Finite State Machines
Assignment

September 18, 2019

1 Submission, Marking

You will be asked to draw some diagrams in this project. Please scan and include image files in your report if you draw diagrams by hand.
You can use any software you like for producing diagrams. We have **ArgoUML** installed on ECS computers. There is also simpler software called **Umbrello**.

Please submit pdf for the answers and one zip with source files.

All code should be written in C++.
Please make sure that cpp file names are in accordance with the Exercise number.
Your code should ask user to entry string from the keyboard and print the result.
It will tested against 10 input sequences, some of which should be accepted and some should be rejected. Correct performance in each case - 1 mark.

Submission due Monday, 7 October.

For each question please submit FSM drawing and code.

2 Acceptors: FSM for pattern recognition

Here is some code to get you started. It is not complete - all transition actions are missing.

Listing 1: Starting code

```
/*Example FSM code*/
/*Reasds string from keyboard*/
/*
Runs FSM on each character of the
string (calling function Pro)
until # character is found
*/
#include <iostream>
#include <string>

using namespace std;

string state;  // current FSM state

/*Describes an FSM with four states: s0, s1,s2 and s3 */
/* s3 is an accepting state*/
/*returns 1 if FSM is in accepting state*/
int ProcessChar(char in_char){
int accept = 0;
if (state == "s0"){
  if (in_char == 'c'){
   // ..
  } else
  if (state == "s1"){
 //..
 } else
 if (state == "s2"){
  // ..
  } else
  if (state == "s3") {
   //..
  }
 }
 if (state == "s3") {accept =1;}
 cout<<"in_char="<<in_char<<" accept="<<accept<<endl;
 return accept;
}
```

```
int main(){
 string input;
 cout<<"enter input string:";
 cin>>input;
 cout<<"Input is "<<input<<" state is"<<state<<endl;
 unsigned int count = 0;
 int string_accepted = 0;
 state = "s0"; //starting state
 count = 0;
 while ( 1 ){
   if (input.at(count) == '#') break; // run until #
   string_accepted = ProcessChar(input.at(count));
   count++;
 }
 cout<<" accepted = "<<string_accepted<<endl;
}
```

Exercise 1, 10 points for diagram, 10 points for code

Input alphabet is comprised of all lower-case letters of English alphabet.
Input is a string. Hash symbol # is indicator of the end of string.
Design an FSM which accepts string with exact word "cat" in it. All other strings should be rejected. String with "cbat", for example, should be rejected. "ccat" should be accepted. "cat" can be anywhere in the string - FSM should enter accepting state after reading sequence "cat" and stay there.
Include picture of FSM into your report. Submit code as "Ex1.cpp" in zip file.

Exercise 2, 10 points for diagram, 10 points for code

Modify FSM from Exercise 1 so that it accepts only strings ending with substring "**cat**" - it should come immediately before termination symbol #.

Exercise 3, 10 points for diagram, 10 points for code

Input alphabet is comprised of all digits: 0 to 9.
Inputs are arranged as set of strings. Hash # is again indication of the end of string.
Design an FSM which accepts strings which are unsigned integers. Unsigned integer is defined as either a single 0 or a non-zero digit followed by any number of digits.

3 Transducers

Exercise 4, 10 points for diagram, 10 points for code

Input alphabet is all English letters and commas.
Design an FSM which takes string which can contain repeating commas and trims commas to one only. **"Mary,,,,had,,,,,a little,lamb"**, for example, should be trimmed to **"Mary,had,a little,lamb"**.
This FSM has an output (string with trimmed commas). Introduce new variable to hold current output character. Print output on the screen.

4 Enhanced FSMs (using variables)

You may want to use FSM with variables for these questions.
Some programming background for next question:

- strings can be treated as $<$ *vector* $>$s in C++. **char** can be **push_back** into the string.
- Applying **string_name.clear()** to string removes all characters to the string
- Function **int stoi(string)** takes string (or $<$ *vector* $>$ of chars) and returns integer number.

Exercise 5, 10 points for diagram, 10 points for code

Parsing Comma-Separated-Values (CSV).
Input alphabet is set of the digits from 0 to 9 and comma.
Design an FSM which breaks the string into set of **tokens**. Each token the number is composed of the digits between commas. Tokens(numbers) are stored into array (vector) when program terminates.
Output array(vector) is printed on the screen when program terminates.