Compiler Construction (List 3)

Hans de Nivelle

16 October 2013

1. (a) Using the translation scheme on the slides, translate the following regular expression into an NDFA
$$(a|b)^*abbc.$$
   (b) Using the subset construction in the slides, transform the NDFA into a DFA.
   (c) Apply the minimization operation of the slides on the DFA that was you obtained in the previous exercise.
   (d) Same procedure on
$$(\epsilon|aa|bb)^*c.$$

2. (a) Consider the regular expression $\Sigma^*(ananas|apple)$.
   (b) Transform the regular expression into an NDFA.
   (c) Transform the NDFA into a DFA, using the algorithm on the slides.
   (d) Simplify the DFA from the previous task, using the minimization procedure on the slides.

3. Let $\Sigma = \{a, b\}$. Let $\mathcal{L}_n$ be the language of words over Σ that can be written in the form $w_1.a.w_2$, where $w_1, w_2 \in \Sigma^*$, and in addition, w_2 has length n.

   - Show that every DFA recognizing $\mathcal{L}_n$ has at least 2^{n+1} states.
   - Show that there exists a DFA with 2^{n+1} states that recognizes $\mathcal{L}_n$.
   - Show that there exists an NDFA recognizing $\mathcal{L}_n$ with $n + 2$ states.

   (The idea of this exercise comes from Aho/Ullman, Principles of Compiler Design.)

4. Consider the following language:
$$\mathcal{L} = \{a^i b^i | i > 0\}.$$
   Is it regular? If yes, give an (automaton/regular expression). If not, give a proof that shows that $\mathcal{L}$ is not regular.

5. Consider the language

$$= \{a^n | n \text{ is prime}\}.$$

Is it regular? Give either an automaton, or a proof that it is not regular.