2D1371 Automata Theory

EXAMINATION PROBLEMS
WITH SELECTED SOLUTIONS
28 May 2007

Dilian Gurov
KTH CSC
08-790 81 98

1. Convert the nondeterministic automaton given below to an equivalent deterministic one using the subset construction (textbook Lecture 6). Omit inaccessible states. Draw the graph of the resulting DFA.

	a	b
$\rightarrow$ q_0	$\{q_2\}$	$\emptyset$
$\rightarrow$ q_1 F	$\{q_0, q_2\}$	$\emptyset$
q_2 F	$\emptyset$	$\{q_1, q_2\}$

Solution: (as a table)

	a	b
$\rightarrow$ $\{q_0, q_1\}$ F	$\{q_0, q_2\}$	$\emptyset$
$\{q_0, q_2\}$ F	$\{q_2\}$	$\{q_1, q_2\}$
$\{q_1, q_2\}$ F	$\{q_0, q_2\}$	$\{q_1, q_2\}$
$\{q_2\}$ F	$\emptyset$	$\{q_1, q_2\}$
$\emptyset$	$\emptyset$	$\emptyset$

2. For the deterministic automaton given below, apply the minimization algorithm of textbook Lecture 14 to compute the equvalence classes of the collapsing relation $\approx$ defined in textbook Lecture 13.

	a	b
$\rightarrow$ q_0	q_2	q_0
q_1	q_0	q_2
q_2 F	q_4	q_3
q_3	q_0	q_4
q_4 F	q_4	q_1

(a) Show clearly the computation steps (use tables).

(b) List the computed equivalence classes.
Solution: These are $\{q_0\}$, $\{q_1, q_3\}$ and $\{q_2, q_4\}$.

(c) Apply the quotient construction of textbook Lecture 13 to derive the minimized automaton. Draw its graph.
Solution: (as a table)

	a	b
$\rightarrow$ $\{q_0\}$	$\{q_2, q_4\}$	$\{q_0\}$
$\{q_1, q_3\}$	$\{q_0\}$	$\{q_2, q_4\}$
$\{q_2, q_4\}$ F	$\{q_2, q_4\}$	$\{q_1, q_3\}$

3. A context–free grammar $G = (N, \Sigma, P, S)$ is called *strongly right–linear* (or SRLG for short) if all its productions are of shape $A \to aB$ or $A \to \epsilon$. Let us call a SRLG *deterministic* if for each non–terminal $A \in N$ and terminal $a \in \Sigma$ there is exactly one production of shape $A \to aB$ (while productions of shape $A \to \epsilon$ are optional).

   Define a *complement construction* on deterministic SRLGs. That is, for deterministic SRLGs G define the complement $\overline{G}$ as a deterministic SRLG for which you show that $L(\overline{G}) = \overline{L(G)}$.

   Solution: Recall that there is a one–to–one corespondence beween DFAs and SRLGs, and recall the complement construction on DFAs. Let $G = (N, \Sigma, P, S)$ be a deterministic SRLG. We define the complement of G as the deterministic SRLG $\overline{G} \overset{\text{def}}{=} (N, \Sigma, \overline{P}, S)$ where $\overline{P} \overset{\text{def}}{=} \{A \to aB \mid A \to aB \in P\} \cup \{A \to \epsilon \mid A \to \epsilon \notin P\}$ is the set of productions of $\overline{G}$. Then,

$$
\begin{array}{rcll}
x \in L(\overline{G}) & \Leftrightarrow & S \to^{+}_{\overline{G}} x & \{\text{Def. } L(G)\} \\
& \Leftrightarrow & \exists! A \in N.\, (S \to^{+}_{\overline{G}} xA \wedge A \to \epsilon \in \overline{P}) & \left\{\overline{G} \text{ a deterministic SRLG}\right\} \\
& \Leftrightarrow & \exists! A \in N.\, (S \to^{+}_{G} xA \wedge A \to \epsilon \notin P) & \left\{\text{Def. } \overline{G}\right\} \\
& \Leftrightarrow & \text{not } S \to^{+}_{G} x & \{G \text{ a deterministic SRLG}\} \\
& \Leftrightarrow & x \notin L(G) & \{\text{Def. } L(G)\} \\
& \Leftrightarrow & x \in \overline{L(G)} & \{\text{Set theory}\}
\end{array}
$$

   and therefore $L(\overline{G}) = \overline{L(G)}$.

---

4. Recall the Chomsky–Schützenberger Theorem (textbook Supplementary Lecture G). Show how this theorem applies to the context–free language

$$A \overset{\text{def}}{=} \{x \in \{a, b\}^* \mid \sharp a(x) = \sharp b(x)\}$$

   over the alphabet $\Sigma = \{a, b\}$, by identifying:

   - a suitable natural number n,
   - a regular language R over the alphabet Σ_n of the n–th balanced parentheses language, and
   - a homomorphism $h : \Sigma_n \to \Sigma^*$,

   for which you argue that $A = h(\text{PAREN}_n \cap R)$ holds.

   Solution: Recalling that A is generated by the grammar $S \to \epsilon \mid aSb \mid bSa \mid SS$ it is easy to see that $A = h(\text{PAREN}_n \cap R)$ holds for $n = 2$, $R = (\Sigma_n)^*$ and h defined by $h([_1) = a$, $h(]_1) = b$, $h([_2) = b$ and $h(]_2) = a$.

---

5. Consider the language:

$$A = \{a^m b^n \mid m \leq n\}$$

   (a) Use the Pumping Lemma for regular languages (as a game with a Deamon) to prove that A is not regular.

   (b) Refer to the closure properties of context–free languages to argue that A is context–free. That is, represent A as the result of some operation(s) on context–free languages (which we already know to be context–free) under which CFLs are closed.

   Solution: $A = B \cdot C$ for $B \overset{\text{def}}{=} \{a^m b^m \mid m \geq 0\}$ and $C \overset{\text{def}}{=} \{b^n \mid n \geq 0\}$, both of which we know to be context–free, and we know CFLs to be closed under language concatenation.

   (c) Give a context–free grammar G generating A.

   Solution: One possibility is the grammar $S \to \epsilon \mid aSb \mid Sb$.

(d) Prove your grammar correct. You are allowed to reuse results proved in class.

(e) Construct an NPDA accepting $A - \{\epsilon\}$ on empty stack. Explain your choice of productions.

Solution: One solution is a NPDA with one control state q and productions:

$$\begin{aligned}
\langle q, S\rangle &\overset{a}{\hookrightarrow} \langle q, SA\rangle \\
\langle q, S\rangle &\overset{a}{\hookrightarrow} \langle q, A\rangle \\
\langle q, A\rangle &\overset{b}{\hookrightarrow} \langle q, A\rangle \\
\langle q, A\rangle &\overset{b}{\hookrightarrow} \langle q, \epsilon\rangle
\end{aligned}$$

The automaton uses the first production for reading all initial a's but the last, then guesses the last a and uses the second production to get rid of the S at the top of the stack. The stack now has as many A's as the a's read so far. The automaton guesses the number of b's in excess of a's in the string, and uses the third production that many times. The stack now has as many A's as there are remaining (unread) b's. The automaton uses production four to empty the stack upon reading the whole string.

---

6. Let M range over deterministic finite automata (DFA).

(a) Describe a uniform, injective encoding of DFAs into strings over the alphabet $\{0, 1\}$.

Solution: Such an encoding was discussed in class.

(b) Let $\hat{M} \in \{0,1\}^*$ denote the encoding of M. As we know from Cantor's theorem, the set

$$A \overset{\text{def}}{=} \left\{\hat{M} \in \{0,1\}^* \mid \hat{M} \notin L(M)\right\}$$

is not regular (since it is the inverted diagonal set). Argue, however, that A is recursive by giving a (reasonably detailed) description of a total Turing machine accepting A.

Solution: (Sketch) The Turing machine T starts by modifying the input $\hat{M}$ to $\hat{M}\sharp\hat{q_0}\sharp\hat{M}$ where q_0 is the initial state of M. The added part $\hat{q_0}\sharp\hat{M}$ represents the initial configuration of M (where a configuration of a DFA is understood as a pair consisting of a state and the suffix of the input string that remains to be read). T continues by simulating M on $\hat{M}$, by iteratively computing and updating the current state of M until the input string $\hat{M}$ has been completely consumed. This is achieved by reading (and consuming) the next symbol of the input string $\hat{M}$ of M, and looking up from $\hat{M}$ (always available in the first segment of the tape) the next state of M. Finally, T checks whether the state in the final configuration of M is an accepting state, and rejects resp. accepts accordingly. Thus, T is a total Turing machine accepting language A.

---

7. Consider the language

$$VTP \overset{\text{def}}{=} \left\{\hat{M}\sharp\hat{x}\sharp\hat{q} \mid M \text{ run on } x \text{ visits } q \text{ twice}\right\}$$

where "M visits q" means that Turing machine M is at a configuration with control state q. Show that language VTP is not recursive by reducing from undecidability of the Membership Problem. That is, given a total Turing machine M_{VTP} deciding VTP, construct a total Turing machine M_{MP} deciding MP, where:

$$MP \overset{\text{def}}{=} \left\{\hat{M}\sharp\hat{x} \mid M \text{ accepts } x\right\}$$

Solution: Assume there is a total Turing machine M_{VTP} accepting *VTP*. Then we can construct a Turing machine M_{MP} as follows.

On input $\hat{M}\sharp\hat{x}$, M_{MP} modifies the input to $\widehat{M'}\sharp\hat{x}\sharp\hat{v}$ where M' is like M but is modified as follows: two new control states u and v are added, the latter of which is made the new accept state of M', and the transition function δ of M is extended to δ' so that $\delta'(t,a) = (u,\natural,R)$ and $\delta'(u,a) = (t,a,L)$ for all a in the input alphabet of M, and $\delta'(t,\natural) = (v,\natural,R)$, where t is the original accept state of M and $\natural$ is a tape symbol of M_{MP} not used elsewhere. Notice that M' visits state t twice on input x exactly when M visits t on x, that is, when M accepts x. M_{MP} then continues as M_{VTP}.

Then:

$$\begin{aligned}
M_{MP} \text{ accepts } \hat{M}\sharp\hat{x} &\Leftrightarrow M_{VTP} \text{ accepts } \widehat{M'}\sharp\hat{x}\sharp\hat{v} \\
&\Leftrightarrow M' \text{ run on } x \text{ visits } v \text{ twice} \\
&\Leftrightarrow M \text{ accepts } x
\end{aligned}$$

Since M_{VTP} is total, so is M_{MP}, and so M_{MP} decides *MP* which we know to be undecidable. Hence there is no total Turing machine M_{VTP} accepting *VTP*.