,2}(\\/[0-9][0-9])?
5  FL  (\\n)
6  heure  [0-9][0-9]:[0-9][0-9]
7  ignore  [\\t ]+
8  nbPlaces  [1-9]?[0-9](\\ places)
9  nomConcert  [A-Z0-9]([A-Z0-9]|(-[A-Z0-9]))*
10 nomPropre  {alpha}({alpha}|(-{alpha}))*
11 prenomNom  ^{nomPropre}\\/{nomPropre}
12
13
14 %%
15 {codeConcert}      {printf("codeConcert");}
16 {codeDossier}      {printf("codeDossier");}
17 {date}             {printf("date");}
18 DOSSIER             {printf("dossier");}
19 <<EOF>>             {printf("FinFichier\\n"); return 0;}
20 {FL}               {printf(" FL\\n");}
21 {heure}            {printf("heure");}
22 {ignore}           {printf(" ");}
23 {nbPlaces}         {printf("nb places");}
24 {nomConcert}       {printf("nomConcert");}
25 {prenomNom}       {printf("prenomNom");}
```

Listing 1 – Première spécification en vue d'un test de reconnaissance des différents champs d'une commande de billets

Spécification de Concert.2.lex : application de la reconnaissance à un besoin 'réel'

```

1  %{
2      char* codeDossier;
3      char* prenomNom;
4      int nbPlaces = 0;
5      int nbConcerts = 0;
6  %}
7
8  alpha [a-zA-Z]
9  codeConcert ^T[0-9]{2,6}
10 codeDossier [0-9]{8}
11 date [0-9][0-9]\/[0-9]{1,2}(\/[0-9][0-9])?
12 FL (\n)
13 heure [0-9][0-9]:[0-9][0-9]
14 ignore [\t ]+
15 nbPlaces [1-9]?[0-9]
16 nomConcert [A-Z0-9]([A-Z0-9]|(-[A-Z0-9]))*
17 nomPropre {alpha}({alpha}|(-{alpha})) *
18 prenomNom ^{nomPropre}\/{nomPropre}
19
20
21 %%
22 {codeConcert}      {nbConcerts++;}
23 {codeDossier}      {codeDossier=strdup(yytext);}
24 {date}             {}
25 DOSSIER            {}
26 <<EOF>>            {return 0;}
27 {FL}               {}
28 {heure}            {}
29 {ignore}           {}
30 {nbPlaces}         {nbPlaces+=strtol(yytext, NULL, 10);}
31 {nomConcert}       {}
32 places            {}
33 {prenomNom}        {prenomNom=strdup(yytext);}
34
35 %%
36 int main()
37 {
38     yylex();
39     printf("Pour le dossier %s, %s a acheté %i places de %i concerts\n",
40           codeDossier, prenomNom, nbPlaces, nbConcerts);
41 }

```

Listing 2 – Seconde spécification appliquant la reconnaissance des différents champs d'une commande de billets

Deuxième partie

Des automates en récursif

Exercice 2 : Programmation en dur de manière récursive

Questions de compréhension

Question : Si votre automate a N états, combien de fonctions reconnaîtRec_ i devez vous écrire ?

Réponse : Si l'automate a N états alors il faudra écrire N fonctions reconnaîtRec_ i . En effet, dans les faits nous sommes en train d'implanter un système d'équations aux langages.

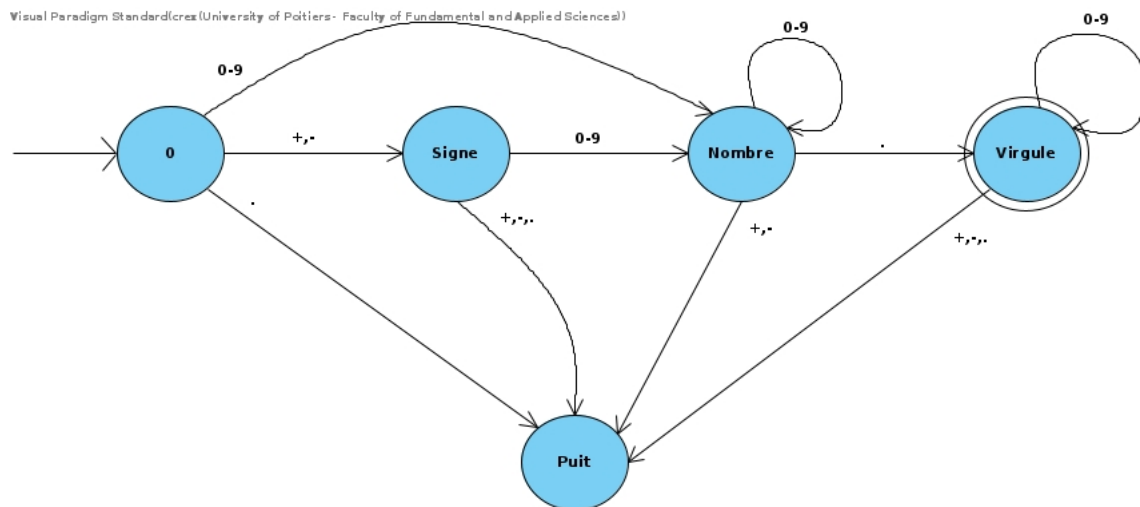
Question : Si l'état i est final, que doit retourner reconnaîtRec_ i ("") ? Et si i n'est pas final ?

Réponse : Un état i final signifie que reconnaîtRec_ i ("") doit retourner 'true', "" étant le mot vide aussi appelé ϵ . Tout état i non final doit alors retourner 'false' pour le mot vide.

Question : Si le paramètre 'mot' n'est pas vide et commence par un caractère c , quelle fonction reconnaîtRec_ i (mot) doit-elle appeler ? Et avec quel paramètre ?

Réponse : Si le paramètre 'mot' n'est pas vide et commence par un caractère c alors on doit appeler la fonction reconnaîtRec_ i (mot) qui correspond à l'état de destination dans la transition $q_{\text{courant}} \xrightarrow{c} q_i$. On appelle alors cette fonction avec pour paramètre le mot 'mot' tronqué de sa première lettre.

Automate des réels



Implantation des reconnaitRec

```
1  (* ----- *)
2  (*  Fonctions Auxiliaires  *)
3  (* ----- *)
4
5  let isCiffer c = c >= '0' && c <= '9';;
6  let isBinOp c = c = '+' || c = '-';;
7  let isComma c = c = '.';;
8  let explode m = List.init (String.length m) (String.get m);;
9
10
11  (* ----- *)
12  (*  Fonctions Principales  *)
13  (* ----- *)
14
15  let rec reconnaitRec_Virgule m =
16    match m with
17    | [] -> true
18    | firstCarac :: tl -> if(isCiffer firstCarac) then reconnaitRec_Virgule tl
19                          else false
19  ;;
20
21  let rec reconnaitRec_Nombre m =
22    match m with
23    | [] -> false
24    | firstCarac :: tl -> if(isCiffer firstCarac) then reconnaitRec_Nombre tl
25                          else if(isComma firstCarac) then reconnaitRec_Virgule
26                                tl
27                                else false
27  ;;
28
29  let reconnaitRec_Signe m =
30    match m with
31    | [] -> false
32    | firstCarac :: tl -> if(isCiffer firstCarac) then reconnaitRec_Nombre tl
33                          else false
34  ;;
35
36  let reconnaitRec_0 m =
37    match m with
38    | [] -> false
39    | firstCarac :: tl -> if(isCiffer firstCarac) then reconnaitRec_Nombre tl
40                          else if(isBinOp firstCarac) then reconnaitRec_Signe
41                                tl
42                                else false
42  ;;
```

Listing 3 – Début du code source d'automateEnDurReels.ml

Implantation de l'automate complet

```
1 let reconnaitReelRec m =  
2   reconnaitRec_0 (explode m)  
3 ;;
```

Listing 4 – Le reste du code source d'automateEnDurReels.ml

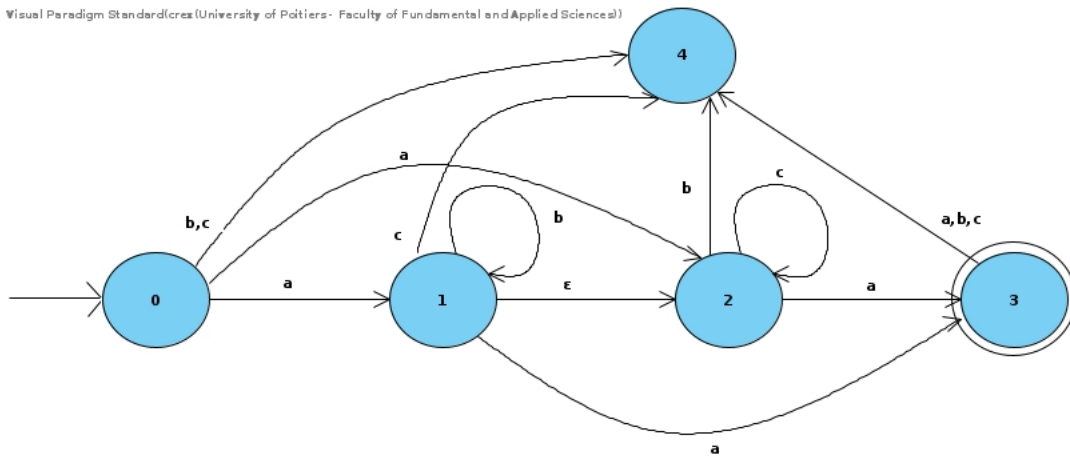
Programme complet

```
1 (* ===== *)  
2 (* AUTEUR: Florian Legendre *)  
3 (* OBJECTIF DE CE MODULE: Être le programme *)  
4 (* final qui sera compilé en un exécutable. *)  
5 (* Il est basé sur le travail produit dans les *)  
6 (* deux autres .ml de ce dossier. *)  
7 (* ===== *)  
8  
9 open List;;  
10 open AutomateEnDurReels;;  
11  
12 let main =  
13 while true do  
14   let wordToRead = read_line() in  
15   if(reconnaitReelRec wordToRead)  
16   then (print_string "True!"; print_newline());  
17   else (print_string "False!"; print_newline());  
18 done;  
19 ;;  
20  
21 main;;
```

Listing 5 – programme final lisant sans cesse sur le flux d'entrée

Exercice 3 : Des automates non déterministes représentés dans le code de manière récursive

Visual Paradigm Standard (crex (University of Poitiers - Faculty of Fundamental and Applied Sciences))



Questions de compréhension

Question : Comment dans le code de `reconnait_0` allez vous représenter le fait qu'en lisant un `a`, on puisse aller soit de l'état 0 à l'état 1, soit de l'état 0 à l'état 2 ?

Réponse : Un automate non déterministe signifie que l'on peut trouver un chemin valide (et donc reconnaître un mot) en passant soit par un chemin soit par un autre. Ce soit/soit ce représente en programmation par l'opérateur OU. Je vais donc faire pour la lettre 'a' à l'état 0 : `reconnaitRec1 resteDuMot || reconnaitRec2 resteDuMot`

Question : Comment dans le code de `reconnait_1` allez vous représenter le fait que l'on peut passer directement, sans rien lire, à l'état 2 ?

Réponse : Similairement au cas si dessus je peux faire un OU avec directement l'appel récursif de `reconnaitRec2` en donnant en paramètre de `reconnaitRec2` le mot complet sans avoir "mangé" de lettre.

Implantation et Tests de l'automate

```
1  (* ----- *)
2  (*  Fonctions Auxiliaires  *)
3  (* ----- *)
4
5  let explode m = List.init (String.length m) (String.get m);;
6
7
8  (* ----- *)
9  (*  Fonctions Principales  *)
10 (* ----- *)
11
12 let reconnaitRec_3 m =
13   match m with
14   | [] -> true
15   | firstCarac::tl -> false
16 ;;
17
18 let rec reconnaitRec_2 m =
19   match m with
20   | [] -> false
21   | firstCarac::tl -> if (firstCarac = 'c') then reconnaitRec_2 tl
22                       else if (firstCarac = 'a') then reconnaitRec_3 tl
23                       else false
24 ;;
25
26 let rec reconnaitRec_1 m =
27   match m with
28   | [] -> false
29   | firstCarac::tl -> (if (firstCarac = 'b') then reconnaitRec_1 tl
30                       else if (firstCarac = 'a') then reconnaitRec_3 tl
31                       else false) || reconnaitRec_2 m
32 ;;
33
34 let reconnaitRec_0 m =
35   match m with
36   | [] -> false
37   | firstCarac::tl -> if (firstCarac = 'a') then (reconnaitRec_1 tl) || (
38   reconnaitRec_2 tl)
39                       else false
40 ;;
41
42 let reconnaitL4 m =
43   reconnaitRec_0 (explode m)
44 ;;
```

Listing 6 – Code source d'automateEnDurEx3.ml

Exercice 4 : Évaluation du réel correspondant à la chaîne de caractères

Questions de compréhension

Question : Comment allez vous gérer votre position dans la partie décimale (x^{eme} position après la virgule = indique la puissance de dix négative?), et vous en servir pour prendre en compte la nouvelle décimale lue?

Réponse :

Question : Comment allez vous gérer le calcul de la partie entière, lorsqu'un nouveau chiffre est lu?

Réponse :

Question : Comment allez vous gérer la transmission du calcul d'une routine récursive à l'autre? Variables globales, paramètres d'entrée sortie, valeurs de retour de fonction?

Réponse :

Implantation de la fonction d'évaluation des réels

Implantation du programme final d'évaluation des réels