Finite State Automata

» Contents

» What is Finite State Automata?

- A Finite State Automata (FSA), also known as a Finite State Machine (FSM), is a mathematical model used to represent and control the execution of processes that can be in a finite number of states
- Imagine it as a simple computer with a limited amount of memory. It can only be in one state at a time and transitions between these states based on the input it receives.

» Definition

Definition

Finite state automata is 3 tuple(S, Σ,T) where

S A finite set of states, one of which is the initial state s_{init}.

Σ The alphabet, of source symbols.

T is a finite set of state transitions defining transitions out of each s_i ϵ S on encountering the symbols Σ.

» State Transitiion Tables

- We label the transitions of FSA using the conventions:
  A transition out of $s_i \epsilon$ S on encountering a symbol symb $\epsilon \ \Sigma$ has the label symb
- We say a symbol symb is recognized by FSA when the FSA makes a transition labelled symb.
- The transitions in an FSA can be represented in the form of a State Transition Table (STT).
- An STT has one row for each state s_i and one column for each symbol symb $\epsilon \ \Sigma$

» State Transition Tables Contd

- An entry in STT(s_i, symb) in the table indicates the id of the new state entered by the FSA if there exists a transition labelled symb in state s_i.
- if the FSA does not contain a transition out of state s_i we leave STT(s_i, symb) blank
- A state transition can also be represented by a triple (old state, source symbol, new state).
- thus the entry STT(s_i, symb) $=$ s_j and the triple (s_i, symb, s_j) are equivalent.

» Types

- Deterministic FSA
- Non Deterministic FSA

» Deterministic FSA

- In DFA Transitions are Deterministic , that is at most one transition exists in state s_i for a symbol symb

Defintion A deterministic finite state automation (DFA) is an FSA such that $t_i \epsilon$ T, $t_1 \equiv (s_i$, symb, $s_j)$
implies $\ni!$ $t_2 \epsilon$ T, $t_2 \equiv (s_i$, symb, $s_k)$

» Deterministic FSA

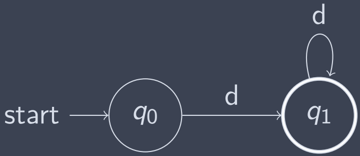
- At any point in time DFA would recognize some prefix α of the source string, possibly the null string.
- The operation of DFA is history sensitive because its current state is a function of the prefix recognized by it.
- The DFA halts when all symbols in the source string are recognized, or an error condition is encountered.
- The string is valid if and only if the DFA recognizes every symbol in the string and find itself in a final state at the end of the sring.

» Transition Table

State	Next Symbol
	d
Start	Int
Int	Int

STT to recognize integer string

» Visual Representation



DFA for Integer strings

» DFA States Description

q_0 Initial state

q_1 Valid identifier state (accepts only integer strings)

Accepts:

- Literals: 42, 314, 1234 100

» Regular Expressions

Definition Regular expressions are a generalization of type 3 production rules.

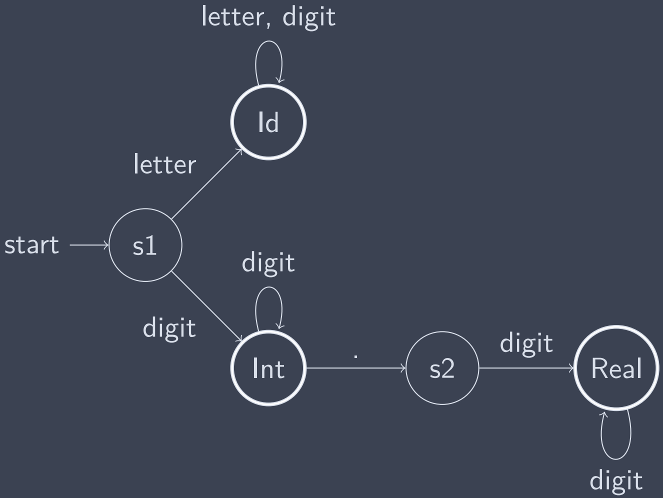
- The regular expression generalizes on Type 3 rules by premitting multiple occurences of a string form, and concatenation of strings.

» Some Regexes

Regular Expression	Meaning
r	string r
s	string s
$r.s$ or rs	concatenation of r and s
(r)	same meaning as r
$r \mid s$ or $(r \mid s)$	alternation, i.e., string r or string s
$(r) \mid (s)$	alternation
$[r]$	an optional occurrence of string r
$(r)^*$	0 or more occurrences of string r
$(r)^+$	1 or more occurrences of string r

Regular Expression Notation

» DFA for integers, real numbers and identifiers



» DFA States Description

s1 Initial state

Id Valid identifier state (accepts letters, digits)

Int Integer literal state

s2 Decimal point after integer state (not a final state)

Real Decimal point state

Accepts:

- Identifiers: x, name, my_var, temp1
- Literals: 42, 3.14, 0.123, 100

» Transition Table

State	Next Symbol		
	l	d	.
s1	Id	int	
Id	Id		
Int		Int	s2
s2		Real	
Real		Real	

Transition Table for integers, real numbers and identifiers

» DFA Flow

- Start State

» DFA Flow

- Start State
- Goes to Id state from Start if Letter is the first input symbol

» DFA Flow

- Start State
- Goes to Id state from Start if Letter is the first input symbol
- Goes to the same state Id for digits and letters as input symbols

» DFA Flow

- Start State
- Goes to Id state from Start if Letter is the first input symbol
- Goes to the same state Id for digits and letters as input symbols
- Goes to Int state from Start state if a digit is received as the input symbol

» DFA Flow

- Start State
- Goes to Id state from Start if Letter is the first input symbol
- Goes to the same state Id for digits and letters as input symbols
- Goes to Int state from Start state if a digit is received as the input symbol
- Goes to the same state Int for Consecutive digit input symbols

» DFA Flow

- Start State
- Goes to Id state from Start if Letter is the first input symbol
- Goes to the same state Id for digits and letters as input symbols
- Goes to Int state from Start state if a digit is received as the input symbol
- Goes to the same state Int for Consecutive digit input symbols
- Goes to s2 state from Int state if a decimal point received as the input symbol

» DFA Flow

- Start State
- Goes to Id state from Start if Letter is the first input symbol
- Goes to the same state Id for digits and letters as input symbols
- Goes to Int state from Start state if a digit is received as the input symbol
- Goes to the same state Int for Consecutive digit input symbols
- Goes to s2 state from Int state if a decimal point received as the input symbol
- Goes to Real state for further digits as input symbols