Problema A

João Luís Freire Fraga

April 4, 2019

Contents

1 Explicacao do problema

O problema consiste em criar um programa que, tendo como input:

- Estados
- Estados iniciais
- Estados finais
- Transicoes possiveis
- Palavra

Diga se a palavra é reconhecida pelo automato. Ser reconhecido pelo automato implica que, comecando num dos estados iniciais, consumindo todas as letras da palavra letra a letra da esquerda para a direita e efectuando as actulizacoes necessarias, existe pelo menos um estado que pertence ao conjunto dos estados finais.

2 Leitura de input

2.1 Numero de estados

```
(* Number of states *)
let (n:int) = Scanf.scanf " %d" (fun x -> x)
```

2.2 Cardinalidade estados iniciais

```
(* Cardinality of S *)
let (cardS:int) = Scanf.scanf " %d" (fun x -> x)
```

2.3 Estados iniciais

Para ler os estados iniciais usei uma lista de tuples de inteiros "(int * int)" (s, x):

- s: state $\{1, \ldots, n\}$
- x: estado interno (inicialmente 0)

```
(* Initial states *)
let (is:(int * int) list) =
  let rec readInitialState lst (i:int) =
    if i = 0 then
      lst
    else
      let state = Scanf.scanf " %d" (fun x -> x) in
      let lst = lst@[(state, 0)] in
      readInitialState lst (i-1)
  in
  readInitialState [] cardS
```

2.4 Cardinalidade estados finais

```
(* Cardinality of F*)
let (cardF:int) = Scanf.scanf " %d" (fun x -> x)
```

2.5 Estados finais

Para ler os estados finais usei uma lista de inteiros

```
(* Final states *)
let (fs:int list) =
  let rec readFinalState lst (i:int) =
    if i = 0 then
      lst
    else
      let s = Scanf.scanf " %d" (fun x-> x) in
      let lst = lst@[s] in
      readFinalState lst (i-1)
  in
  readFinalState [] cardF
```

2.6 Numero de transicoes

```
(* Number of transitions *)
let (m:int) = Scanf.scanf " %d" (fun x -> x)
```

2.7 Transicoes

Para ler as transicoes usei uma lista de tuples do tipo "(int * char * string * int * int * int)" (i, c, op, a, b, j):

- i: inicio
- c: caracter consumido
- op: operaçao de comparacao
- a: valor a ser comparado
- b: valor a ser actualizado
- j: fim

```
(* Transitions *)
let (t:(int * char * string * int * int * int) list) =
  let rec readTransitions lst (k:int) =
    if k = 0 then
      lst
    else
      let i  = Scanf.scanf " %d" (fun x -> x) in
      let c  = Scanf.scanf " %c" (fun x -> x) in
      let op = Scanf.scanf " %s" (fun x -> x) in
      let a  = Scanf.scanf " %c" (fun x -> x) in
      let a = if a = '_' then (-1) else (int_of_char a - 48) in
      let b  = Scanf.scanf " %c" (fun x -> x) in
      let b = if b = '_' then (-1) else (int_of_char b - 48) in
      let j  = Scanf.scanf " %d" (fun x -> x) in
      let transition   = (i, c, op, a, b, j) in
      let lst = lst@[transition] in
      readTransitions lst (k-1)
  in
  readTransitions [] m
```

2.8 Palavra

```
(* Word *)
let (w:string) = Scanf.scanf " %s" (fun x -> x)
```

3 Funcoes

3.1 checkState

Esta funcao percorre a lista de estados actuais e verifica se algum deles se encontra na lista de estados finais.

```
(* This function checks if any of the states is a final state *)
let rec checkState state finalState =
  match state with
  | [] -> false
  | (s, _)::tl ->
    if List.mem s finalState then
      true
    else
      checkState tl finalState
```

3.2 operation

Esta função converte uma string operação no valor booleano do resultado dessa operacao usando x e a

```
(* This function converts the variables and operation into a boolean expression *)
let operation x op a =
  match op with
  | "_" -> true
  | "<" -> x < a
  | "<=" -> x <= a
  | "=" -> x = a
  | "!=" -> x <> a
  | ">=" -> x >= a
  | ">" -> x > a
  | _ -> false
```

3.3 followE

Esta funcao adiciona todos os estados onde e possivel chegar a partir dos estados actuais usando uma transicao epsilon

```
(* This function follows current state's e's *)
let rec followE state newState transitions =
  match state with
  | [] -> newState
  | (s, x)::stl ->
    let newState =
      if not (List.mem (s, x) newState) then
        newState@[(s, x)]
      else
        newState
    in
    let rec followETransitions newState transitions =
      match transitions with
      | [] -> newState
      | (i, c, op, a, b, j)::ttl ->
        let newState =
          if c = '_' && s = i then
            if b = (-1) then
              if not (List.mem (j, x) newState) then
                newState@[(j, x)]
              else
                newState
            else
            if not (List.mem (j, b) newState) then
              newState@[(j, b)]
            else
              newState
          else
            newState
        in
        followETransitions newState ttl
    in
    let newState = followETransitions newState transitions in
    followE stl newState transitions
```

3.4 followAllE

Esta funcao executa a funcao anterior enquanto esta produzir estados novos

```
(* This function follows all e's possible *)
let rec followAllE state lastState transitions =
  let state = followE state [] transitions in
  if state <> lastState then
    let lastState = state in
    followAllE state lastState transitions
  else
    state
```

3.5 consumeL

Esta funcao segue todas as transicoes que partem dos estados actuais e consomem a letra l, quando permitido pela guarda, e actualiza os tornando todos estes estados resultantes, os estados actuais.

```
(* This function follows the letters *)
let rec consumeL l state newState transitions =
  match state with
  | [] -> newState
  | (s, x)::stl ->
    let rec followLTransitions newState transitions =
      match transitions with
      | [] -> newState
      | (i, c, op, a, b, j)::ttl ->
        let newState =
          if c = l && s = i && operation x op a then
            if b = (-1) then
              newState@[(j, x)]
            else
              newState@[(j, b)]
          else newState
        in
        followLTransitions newState ttl
    in
    let newState = followLTransitions newState transitions in
    consumeL l stl newState transitions
```

3.6 solution

Esta funcao utiliza todas as funcoes anteriores para decidir se a palavra w e reconhecida pelo automato. Se a palavra foi totalmente consumida e um dos estados actuais e um estado final então sim, caso contrario nao

```
(* This function checks if the word is recognized *)
let rec solution w state finalState transitions =
  if w = "" || state = [] then
    let state = followAllE state state transitions in
    checkState state finalState
  else
    let l = String.get w 0 in
    let state = followAllE state state transitions in
    let state = consumeL l state [] transitions in
    let w = (String.sub w 1 ((String.length w) - 1)) in
    solution w state finalState transitions
```

4 Execucao

Imprimir a solucao para o stdout

```
(* Get answer *)
let () =
  if solution w is fs t then
    Printf.printf "YES\n"
  else
    Printf.printf "NO\n"
```