COMPSCI 501: Formal Language Theory
Lecture 2: Deterministic Finite Automata

Marius Minea
marius@cs.umass.edu
University of Massachusetts Amherst

January 25, 2019

Proposed Revised Grading

- **Homework**: 36% (six homeworks)
- **Moodle quizzes**: 4% (throughout semester)
- **Midterm 1**: 20% (Thu Feb 21, 7 pm, ILC S131)
- **Midterm 2**: 20% (Wed Apr 10, 7 pm, ILC S131)
- **Final**: 20% (Thu May 9, 10:30 am, Goessmann 20)

Automata: the simplest computers

Switch: just one bit of memory

- What happens if we press "stop" in "off" state?
  - Should describe behavior completely

Strings and Languages

- **Alphabet** (usually $\Sigma$): any nonempty finite set
- **String**: finite sequence of symbols from the alphabet
- **$\Sigma^*$**: set of all (finite) strings over $\Sigma$, incl. empty string (denoted $\epsilon$).

A language is an arbitrary subset of strings (of $\Sigma^*$)

- **Decision problem** for a language $X$: for a given input string $w \in \Sigma^*$, answer whether $w \in X$

Parity: Even or Odd ?

- We'll use automata to recognize languages.

- Bit strings (0 or 1) with an odd number of ones.
  - $q_0$: initial state; $q_1$: accepting state
Formal Definition

A (deterministic) finite automaton is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\):

- \(Q\) is a finite set of states
- \(\Sigma\) is a finite alphabet (of input symbols)
- \(\delta : Q \times \Sigma \to Q\) is the transition function
- \(q_0 \in Q\) is the start state
- \(F \subseteq Q\) is the set of accept states.

Language of a DFA

Informally: \(M\) accepts string \(w\) if on input \(w\) it ends up in an accepting state.

More precisely: let \(w = w_1 w_2 \ldots w_n\). Then \(M\) accepts \(w\) if there is a sequence of states \(r_0, r_1, \ldots, r_n\) from \(Q\) such that

- \(r_0 = q_0\) (initial state)
- \(\delta(r_i, w_{i+1}) = r_{i+1}\) (transitions on symbols from string)
- \(r_n \in F\) (accept state)

\(L(M)\) (language of \(M\)) = set of strings accepted by \(M\)

Def. A language is called a regular language if some finite automaton recognizes it.

Some Simple Patterns

Strings that

- **start** with a given string
  - if not, go to “dead state” (will never accept)

- **end** with a given string
  - in general, must “remember” last \(k\) symbols (pattern length)

- **contain** a given string (pattern search)

Knuth-Morris-Pratt algorithm constructs DFA for pattern contains/ends with pattern: easy via NFA - DFA construction

Automata for a finite set of strings:

\{"they", "are", "more"\}

\{"this", "is", "his"\}

(Finite State) Transducers

Outputs a string (one output symbol for each input)

a.k.a. Mealy machines

Change in formal definition:

- output alphabet \(\Gamma\)
- transition function: \(\delta : Q \times \Sigma \to Q \times \Gamma\)

Finite State Machines for Testing

Some Fundamental Testing problems:

- Determine the state after a test (homing/distinguishing sequence)
- Verify that \(M\) is in a given state \(s\) (state verification)
- Conformance testing: Given \(M\) (black-box) and a FSM \(S\) (specification), determine whether \(M\) is equivalent to \(S\)
- Machine identification: identify unknown black-box machine \(M\)
Automata Learning
Learn a FSM model of a black-box system
for protocols, security, legacy software

Regular Operations on Languages
If A and B are languages, we define:
Union: \( A \cup B = \{ x \mid w \in A \text{ or } x \in B \} \)
Concatenation: \( A \circ B = \{ xy \mid x \in A \text{ and } y \in B \} \)
(Kleene) Star: \( A^* = \{ x_1x_2 \ldots x_k \mid k \geq 0 \text{ and } x_i \in A \} \)
We’ll see that regular languages are closed under each of these operations.

Product Construction for Union
Let \( M_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1) \) and \( M_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2) \)
Is string \( w \) accepted by \( M_1 \) or \( M_2 \)?
Can only read string once \( \Rightarrow \) run both automata in parallel
know state of \( M_1 \) and \( M_2 \) at each point: pair of states
\( \Rightarrow \) state space is cartesian product \( Q_1 \times Q_2 \)
\( Q = Q_1 \times Q_2 \)
\( \Sigma \) is the same (common) alphabet
\( \delta((r_1, r_2), a) = (\delta(r_1, a), \delta(r_2, a)) \)
\( q_0 = (q_0^1, q_0^2) \)
\( F = \{ (r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2 \} \)
at least one automaton accepts

Other Language Operations
- **Intersection**
  product (like for union), but \( F = F_1 \times F_2 \) (both must accept)
- **Complement**
  same automaton structure, swap accept/non-accept states
- **Concatenation**
  must split string \( w \) into \( x \in L(M_1) \) and \( y \in L(M_2) \): where?
  construction will use nondeterminism