CIS 570 — Introduction to Formal Language Theory — Fall 2007

Final Examination

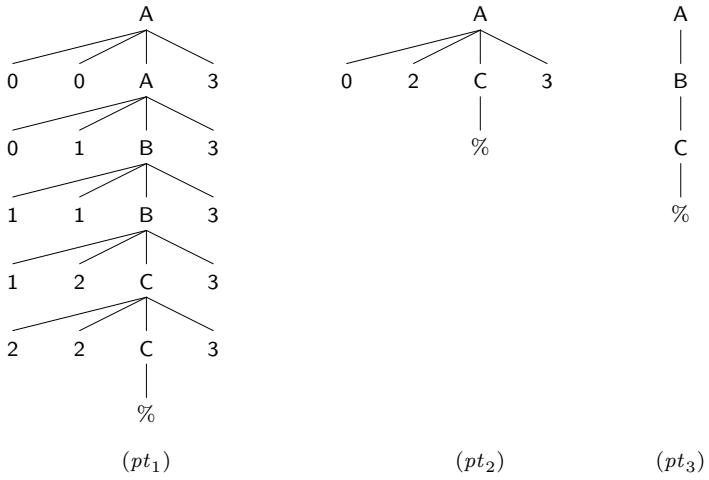
Model Answers

Question 1

(a)

$$\mathsf{A} \to \mathsf{00A3} \mid \mathsf{01B3} \mid \mathsf{02C3} \mid \mathsf{B},$$
$$\mathsf{B} \to \mathsf{11B3} \mid \mathsf{12C3} \mid \mathsf{C},$$
$$\mathsf{C} \to \mathsf{22C3} \mid \%.$$

(b) Let pt_1, pt_2 and pt_3 be:



(pt_1) (pt_2) (pt_3)

We have that $\mathbf{yield}\, pt_1 = \mathsf{000111122233333}$, $\mathbf{yield}\, pt_2 = \mathsf{023}$ and $\mathbf{yield}\, pt_3 = \%$.

Question 2

First, we carry out four standard declarations:

```
- val efaToDFA = nfaToDFA o efaToNFA;
val efaToDFA = fn : efa -> dfa
- val regToEFA = faToEFA o regToFA;
val regToEFA = fn : reg -> efa
- val regToDFA = efaToDFA o regToEFA;
val regToDFA = fn : reg -> dfa
- val minAndRen = DFA.renameStatesCanonically o DFA.minimize;
val minAndRen = fn : dfa -> dfa
```

Next, we declare `allStrDFA` to be a DFA accepting $\{0,1\}^*$:

```
- val allStrDFA = minAndRen(regToDFA(Reg.fromString "(0 + 1)*"));
val allStrDFA = - : dfa
```

Next, we declare DFAs `has000011DFA` and `has110000DFA` accepting $\{\, w \in \{0,1\}^* \mid 000011$ is a substring of $w \,\}$ and $\{\, w \in \{0,1\}^* \mid 110000$ is a substring of $w \,\}$, respectively:

```
- val has000011DFA = minAndRen(regToDFA(Reg.fromString "(0+1)*000011(0+1)*"));
val has000011DFA = - : dfa
- val has110000DFA = minAndRen(regToDFA(Reg.fromString "(0+1)*110000(0+1)*"));
val has110000DFA = - : dfa
```

Next, we declare a DFA `hasNot000011DFA` accepting $\{\, w \in \{0,1\}^* \mid 000011$ is not a substring of $w \,\}$:

```
- val hasNot000011DFA = DFA.minus(allStrDFA, has000011DFA);
val hasNot000011DFA = - : dfa
```

Next, we declare an EFA `efa` accepting X:

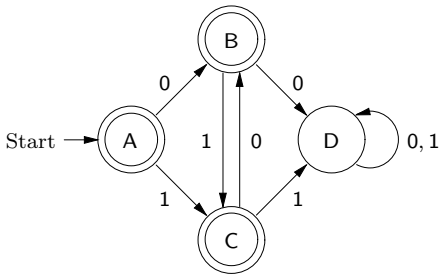
```
- val efa = EFA.union(injDFAToEFA hasNot000011DFA, injDFAToEFA has110000DFA);
val efa = - : efa
```

Finally, we declare a minimized DFA `dfa` accepting X:

```
- val dfa = minAndRen(efaToDFA efa);
val dfa = - : dfa
```

Question 3

(a)



(b) First, we note that, for all $w \in \{0,1\}^*$:

- $w \in X$ iff 00 is not a substring of w and 11 is not a substring of w;
- $w \notin X$ iff 00 is a substring of w or 11 is a substring of w.

Next, we use induction on Λ to prove that:

(A) For all $w \in \Lambda_{\mathsf{A}}$, $w = \%$.

(B) For all $w \in \Lambda_{\mathsf{B}}$, $w \in X$ and $\mathsf{0}$ is a suffix of w.

(C) For all $w \in \Lambda_{\mathsf{C}}$, $w \in X$ and $\mathsf{1}$ is a suffix of w.

(D) For all $w \in \Lambda_{\mathsf{D}}$, $w \notin X$.

There are nine steps to show.

- (empty string) We have that $\% = \%$.
- $(\mathsf{A}, \mathsf{0} \to \mathsf{B})$ Suppose $w \in \Lambda_{\mathsf{A}}$, and assume the inductive hypothesis, $w = \%$. Thus $w\mathsf{0} = \mathsf{0} \in X$ and $\mathsf{0}$ is a suffix of $w\mathsf{0}$.
- $(\mathsf{A}, \mathsf{1} \to \mathsf{C})$ Suppose $w \in \Lambda_{\mathsf{A}}$, and assume the inductive hypothesis, $w = \%$. Thus $w\mathsf{1} = \mathsf{1} \in X$ and $\mathsf{1}$ is a suffix of $w\mathsf{1}$.
- $(\mathsf{B}, \mathsf{0} \to \mathsf{D})$ Suppose $w \in \Lambda_{\mathsf{B}}$, and assume the inductive hypothesis, $w \in X$ and $\mathsf{0}$ is a suffix of w. Because $\mathsf{0}$ is a suffix of w, we have that $\mathsf{00}$ is a suffix of $w\mathsf{0}$. Thus $w\mathsf{0} \notin X$.
- $(\mathsf{B}, \mathsf{1} \to \mathsf{C})$ Suppose $w \in \Lambda_{\mathsf{B}}$, and assume the inductive hypothesis, $w \in X$ and $\mathsf{0}$ is a suffix of w. Because $w \in X$ and $\mathsf{0}$ is a suffix of w, we have that $w\mathsf{1} \in X$. And $\mathsf{1}$ is a suffix of $w\mathsf{1}$.
- $(\mathsf{C}, \mathsf{0} \to \mathsf{B})$ Suppose $w \in \Lambda_{\mathsf{C}}$, and assume the inductive hypothesis, $w \in X$ and $\mathsf{1}$ is a suffix of w. Because $w \in X$ and $\mathsf{1}$ is a suffix of w, we have that $w\mathsf{0} \in X$. And $\mathsf{0}$ is a suffix of $w\mathsf{0}$.
- $(\mathsf{C}, \mathsf{1} \to \mathsf{D})$ Suppose $w \in \Lambda_{\mathsf{C}}$, and assume the inductive hypothesis, $w \in X$ and $\mathsf{1}$ is a suffix of w. Because $\mathsf{1}$ is a suffix of w, we have that $\mathsf{11}$ is a suffix of $w\mathsf{1}$. Thus $w\mathsf{1} \notin X$.
- $(\mathsf{D}, \mathsf{0} \to \mathsf{D})$ Suppose $w \in \Lambda_{\mathsf{D}}$ (so $w \in \{\mathsf{0}, \mathsf{1}\}^*$), and assume the inductive hypothesis, $w \notin X$. Because $w \notin X$, we have that $w\mathsf{0} \notin X$.
- $(\mathsf{D}, \mathsf{1} \to \mathsf{D})$ Suppose $w \in \Lambda_{\mathsf{D}}$ (so $w \in \{\mathsf{0}, \mathsf{1}\}^*$), and assume the inductive hypothesis, $w \notin X$. Because $w \notin X$, we have that $w\mathsf{1} \notin X$.

Now, we use the result of our induction on Λ to prove that $L(M) = X$.

- $(L(M) \subseteq X)$ Suppose $w \in L(M)$. Hence $w \in L(M) = \Lambda_{\mathsf{A}} \cup \Lambda_{\mathsf{B}} \cup \Lambda_{\mathsf{C}}$. Thus, there are three cases to consider.
  - Suppose $w \in \Lambda_{\mathsf{A}}$. By Part (A), we have that $w = \% \in X$.
  - Suppose $w \in \Lambda_{\mathsf{B}}$. By Part (B), we have that $w \in X$.
  - Suppose $w \in \Lambda_{\mathsf{C}}$. By Part (C), we have that $w \in X$.
- $(X \subseteq L(M))$ Suppose $w \in X$. Since $X \subseteq \{\mathsf{0}, \mathsf{1}\}^*$, we have that $w \in \{\mathsf{0}, \mathsf{1}\}^*$. Suppose, toward a contradiction, that $w \notin L(M)$. Thus $w \notin L(M) = \Lambda_{\mathsf{A}} \cup \Lambda_{\mathsf{B}} \cup \Lambda_{\mathsf{C}}$. But $w \in \{\mathsf{0}, \mathsf{1}\}^* = (\textbf{alphabet}\, M)^* = \Lambda_{\mathsf{A}} \cup \Lambda_{\mathsf{B}} \cup \Lambda_{\mathsf{C}} \cup \Lambda_{\mathsf{D}}$, so that $w \in \Lambda_{\mathsf{D}}$. Thus by Part (D), we have that $w \notin X$—contradiction. Thus $w \in L(M)$.

Question 4

The two pumping lemmas have similar structure. With PLRL, one gives a regular language L to the lemma, gets back a natural number n, gives the lemma a string $z \in L$ of length at least n, and gets back a splitting uvw of z such that (1) $|uv| \leq n$, (2) $v \neq \%$ and (3) $uv^i w \in L$ for all $i \in \mathbb{N}$. With PLCFL, one gives a context-free language L to the lemma, gets back a natural number n, gives the lemma a string $z \in L$ of length at least n, and gets back a splitting $uvwxy$ of z such that (1) $|vwx| \leq n$, (2) $vx \neq \%$ and (3) $uv^i wx^i y \in L$ for all $i \in \mathbb{N}$.

PLRL is easier to work with because, by picking z carefully, we can often control what the string v to be pumped consists of. For instance, when proving that $L = \{ \mathsf{0}^n\mathsf{1}^n \mid n \in \mathbb{N} \}$ is not regular, we can pick z to be $\mathsf{0}^n\mathsf{1}^n$, thus forcing v to consist of between 1 and n $\mathsf{0}$'s. On the other hand, with PLCFL, vwx could be any substring of z with length between 1 and n, and we don't know which of v and x (possibly both) are non-empty. But this still give us *some* control over what vx can be. For instance, when proving that $L = \{ \mathsf{0}^n\mathsf{1}^n\mathsf{2}^n \mid n \in \mathbb{N} \}$ is not context-free, by picking z to be $\mathsf{0}^n\mathsf{1}^n\mathsf{2}^n$, we can stop vx from having both $\mathsf{0}$'s and $\mathsf{2}$'s.