DD2372 Automata and Languages

Examination Problems
with Solution Sketches
27 May 2014, 8^{oo} - 13^{oo}

Dilian Gurov
KTH CSC
08-790 81 98

1 Level E

For passing level E you need 8 (out of 10) points from this section.

1. Consider the following DTD, which was a test case for your second laboratory assignment: 3p

```
<!DOCTYPE AddressBook [
     <!ELEMENT ADDRESSBOOK (CONTACT*)>
     <!ELEMENT CONTACT (NAME, PHONE+)>
     <!ELEMENT NAME (\#PCDATA)>
     <!ELEMENT PHONE (\#PCDATA)>
 ]>
```

   (a) Construct a DFA for checking adherence of XML documents to this DTD.

   Solution: Straightforward, following a similar discipline as the inductive transformation of regular expressions into ϵ-NFAs (DFA omitted).

   (b) In the laboratory assignment you used NPDAs for the general case. Explain why a DFA suffices for the task above.

   Solution: Each element type has as arguments only element types that are defined strictly below its own definition. This results naturally in a regular language.

2. Consider a black-box system with two buttons, a and b, an indicator lamp, and a test-and-reset button. Consider the following observation set: 3p

$$\{(\epsilon, -), (a, -), (b, +), (aa, -), (ab, +), (ba, -), (bb, +)\}$$

   Use *regular inference* to construct a minimal DFA consistent with this observation set, considering observation prefixes as histories, observation postfixes as experiments, and states as sets of histories not distinguished by any existing experiment.

   Solution: Considering all observation prefixes against all observation postfixes (table omitted here), one can group the prefixes into two sets of indistinguishable histories:

$$\{\epsilon, a, aa, ba\} \text{ and } \{b, ab, bb\}$$

   which form the two states of our hypothesis automaton, with short representatives ϵ and b, respectively. Using the short representatives to find the transitions of the automaton, we arrive at the following DFA, given as a table:

		a	b
$\rightarrow$	$[\epsilon]$	$[\epsilon]$	$[b]$
$\star$	$[b]$	$[\epsilon]$	$[b]$

3. Consider the following context-free grammar: 4p

$$S \to \epsilon \mid aSb \mid bSa \mid SS$$

(a) Which language does this grammar generate?

Solution: The grammar generates the set of strings over $\{a, b\}$ that have an equal number of occurrences of a's and b's (recall homework assignment 4).

(b) Show a *parse tree* for the string $abab$. How many derivations correspond to this parse tree? Show the leftmost one.

Solution: There is a parse tree giving rise to 6 derivations (tree omitted), the leftmost one being:

$$S \Rightarrow SS \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSb \Rightarrow abab$$

(c) Is the parse tree for $abab$ unique? If not, show all other parse trees.

Solution: There are ininitely many parse trees: one can apply the last production any number of times, producing any number of S-nonterminals, and then the effect of all but two of them can be "cancelled" by applying the first production (trees omitted).

---

2 Level C

For grade D you need to have passed level E and obtained 5 (out of 10) points from this section. For passing level C you need 8 points from this section.

1. What is the most straightforward way of converting a DFA into an equivalent NPDA that you can think of? Formalize your construction and show that it is language-preserving. 4p

Solution: One very simple and direct idea is to convert the DFA into an NPDA with a dummy stack symbol (the start symbol) that is just rewritten to itself, and a control automaton that is isomorphic to the DFA, accepting on final states. Formalizing this idea, let $D = (Q, \Sigma, \delta_D, q_0, F)$ be a DFA. Define $P \stackrel{\text{def}}{=} (Q, \Sigma, \{S\}, \delta_P, q_0, S, F)$, where $\delta_P(q, a, S) \stackrel{\text{def}}{=} \{(\delta_D(q, a), S)\}$. Language preservation is easily established by proving and using the following Lemma: For any $q, q' \in Q$ we have:

$$(q, x \cdot y, S) \stackrel{*}{\vdash} (q', y, S) \;\Leftrightarrow\; \hat{\delta}_D(q, x) = q'$$

2. Recall the *Chomsky-Schützenberger Theorem* (see lecture notes). Show how this theorem applies to the language A defined as: 4p

$$A \stackrel{\text{def}}{=} \left\{ a^l b^k a^m \mid k = 2l + m \right\}$$

That is, identify:

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

for which you argue that $A = h(PAREN_n \cap R)$ holds.

Solution: Observing that the strings of A can be presented in the shape $a^l b^{2l} b^m a^m$, we can suggest the use of two types of parentheses, namely $n \stackrel{\text{def}}{=} 2$, R defined by the regular expression $[_1^*\,]_1^*\, [_2^*\,]_2^*$, and h defined as $h([_1) \stackrel{\text{def}}{=} a$, $h(]_1) \stackrel{\text{def}}{=} bb$, $h([_2) \stackrel{\text{def}}{=} b$ and $h(]_2) \stackrel{\text{def}}{=} a$.

3. Is it decidable whether the languages of two *Turing machines* are disjoint? Justify your answer. [2p]

   Solution: The problem is not decidable. If it were, we could fix the second Turing machine to one that accepts exactly one string, say *abba*, which is straightforward to construct, and we could then use this to decide the problem of whether a Turing machine accepts *abba*. But the latter problem is not decidable, so the former one cannot be decidable either.

---

3 Level A

For grade B you need to have passed level C and obtained 5 (out of 10) points from this section. For grade A you need 8 points from this section.

1. When modelling system behaviour as a DFA, we argued that it is natural to consider all system states as accepting, except for one so-called "trap" state, which captures that one has violated the behaviour of the system. The language of a DFA modelling a system would then by necessity be prefix-closed (i.e., every prefix of every string in the language is itself in the language). Now, justify this intuition by proving that: *The minimal DFA for any regular prefix-closed language* $A \subseteq \Sigma^*$ *has a unique non-accepting state.* [5p]

   Hint: one can refer to the *Myhill-Nerode Theorem* and automaton to show this, based on the relation $\equiv_A$ defined by:

   $$x \equiv_A y \overset{\text{def}}{\iff} \forall z \in \Sigma^*.\ (x \cdot z \in A \Leftrightarrow y \cdot z \in A)$$

   Solution: Let $A \subseteq \Sigma^*$ be regular and prefix-closed. Let $x, y \in \Sigma^*$ be two arbitrary strings such that $x, y \notin A$. Since A is prefix-closed, for any $z \in \Sigma^*$ we must have that $x \cdot z \notin A$ and $y \cdot z \notin A$. Therefore $x \equiv_A y$ whenever $x, y \notin A$. Thus, the Myhill-Nerode automaton for the language A, which is the minimal DFA accepting A, will have a unique non-accepting state, namely the equivalence class $\overline{A}$.

2. Why cannot one determinize NPDAs by defining a construction similar to the subset construction for NFAs, but applied to the control automaton of the NPDA? Explain your reasoning on an example, but try to be general. [5p]

   Solution: In a nutshell, the subset construction can only take into account the next letter of the input alphabet and the top of the stack symbol, but cannot handle properly the stack symbols that are pushed on the stack.

   Would things be different if we had a bound k on the length of the strings to be recognized?

   Solution: Yes, but only if we assume that there are no ϵ-transitions. We could then just construct the graph of the configurations of the NPDA that can be reached on all inputs of length up to k. This graph must be finite, and viewing it as an NFA, we can apply the subset construction on it to obtain an equivalent DFA.
