Laporan Tugas Teori Bahasa Formal & Otomata : Implementasi DFA pada *Marble Rolling Toy*

Turfa Auliarachman, 13515133

19 September 2016

Chapter 1

Deskripsi DFA Terkait

1.1 Deskripsi Persoalan

Terdapat sebuah *marble rolling toy* seperti pada gambar berikut :

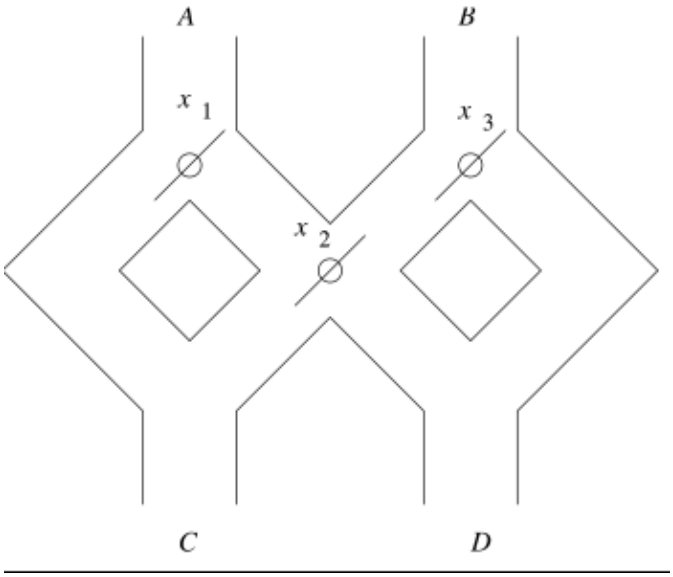


Figure 1.1: Ilustrasi *Marble Rolling Toy*

Marble akan digelindingkan dari A atau B. Ketika mengenai penghalang (x_1, x_2, x_3), *marble* akan diarahkan sesuai penghalang tersebut. Lalu, penghalang yang dilalui *marble* akan berganti arah. Jika *marble* keluar di D, maka permainan dianggap berhasil. Mula-mula, semua penghalang menghadap ke kiri.

Permasalahan di atas ditranslasikan sebagai permasalahan DFA. Lalu, harus dibuat sebuah program dalam bahasa C atau Pascal yang memproses permasalahan DFA yang definisinya ada di sebuah file eksternal, lalu mengetes apakah string masukan pengguna diterima oleh DFA atau tidak. Program

DFA itu akan digunakan untuk mengetes pada persoalan *Marble Rolling Toy* apakah *marble* terakhir masukan pengguna keluar di D atau tidak.

1.2 DFA

Translasi permasalahan *Marble Rolling Toy* dalam notasi DFA :

$$A = (Q, \Sigma, \delta, q_0, F)$$

di mana

$$\begin{aligned} Q &= \{LLLC, RLLC, LRRC, LRLC, RRRC, LRLD, \\ &\quad RRLC, LLRD, RRLD, LLLD, RLRD, RLRC, RLLD\} \\ \Sigma &= \{A, B\} \\ q_0 &= LLLC \\ F &= \{LRLD, LLRD, RRLD, LLLD, RLRD, RLLD\} \end{aligned}$$

dan δ dalam tabel berikut :

Table 1.1: Tabel δ (bukan tabel notasi sederhana DFA)

q	$\delta(q, A)$	$\delta(q, B)$
$LLLC$	$RLLC$	$LRRC$
$RLLC$	$LRLC$	$RRRC$
$LRRC$	$RRRC$	$LRLD$
$LRLC$	$RRLC$	$LLRD$
$RRRC$	$LLRD$	$RRLD$
$LRLD$	$RRLC$	$LLRD$
$RRLC$	$LLLD$	$RLRD$
$LLRD$	$RLRC$	$LLLD$
$RRLD$	$LLLD$	$RLRD$
$LLLD$	$RLLC$	$LRRC$
$RLRD$	$LRRC$	$RLLD$
$RLRC$	$LRRC$	$RLLD$
$RLLD$	$LRLC$	$RRRC$

Chapter 2

Pembuatan Program

2.1 Daftar Asumsi

Beberapa asumsi yang digunakan dalam pembuatan program DFA ini adalah

- Banyak *state* maksimal 1000.
- Simbol berupa karakter.
- File eksternal yang digunakan untuk mendapatkan deskripsi DFA bernama *deskripsi.dat* dan berformat sebagai berikut:

```
(Jumlah state)
(Daftar state, dipisahkan spasi)
(Daftar simbol, tidak dipisahkan spasi)
(State awal)
(Jumlah final state)
(Daftar state akhir, dipisahkan spasi)
(Fungsi transisi berbentuk tabel)
```

Untuk fungsi transisi, ketentuannya sebagai berikut :

- Urutan *state* sesuai penulisan di *deskripsi.dat*
- Urutan simbol sesuai penulisan di *deskripsi.dat*
- Tabel fungsi transisi terdiri dari sejumlah state baris, yang tiap baris berisi sejumlah simbol state
- Untuk setiap i dan j dengan $1 \leq i \leq jumlahstate$ dan $1 \leq j \leq jumlahsimbol$, simbol ke-$j$ akan mengarahkan state i ke state ke-j yang ada di baris ke-i

2.2 Isi File Eksternal

Berdasarkan DFA yang dijelaskan pada bagian 1.2 dan format isi file eksternal yang dijelaskan pada bagian 2.1, dibuat file *deskripsi.dat* yang isinya :

Listing 2.1: deskripsi.dat

```
13
LLLC RLLC LRRC LRLC RRRC LRLD RRLC LLRD RRLD LLLD RLRD
    RLRC RLLD

AB

LLLC

6
LRLD LLRD RRLD LLLD RLRD RLLD

RLLC LRRC
LRLC RRRC
RRRC LRLD
RRLC LLRD
LLRD RRLD
RRLC LLRD
LLLD RLRD
RLRC LLLD
LLLD RLRD
RLLC LRRC
LRRC RLLD
LRRC RLLD
LRLC RRRC
```

2.3 *Source Code*

Program dibuat dalam bahasa C. Untuk memudahkan penulisan dan pembacaan kode, *source code* dibagi ke dalam dua file program yaitu *main.c* dan *dfa.c* serta satu *header* yaitu *dfa.h*, yang merupakan *header* untuk file *dfa.c*.

File *dfa.c* mengimplementasikan DFA dalam bahasa C, sedangkan *main.c* menyelesaikan masalah yang diberikan soal menggunakan *dfa.c*.

Listing 2.2: dfa.h

```
#ifndef __DFA_H
#define __DFA_H

#define bool short
```

```
#define true 1
#define false 0

#define MAX_ELEMENT 1000
#define MAX_SYMBOL 256
#define MAX_LENGTH 1000

extern char * idElement[MAX_ELEMENT];
extern char idSymbol[MAX_SYMBOL];
extern int toWhere[MAX_ELEMENT][MAX_SYMBOL];
extern bool isFinal[MAX_ELEMENT];

extern int nElement, nSymbol, nFinal;
extern int startState;

void initDFA();

void setIdElement(int number, char * str);

void setIdSymbol(int number, char c);

int getIdElement(char * str);

int getIdSymbol(char c);

int next(int now, char c);

#endif
```

Listing 2.3: dfa.c

```
#include "dfa.h"
#include <string.h>
#include <stdlib.h>
#include <assert.h>

char * idElement[MAX_ELEMENT];
char idSymbol[MAX_SYMBOL];
int toWhere[MAX_ELEMENT][MAX_SYMBOL];
bool isFinal[MAX_ELEMENT];

int nElement, nSymbol, nFinal;
int startState;

void initDFA(){
```

```
  int i;

  for(i=0;i<MAX_ELEMENT;i++){
    idElement[i] = NULL;
  }

  memset(idSymbol, 0, sizeof(idSymbol));
  memset(toWhere, -1, sizeof(toWhere));
  memset(isFinal, 0, sizeof(isFinal));
}

void setIdElement(int number, char * str){
  idElement[number] = malloc(strlen(str)+1);
  strcpy(idElement[number],str);
}

void setIdSymbol(int number, char c){
  idSymbol[number] = c;
}

int getIdElement(char * str){
  int i=0;
  bool found = false;

  while(true){
    if (i==MAX_ELEMENT) break;
    if (idElement[i]==NULL) break;
    if (strcmp(idElement[i],str)==0){
      found = true;
      break;
    }

    i++;
  }

  assert(found);
  return i;
}

int getIdSymbol(char c){
  int i=0;
  bool found = false;

  while(true){
```

```
    if (i==MAX_SYMBOL) break;
    if (idSymbol[i]==0) break;
    if (idSymbol[i]==c){
      found = true;
      break;
    }

    i++;
  }

  assert(found);
  return i;
}

int next(int now, char c){
  int ret = toWhere[now][getIdSymbol(c)];
  assert(ret!=-1);

  return ret;
}
```

Listing 2.4: main.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>
#include "dfa.h"

#define LANG_MAX_LENGTH 1000

void readFile(char * filename);

int main(int argc, char ** argv){
  int i;
  int now;
  int len;

  char str[LANG_MAX_LENGTH+1];

  initDFA();
  if (argc>1) readFile(argv[1]);
  else readFile("deskripsi.dat");

  printf("Tulis string input, maksimal %d karakter :\n",
```

```
        LANG_MAX_LENGTH);
    fgets(str, LANG_MAX_LENGTH+1, stdin);
    len = strlen(str);

    now = startState;

    printf("->%s\n", idElement[startState]);

    for(i = 0; i < len-1; i++){
      now = next(now, str[i]);
      printf("Mendapat input '%c', sekarang ada di state %
          s\n", str[i], idElement[now]);
    }

    if (isFinal[now]){
      printf("Berakhir di Final State, yaitu %s. String
          diterima\n", idElement[now]);
    }
    else{
      printf("Berakhir tidak di Final State, yaitu %s.
          String ditolak\n", idElement[now]);
    }
}

void readFile(char * filename){
    FILE *f;
    int n,i,j;
    char *inp = malloc(MAX_LENGTH+5);

    f = fopen(filename, "r");

    assert(f!=NULL);

    /*(Jumlah state)*/
    fscanf(f, "%d", &n);
    assert(n <= MAX_ELEMENT);
    nElement = n;

    /*(Daftar state, dipisahkan spasi)*/
    for(i=0;i<nElement;i++){
      fscanf(f,"%s", inp);

      for(j=0;j<i;j++){
        assert(strcmp(inp,idElement[j])!=0);
```

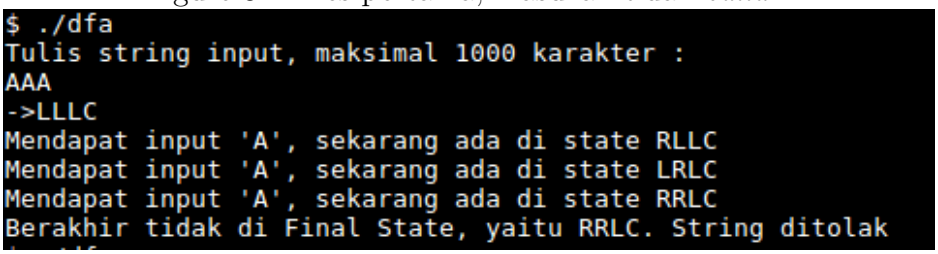
```
    }

    setIdElement(i,inp);
  }

  /*(Daftar simbol, tidak dipisahkan spasi)*/
  fscanf(f, "%s", inp);
  assert(strlen(inp)<=MAX_SYMBOL);
  nSymbol = strlen(inp);

  for(i = 0; i < nSymbol; i++){
    for(j = 0; j < i; j++){
      assert(inp[i]!=inp[j]);
    }

    setIdSymbol(i, inp[i]);
  }

  /*(State awal)*/
  fscanf(f, "%s", inp);
  startState = getIdElement(inp);

  /*(Jumlah final state)*/
  fscanf(f, "%d", &n);
  assert(n <= nElement);
  nFinal = n;

  /*(Daftar state akhir, dipisahkan spasi)*/
  for(i=0;i<nFinal;i++){
    fscanf(f, "%s", inp);

    j = getIdElement(inp);
    assert(!(isFinal[j]));
    isFinal[j] = true;
  }

  /*(Transition function berbentuk tabel)*/
  for(i=0;i<nElement;i++){
    for(j=0;j<nSymbol;j++){
      fscanf(f,"%s",inp);
      toWhere[i][j] = getIdElement(inp);
    }
  }
}
```

Chapter 3

Testing

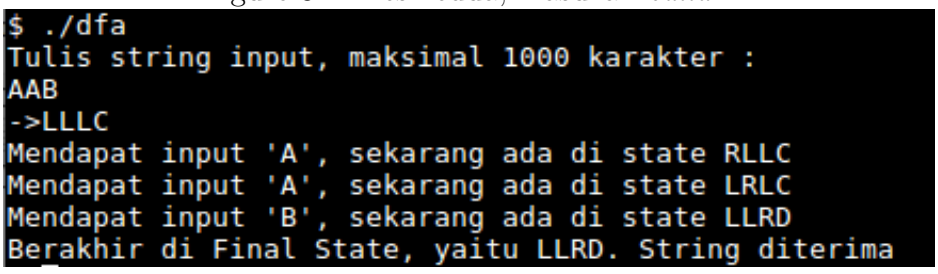
Program dites menggunakan dua masukan. Yang pertama, program dites menggunakan masukan yang tidak *valid*, yaitu AAA. Hasilnya adalah :

Figure 3.1: Tes pertama, masukan tidak *valid*



Lalu, program dites menggunakan masukan yang *valid*, yaitu AAB. Hasilnya adalah :

Figure 3.2: Tes kedua, masukan *valid*



Dari kedua tes tersebut, didapat bahwa program melakukan simulasi *state-state* yang dilalui dengan benar. Selain itu, program juga memberikan kesimpulan tentang diterima atau tidaknya sebuah masukan dengan benar.