Lecture 8
Deterministic Finite Automata

CS 241: Foundations of Sequential Programs
Spring 2017

Troy Vasiga et al. (where Dan Holtby ∈ al)
University of Waterloo
Review

- formal languages give a theoretical basis for communication and organizing processes
- terminology (alphabet, word, language)
- specification vs. recognition
- studying language levels that increase in power/complexity
- regular languages are composed of union, concatenation and repetition
Recognizers: Finite Automata

Regular languages can be recognized by *finite automata*. We begin with *deterministic finite automata*, also called DFAs.

- alphabet ($\Sigma$)
- finite set of states (or it wouldn’t be a *finite* automata)
- start state
- accepting (final) states(s)
- transition function (or table, or diagram)
Finite Automata Example 1

Example: selected opcodes from MIPS assembly language, where alphabet is the ASCII characters.
Observations About Finite Automata

- ability to trace
- transitions out of a state are unique
- errors
- size of this language
- DFA M and language L(M)
Finite Automata Example 2

Example: MIPS labels, where the alphabet is ASCII characters.
More Finite Automata Examples

Let $\Sigma = \{a, b, c\}$.

- strings with exactly one $a$ and exactly one $b$ and no $c$'s

- strings with one $a$, one $b$ and one $c$

- strings with at least one $a$

- string with an even number of $a$'s
More Finite Automata Examples

Let $\Sigma = \{a, b, c\}$.

- strings with an even number $a$’s and odd number of $b$’s
- strings with an even number $a$’s or odd number of $b$’s
More Finite Automata Examples

Let $\Sigma = \{a - z\}$.

- All strings that contain "dan"

- All strings that contain "adam"
DFA summary

- a.k.a. finite state machines
- start state
- final/accepting states
- implicit error state
- accepted and rejected words
- \( L(M) \) – the language recognized by DFA \( M \)
- Notice that \( L(M) = L(M') \) even though \( M \neq M' \)
Formal Definition

A DFA is a 5-tuple \((\Sigma, Q, q_0, A, \delta)\) where

- finite alphabet \(\Sigma\)
- finite set of states \(Q\)
- start state \(q_0\)
- set of final/accepting states \(A \subseteq Q\)
- transition function: \(\delta : Q \times \Sigma \rightarrow Q\)
DFA Interpreter Algorithm

Input: A word $w = w_1 w_2 ... w_n$, where each $w_i \in \Sigma$
Output: true if accepted, false if rejected
Implementing DFAs

Need to implement the transition function somehow
Where are DFAs used?

Everywhere!

- A microwave is a DFA (probably).
- A Turing machine is a DFA plus infinite memory (magnetic tape)
- A CPU is a DFA plus finite memory (registers)
- A computer is a CPU (DFA) plus a lot of finite memory (RAM)
  - 4GB of RAM has $2^{34359738368}$ states
  - Not a practical way to model a computer