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'

```
%{
    char* codeDossier;
    char* prenomNom;
    int nbPlaces = 0;
    int nbConcerts = 0;
%}

alpha [a-zA-Z]
codeConcert ^T[0-9]{2,6}
codeDossier [0-9]{8}
date [0-9][0-9]\/[0-9]{1,2}(\/[0-9][0-9])?
FL (\n)
heure [0-9][0-9]:[0-9][0-9]
ignore [\t ]+
nbPlaces [1-9]?[0-9]
nomConcert [A-Z0-9]([A-Z0-9]|(-[A-Z0-9]))*
nomPropre {alpha}({alpha}|(-{alpha}))*
prenomNom ^{nomPropre}\/{nomPropre}


%%
{codeConcert}     {nbConcerts++;}
{codeDossier}     {codeDossier=strdup(yytext);}
{date}            {}
DOSSIER           {}
<<EOF>>           {return 0;}
{FL}              {}
{heure}           {}
{ignore}          {}
{nbPlaces}        {nbPlaces+=strtol(yytext, NULL, 10);}
{nomConcert}      {}
places            {}
{prenomNom}       {prenomNom=strdup(yytext);}

%%
int main()
{
    yylex();
    printf("Pour le dossier %s, %s a acheté %i places de %i concerts\n",
       codeDossier, prenomNom, nbPlaces, nbConcerts);
}
```

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 reconnaitRec_i devez vous écrire ?

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

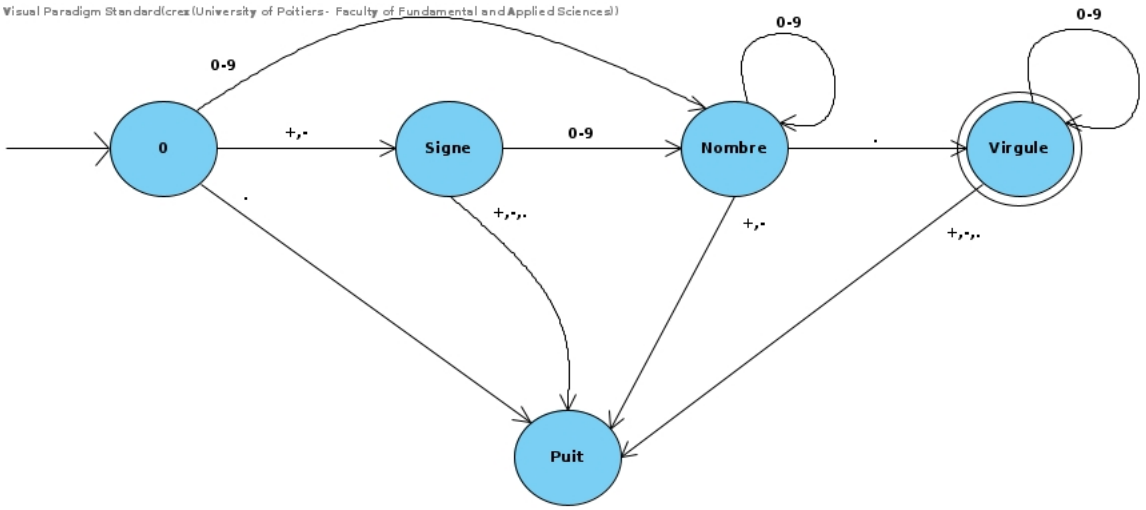
Question : Si l'état i est final, que doit retourner reconnaitRec_i("") ? Et si i n'est pas final ?

Réponse : Un état i final signifie que reconnaitRec_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 reconnaitRec_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 reconnaitRec_i(mot) qui correspond à l'état de destination dans la transition $q_{courant} \xrightarrow{c} q_i$. On appelle alors cette fonction avec pour paramètre le mot 'mot' tronqué de sa première lettre.

Automate des réels



Implantation des reconnaitRec

```
(* ------------------------ *)
(*  Fonctions Auxiliaires  *)
(* ------------------------ *)

let extractFirstChar m =
  String.get m 0
;;

let isCiffer c =
  c >= '0' && c <= '9'
;;

let isBinOp c =
  c = '+' || c = '-'
;;

let isComma c =
  c = '.'
;;

let truncateWord m =
  String.sub m 1 ((String.length m) - 1)
;;


(* ------------------------ *)
(*  Fonctions Principales  *)
(* ------------------------ *)

let rec reconnaitRec_Virgule(m : string) : bool =
  if(m = "")
  then true

  else
    (
      let firstCarac = extractFirstChar m in
      let truncatedWord = truncateWord m in

      if(isCiffer firstCarac)
      then reconnaitRec_Virgule truncatedWord

      else false
    )
;;

let rec reconnaitRec_Nombre(m : string) : bool =
  if(m = "")
  then false

  else
    (
      let firstCarac = extractFirstChar m in
      let truncatedWord = truncateWord m in

```

```
55        if(isCiffer firstCarac)
56        then reconnaitRec_Nombre truncatedWord
57
58        else if(isComma firstCarac)
59        then reconnaitRec_Virgule truncatedWord
60
61        else false
62      )
63  ;;
64
65  let reconnaitRec_Signe(m : string) : bool =
66    if(m = "")
67    then false
68
69    else
70      (
71        let firstCarac = extractFirstChar m in
72        let truncatedWord = truncateWord m in
73
74        if(isCiffer firstCarac)
75        then reconnaitRec_Nombre truncatedWord
76
77        else false
78      )
79  ;;
80
81
82  let reconnaitRec_0(m : string) : bool =
83    if(m = "")
84    then false
85
86    else
87      (
88        let firstCarac = extractFirstChar m in
89        let truncatedWord = truncateWord m in
90
91        if(isCiffer firstCarac)
92        then reconnaitRec_Nombre truncatedWord
93
94        else if(isBinOp firstCarac)
95        then reconnaitRec_Signe truncatedWord
96
97        else false
98      )
99  ;;
```

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

Implantation de l'automate complet

```
1  let reconnaitReelRec(m : string) : bool =
2    reconnaitRec_0 m
3  ;;
```

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

Programme complet

```
(* ============================================ *)
(* AUTEUR: Florian Legendre                     *)
(* OBJECTIF DE CE MODULE: Être le programme     *)
(* final qui sera compilé en un exécutable.     *)
(* Il est basé sur le travail produit dans les *)
(* deux autres .ml de ce dossier.               *)
(* ============================================ *)

open List;;
open AutomateEnDurReels;;

let main =
while true do
  let wordToRead = read_line() in
  if(reconnaitReelRec wordToRead)
  then (print_string "True!"; print_newline();)
  else (print_string "False!"; print_newline();)
done;
;;

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

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 :

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 :

Implantation et Tests de l'automate

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