Institut for Matematik og Datalogi
Syddansk Universitet

September 11, 2014
JBJ

DM517 – Fall 2014 – Weekly Note 3

Lecture in week 37:

We covered Section 1.4 and Section 2.1 except the last part on Chomsky grammars Key points:

- Not all languages are regular, for example $L = \{a^n b^n : n \geq 0\}$ is not, since a FA cannot remember how many as it has read.
- The Pumping Lemma can be used to prove that a language is not regular. The proof goes by contradiction, so assume that the language in question is regular and apply the lemma to yield a string that should be in the language but is not.
- The closure properties of regular languages (union, complement, intersection, difference, concatenation, and Kleene star) can also be used to prove the nonregularity of a language. One is example is $L = \{w \in \{0,1\}^* | \#_0 = \#_1\}$. Suppose L is regular, then $L' = L \cap 0^*1^*$ is also regular as regular languages are closed under interesection. However, L' is precisely the language $\{o^n 1^n | n \geq 0\}$ which we already know to be non-regular, contradiction. So L cannot be regular.
- Context-free grammars specify context-free languages and are more powerful than regular expressions.
- A derivation of a string in a context-free grammar can be visualized by a parse tree.
- If some string has more than one parse tree in a context-free grammar, the grammar is said to be ambiguous.
- The membership problem for NFAs is as follows: given an NFA $M = (Q, \Sigma, \mathcal{P}(Q), s, F)$ and a string $w \in \Sigma^*$ decide whether M will accept w. If M is a DFA this is trivial, just follow the unique path starting in the initial state and "spelling" w. Accept w precisely if this path ends in an accepting state. Now assume that M is non-deterministic. We can obtain an algorithm for the membership problem by first converting M to an equivalent DFA M' using the algorithm in Section 2.3 and then check whether M' accepts w as above. This may take exponential time as M' may have $2^{|Q|}$ states. There is also an algorithm of complexity $\mathcal{O}(|Q|^3 + |w||Q|^2)$ which works as follows: first calculate, for each $q \in Q$ the set $E(q)$ which can be reached from q via ϵ-transitions. This can be done in time $\mathcal{O}(|Q|^3)$. Now let $w = w_1 w_2 \dots w_{|w|}$ and let $S = E(s)$ (the ϵ-closure of the initial state. Let S' denote all states we can

reach from some $q \in S$ by reading the first symbol w_1 and let $S'' = \bigcup_{r \in S'} E(r)$ denote the ϵ-closure of S'. Now let $S = S''$ and process the next character w_2 of w. Proceeding this way, after w such iterations we have that the current S is precisely those states we can reach from s via w and possibly some ϵ-transitions. It is not hard to check that this algorithm has the claimed complexity.

- Checking whether $L(M) = \emptyset$ for an NFA $M = (Q, \Sigma, \mathcal{P}(Q), s, F)$ is very simple: just check whether there is any path from s to a final state in M (no matter which symbols are on the edges, i.e. ignore these). If there is such a path $L(M) \neq \emptyset$ and otherwise $L(M) = \emptyset$ (note that we may have $L(M) = \{\epsilon\}$ which is NOT the empty set). So in time $\mathcal{O}(|Q|^2)$ we can decide if $L(M) = \emptyset$.

Lecture September 15, 2014:

- Last part of Section 2.1
- Push-down automata. Sipser pages 111-116.
- Non context-free languages. Section 2.3

Exercises September 17, 2014:

- 1.29(a),(b) and 1.30.
- 1.35
- 1.51(a),(c),(d)
- 2.2
- 2.4
- 2.6(b),(d)
- 2.14 and 2.16
- Problem 1 and 2 January 2009.