1 Abstract machines

- Basic components of a computer: (i) central processing unit; (ii) memory; (iii) input-output devices.
- We are concerned with abstract machines, i.e. mathematical descriptions of computing machinery.

2 Deterministic finite automata

- A **dfa** consists of a **reading head**, a **tape** and a **finite control**.
- The tape consists of squares which can hold a symbol.
- The finite control consists of a set of finite states and an indicator which shows the current state of the automaton.
- The reading head starts at the leftmost position on the tape. At each turn the reading head advances one square right reading the symbol in its starting position, and the finite control updates the state of the machine according to a **transition function** which maps pairs of symbols and states to states.

  Definition 2.1 (Deterministic finite automaton)**.**
  A dfa is a quintuple $\langle Q, \Sigma, \delta, q_0, F\rangle$ where

  Q is a set of **states**,

  Σ is an **alphabet**,

  $q_0 \in Q$ is the **initial state**,

  $F \subseteq Q$ is the set of **final states**.

  and δ, the **transition function**, is a function from $Q \times \Sigma$ to Q.

  Exercise 2.2.

  Take the fa $M = \langle \{q_0, q_1\}, \{a, b\}, \delta, q_0, \{q_0\}\rangle$ where δ is,

q	σ	$\delta(q, \sigma)$
q_0	a	q_0
q_0	b	q_1
q_1	a	q_1
q_1	b	q_0

  Trace the processing of the string *aabba* by M.

- As made clear by example 2.2 at a certain point of the operation of an fa, in which state the fa will end up when it runs out of input is entirely determined by the current state and the string to be read ahead.

  Call (q, w) a **configuration** of an fa M, where q is the current state and w is the string on the tape including the current symbol and what lies to the right of it.

  For an fa $M = \langle Q, \Sigma, \delta, q_0, F\rangle$ the relation **yields in one step**, defined over set of configurations and designated with $\models_M$, is such that,

  $$(q, w) \models_M (q', w') \text{ iff } w = \sigma w' \text{ for some } \sigma \in \Sigma \text{ and } \delta(q, \sigma) = q'$$

  The relation **yields**, designated as $\models_M^*$, is the reflexive transitive closure of $\models_M$,

  An fa $M = \langle Q, \Sigma, \delta, q_0, F\rangle$ **accepts** a string $w \in \Sigma^*$ iff $(q_0, w) \models_M^* (q_n, \varepsilon)$ for some $q_n \in F$.

  The **language accepted** by M, designated $L(M)$ is the set of strings accepted by M.

- Fa can be conveniently represented by directed graphs called **state diagrams**. Let's draw a state diagram for the language accepted by the fa in example 2.2.

  Exercise 2.3.
  Draw the state diagrams of an accepting dfa for each language below:

  1. $\{w \in \{a, b\} \mid$ each a in w is immediately preceeded and immediately followed by $b\}$

2. $\{w \in \{a,b\} \mid w$ has $abab$ as a substring$\}$.
3. $\{w \in \{a,b\} \mid w$ has odd number of a's and an even number of b's$\}$.
4. $\{w \in \{a,b\} \mid w$ has both ab and ba as substrings$\}$.
5. $\{w \in \{a,b\} \mid w$ does not contain three consecutive b's$\}$.
6. $(ab \cup aba)^*$

3 Non-deterministic finite automata

- A **non-deterministic finite automaton** extends the notion of dfa in the following respects:

  i. δ is no longer required to be a function;

  ii. **empty transitions** are allowed, i.e. the finite control is allowed to advance the state without reading any symbol from the input tape;

  ii. transitions are not limited to symbols, an nfa can read $w \in \Sigma^*$.

  Exercise 3.1.
  Provide an nfa for each language of exercise 2.3.

- Finite automata M_1 and M_2 are **equivalent** if and only if $L(M_1) = L(M_2)$.

  Theorem 3.2.
  For each non-deterministic finite automaton there exists an equivalent deterministic finite automaton.

4 Equivalence of FALs and regular languages

- Finite automata and regular expressions are two alternative ways of characterizing the same class of languages.

  Theorem 4.1 (Kleene)**.**
  A set of strings is a finite automaton language if and only if it is a regular language.

- We will prove by induction only the *regular expression-to-finite automaton* side of the theorem:
  - We start by showing that the three basic regular expressions, $\emptyset$, ε and the unit language **a** for an arbitrary symbol a, have equivalent NFAs.
  - Then we show that finite automata are closed under union, concatenation and closure.

Self study

Sudkamp (1997), Chapter 6 up to Section 6.4 covers most of what we have seen so far. At certain points there are references to context-free grammars; ignore them for now. You may have difficulty in understanding some formal definitions in the book, concentrate on the examples.

References

Sudkamp, T. A. (1997). *Languages and Machines: An Introduction to the Theory of Computer Science*. Addison-Wesley, Reading, MA, 2nd edition edition.