Theory of Computation Notes

paraphrased by Tyler Wright

An important note, these notes are absolutely ***NOT*** *guaranteed to be correct, representative of the course, or rigorous. Any result of this is not the author's fault.*

Contents

1 The Basics of Computation

1.1 Decision Problems

A decision problem is a problem which has a `Yes` or `No` answer.

1.1.1 Decomposing Decision Problems

A decision problem can be decomposed into two sets, the `Yes` and `No` instances of the problem.

1.2 Alphabets

An alphabet is finite set whose members are called symbols (or equivalently letters or characters).

1.2.1 Strings

A string (or equivalently word) over an alphabet Σ is a finite sequence of symbols from Σ. The sequence may be empty, such sequences are denoted by ϵ. The amount of symbols in a string w is denoted by $|w|$.

1.2.2 The Set of Strings

The set of all strings over Σ is denoted by Σ^*.

1.2.3 Substrings and Concatenation

For two strings v, w, v is a substring of w if it appears consecutively in w.

We write vw to denotes v concatenated with w and for k in $\mathbb{Z}_{>0}$, we say v^k is the k-fold concatenation of v with itself (k copies of v).

2 Finite State Automaton

2.1 Deterministic Finite State Automaton

A deterministic finite state automaton (DFA) is a 5-tuple $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ where:

$$\begin{aligned} Q = &\text{ any finite set, called the states,} \\ \Sigma = &\text{ any alphabet,} \\ \delta \in &\{Q \times \Sigma \to Q\} \text{ is the the transition function,} \\ q_0 \in &\, Q \text{ is the initial state,} \\ F \subseteq &\, Q \text{ is the set of accept states.} \end{aligned}$$

We say that M accepts a word w in Σ if there is a sequence of states $r_0, \ldots, r_n$ in Q satisfying:

- $r_0 = q_0$,
- $\delta(r_i, w_{i+1}) = r_{i+1}$,
- r_n is in F.

2.1.1 Product Automaton

For the two DFA:

$$M_1 = \langle Q_1, \Sigma, \delta_1, q_1, F_1 \rangle, M_2 = \langle Q_2, \Sigma, \delta_2, q_2, F_2 \rangle,$$

the product automaton M is:

$$M = M_1 \times M_2 = \langle Q, \Sigma, \delta, q_0, F \rangle,$$

where:

$$\begin{aligned} Q &= Q_1 \times Q_2 \\ \delta((p_1, p_2), a) &= (\delta_1(p_1, a), \delta_2(p_2, a)), \\ q_0 &= (q_1, q_2), \\ F &= F_1 \times F_2. \end{aligned}$$

2.2 Non-deterministic Finite State Automaton

A non-deterministic finite state automaton (NFA) is identical to a DFA except our transition function is from $Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$ where Σ_ϵ is an alphabet Σ with the empty word added.

Transitioning on the empty word doesn't consume a letter of our input word and arbitrary choices are made by the automaton when choices present themselves. We have that a word is accepted in an NFA if and only if there is at least one computation where the word is accepted.

2.3 Languages

For a DFA M, the language set of M denoted by $L(M)$ is the maximal set of words in the alphabet of M such that for each w in $L(M)$, M accepts w. We say M recognises a language A if $L(M) = A$.

2.3.1 Regular Languages

A language is regular if it is recognised by some DFA.

2.3.2 The Intersection of Regular Languages

For the two DFA M_1 and M_2 with languages A and B (resp.), we have that $A \cap B$ is recognised by $M_1 \times M_2$ the product automaton.

2.4 Epsilon Closure

For the NFA $M = \langle Q, \Sigma, \delta, q_0, F \rangle$, and $R \subseteq Q$, we define the epsilon closure of R to be:

$$E(R) := \left\{ q \in Q : \begin{array}{c} \text{where there is a series of transitions solely over} \\ \epsilon \text{ from some } r \text{ in } R \text{ to } q \end{array} \right\}$$

2.4.1 Simulating NFA with DFA

We can simulate an arbitrary NFA:

$$M = \langle Q, \Sigma, \delta, q_0, F \rangle$$

with a DFA:

$$M' = \langle Q', \Sigma_\epsilon, \delta', q_0', F' \rangle$$

where:

$$\begin{aligned} Q' &= \mathcal{P}(Q), \\ \delta'(q, a) &= \{q : \text{for some } r \in R, q \in E(\delta(r, a))\} \\ q_0' &= E(\{q_0\}), \\ F' &= \{q' \in Q' : \text{for some } q \in q', q \in F\}. \end{aligned}$$