CIS 570 — Introduction to Formal Language Theory — Fall 2007

Exercise Set 5

Model Answers

Exercise 1

First, we let the DFA $M'$ be $\text{determinSimplify}(M, \emptyset)$. $M'$ is the same as $M$. Next, we construct the set $X$ of pairs of states of $M'$, as follows.

First, we add to $X$ all pairs consisting of an accepting state and a non-accepting state: $(A, B)$, $(B, A)$, $(A, D)$, $(D, A)$, $(B, C)$, $(C, B)$, $(C, D)$ and $(D, C)$. Now we must handle each of these 8 pairs.

Since there are no $(0, 0)$-transitions leading into $B$, nothing can be added to $X$ using $(A, B)$, $(B, A)$ and $(0, 0)$-transitions. Since there are no $(1, 0)$-transitions leading into $A$, nothing can be added to $X$ using $(A, B)$, $(B, A)$ and $(1, 0)$-transitions.

Since $(A, D)$ and $(D, A)$ are in $X$, and $(A, 0, A)$, $(C, 0, A)$, $(B, 0, D)$ and $(D, 0, D)$ are the $(0, 0)$-transitions leading into $A$ and $D$, we would add the pairs $(A, B)$, $(B, A)$, $(A, D)$, $(D, A)$, $(B, C)$, $(C, B)$, $(C, D)$ and $(D, C)$ to $X$, if they weren’t already there. Since there are no $(1, 0)$-transitions leading into $A$, nothing can be added to $X$ using $(A, D)$, $(D, A)$ and $(1, 0)$-transitions.

Since there are no $(0, 0)$-transitions leading into $B$, nothing can be added to $X$ using $(B, C)$, $(C, B)$ and $(1, 0)$-transitions. Since $(B, C)$ and $(C, B)$ are in $X$, and $(A, 1, B)$, $(B, 1, C)$ and $(D, 1, C)$ are the $(1, 0)$-transitions leading into $B$ and $C$, we would add the pairs $(A, B)$, $(B, A)$, $(A, D)$ and $(D, A)$ to $X$, if they weren’t already there.

Since there are no $(0, 0)$-transitions leading into $C$, nothing can be added to $X$ using $(C, D)$, $(D, C)$ and $(0, 0)$-transitions. Since $(C, D)$ and $(D, C)$ are in $X$, and $(B, 1, C)$, $(D, 1, D)$ and $(C, 1, D)$ are the $(0, 0)$-transitions leading into $C$ and $D$, we would add the pairs $(B, C)$, $(C, B)$, $(D, C)$ and $(C, D)$ to $X$, if they weren’t already there.

We have now handled all of the elements of $X$ that were initially added to $X$, using rules (1) and (2). And no new elements were added to $X$, and so we have that $X$ consists of the following 8 pairs: $(A, B)$, $(A, D)$, $(B, A)$, $(B, C)$, $(C, B)$, $(C, D)$, $(D, A)$ and $(D, C)$. Thus the set $Y$ consists of the following 8 pairs: $(A, A)$, $(A, C)$, $(B, B)$, $(B, D)$, $(C, A)$, $(C, C)$, $(D, B)$ and $(D, D)$. Hence the set $Z$ consists of the following equivalence classes: $\{A, C\}$ and $\{B, D\}$.

Hence $N$ has the following states: $(A, C)$ and $(B, D)$. Since $A$ is the start state of $N$, we have that the start state of $N$ is $(A, C)$. Since $A$ and $C$ are the accepting states of $M'$, we have that $(A, C)$ is the only accepting state of $N$. It remains to compute the transitions of $N$.

Since $\{A, C\} \in Z$, and $\delta_M(A, 0) = \{A\} = \{A, C\}$, we have that $\langle (A, C), 0, (A, C) \rangle \in T_N$. Since $\{A, C\} \in Z$, and $\delta_M(A, 1) = \{B\} = \{B, D\}$, we have that $\langle (A, C), 1, (B, D) \rangle \in T_N$.

Since $\{B, D\} \in Z$, and $\delta_M(B, 0) = \{D\} = \{B, D\}$, we have that $\langle (B, D), 0, (B, D) \rangle \in T_N$. Since $\{B, D\} \in Z$, and $\delta_M(B, 1) = \{C\} = \{A, C\}$, we have that $\langle (B, D), 1, (A, C) \rangle \in T_N$.

Here is a drawing of $N$:
To check that our final answer is correct, we put the text

\[
\text{states} \\
A, B, C, D \\
\text{start state} \\
A \\
\text{accepting states} \\
A, C \\
\text{transitions} \\
A, 0 \rightarrow A; A, 1 \rightarrow B; B, 0 \rightarrow D; B, 1 \rightarrow C; \\
C, 0 \rightarrow A; C, 1 \rightarrow D; D, 0 \rightarrow D; D, 1 \rightarrow C
\]

in the file es5-ex1-dfa. Then we invoke Forlan and proceed as follows:

```ml
- val dfa = DFA.input "es5-ex1-dfa";
val dfa = _ : dfa
- val dfa' = DFA.minimize dfa;
val dfa' = _ : dfa
- DFA.output("", dfa');
{states}
\langle A,C\rangle, \langle B,D\rangle
{start state}
\langle A,C\rangle
{accepting states}
\langle A,C\rangle
{transitions}
\langle A,C\rangle, 0 \rightarrow \langle A,C\rangle; \langle A,C\rangle, 1 \rightarrow \langle B,D\rangle; \langle B,D\rangle, 0 \rightarrow \langle B,D\rangle; \langle B,D\rangle, 1 \rightarrow \langle A,C\rangle
val it = () : unit
```

It is easy to check that the outputted DFA is \(N\).

**Exercise 2**

Since

\[
\text{Btw}(1,1,0) = \{\%, 1\}, \\
\text{Btw}(1,2,0) = \{0\}, \\
\text{Btw}(2,1,0) = \{1\}, \\
\text{Btw}(2,2,0) = \{\%, 0\},
\]

we have that

\[
\text{btw}(1,1,0) = \text{simp}(\% + 1) = \% + 1, \\
\text{btw}(1,2,0) = \text{simp}(0) = 0, \\
\text{btw}(2,1,0) = \text{simp}(1) = 1, \\
\text{btw}(2,2,0) = \text{simp}(\% + 0) = \% + 0.
\]
Thus

\[
\text{btw}(1,2,1) = simp(\text{btw}(1,2,0) + \text{btw}(1,1,0) \text{btw}(1,1,0)^* \text{btw}(1,2,0))
= simp(0 + (% + 1)(% + 1)^0)
= 1^0,
\]
\[
\text{btw}(2,2,1) = simp(\text{btw}(2,2,0) + \text{btw}(2,1,0) \text{btw}(1,1,0)^* \text{btw}(1,2,0))
= simp((% + 0) + 1(% + 1)^0)
= % + 0 + 11^0,
\]
\[
\text{btw}(1,2,2) = simp(\text{btw}(1,2,1) + \text{btw}(1,2,1) \text{btw}(2,2,1)^* \text{btw}(2,2,1))
= simp(1*0 + (1*0)(% + 0 + 11*0)^*(% + 0 + 11*0))
= 1^0(0 + 11^0)^* ,
\]
\[
\alpha = simp(\text{btw}(1,2,2))
= simp(1*0(0 + 11^0)^*)
= 1^0(0 + 11^0)^* .
\]

Here is the Forlan transcript showing the above simplifications:

```
- val simp = Reg.simplify Reg.weakSubset;
  val simp = fn : reg -> reg
  - fun outSimp s = Reg.output("", simp(Reg.fromString s));
  val outSimp = fn : string -> unit
  - outSimp "% + 1";
    % + 1
    val it = () : unit
  - outSimp "0";
    0
    val it = () : unit
  - outSimp "1";
    1
    val it = () : unit
  - outSimp "% + 0";
    % + 0
    val it = () : unit
  - outSimp "0 + (%+1)(%+1)*0";
    1*0
    val it = () : unit
  - outSimp "(%+0) + 1(%+1)*0";
    % + 0 + 11*0
    val it = () : unit
  - outSimp "1*0 + (1*0)(%+0+11*0)*(%+0+11*0)";
    1*0(0 + 11*0)*
    val it = () : unit
  - outSimp "1*0(0 + 11*0)*";
    1*0(0 + 11*0)*
    val it = () : unit
```
To check that our final answer is correct, we put the text

```ml
{states}
A, B
{start state}
A
{accepting states}
B
{transitions}
A, 0 -> B; A, 1 -> A;
B, 0 -> B; B, 1 -> A
```

in the file `es5-ex2-fa`, and then proceed as follows.

```ml
- val fa = FA.input "es5-ex2-fa";
- val reg = - : reg
- val fa = FA.input "es5-ex2-fa";
B, 0 -> B; B, 1 -> A
A, 0 -> B; A, 1 -> A;
{transitions}
{accepting states}
{states}
```

**Exercise 3**

(a) Define functions `HasSuf : {0,1,2}^* \to \text{Lan}`, `HasNotSuf : {0,1,2}^* \to \text{Lan}`, `HasPref : {0,1,2}^* \to \text{Lan}`, `HasNotPref : {0,1,2}^* \to \text{Lan}` and `NotSur : {0,1,2}^* \times {0,1,2}^* \times {0,1,2}^* \to \text{Lan}` by:

- for all \( x \in \{0,1,2\}^* \), `HasSuf(x) = \{ w \in \{0,1,2\}^* \mid x \text{ is a suffix of } w \}`;
- for all \( x \in \{0,1,2\}^* \), `HasNotSuf(x) = \{ w \in \{0,1,2\}^* \mid x \text{ is not a suffix of } w \}`;
- for all \( x \in \{0,1,2\}^* \), `HasPref(x) = \{ w \in \{0,1,2\}^* \mid x \text{ is a prefix of } w \}`;
- for all \( x \in \{0,1,2\}^* \), `HasNotPref(x) = \{ w \in \{0,1,2\}^* \mid x \text{ is not a prefix of } w \}`;
- for all \( x, y, z \in \{0,1,2\}^* \), `NotSur(x, y, z) = \{ w \in \{0,1,2\}^* \mid \text{there are } u, v \in \{0,1,2\}^* \text{ such that } w = uyyv \text{ and either } x \text{ is not a suffix of } u \text{ or } z \text{ is not a prefix of } v \}`.

**Lemma ES5.3.1**

1. For all \( x \in \{0,1,2\}^* \), \( \text{HasSuf}(x) = \{0,1,2\}^* \{x\} \).
2. For all \( x \in \{0,1,2\}^* \), \( \text{HasNotSuf}(x) = \{0,1,2\}^* \setminus \text{HasSuf}(x) \).
3. For all \( x \in \{0,1,2\}^* \), \( \text{HasPref}(x) = \{x\} \{0,1,2\}^* \).
4. For all \( x \in \{0,1,2\}^* \), \( \text{HasNotPref}(x) = \{0,1,2\}^* \setminus \text{HasPref}(x) \).
5. For all \( x, y, z \in \{0,1,2\}^* \), \( \text{NotSur}(x, y, z) = \text{HasNotSuf}(x) \{y\} \{0,1,2\}^* \cup \{0,1,2\}^* \{y\} \text{HasNotPref}(z) \).
(6) For all \(x, y, z \in \{0, 1, 2\}^*\), \(\text{Sur}(x, y, z) = \{0, 1, 2\}^* - \text{NotSur}(x, y, z)\).

Proof.

(1) Suppose \(x \in \{0, 1, 2\}^*\). We must show that \(\text{HasSuf}(x) = \{0, 1, 2\}^* \{x\}\). It will suffice to show that \(\text{HasSuf}(x) \subseteq \{0, 1, 2\}^* \{x\} \subseteq \text{HasSuf}(x)\).

Suppose \(w \in \text{HasSuf}(x)\). Then \(w \in \{0, 1, 2\}^*\) and \(x\) is a suffix of \(w\). Thus \(w = ux\) for some \(u \in \{0, 1, 2\}^*\), so that \(w = ux \in \{0, 1, 2\}^* \{x\}\).

Suppose \(w \in \{0, 1, 2\}^* \{x\}\). Then \(w = ux\) for some \(u \in \{0, 1, 2\}^*\). Hence \(x\) is a suffix of \(w\), so that \(w \in \text{HasSuf}(x)\).

(2) Suppose \(x \in \{0, 1, 2\}^*\). We must show that \(\text{HasNotSuf}(x) = \{0, 1, 2\}^* - \text{HasSuf}(x)\). It will suffice to show that \(\text{HasNotSuf}(x) \subseteq \{0, 1, 2\}^* - \text{HasSuf}(x) \subseteq \text{HasNotSuf}(x)\).

Suppose \(w \in \text{HasNotSuf}(x)\). Then \(w \in \{0, 1, 2\}^*\) and \(x\) is not a suffix of \(w\). Suppose, toward a contradiction, that \(w \in \text{HasSuf}(x)\). Then \(x\) is a suffix of \(w\) — contradiction. Thus \(w \notin \text{HasSuf}(x)\), completing the proof that \(w \in \{0, 1, 2\}^* - \text{HasSuf}(x)\).

Suppose \(w \in \{0, 1, 2\}^* - \text{HasSuf}(x)\). Then \(w \in \{0, 1, 2\}^*\) and \(w \notin \text{HasSuf}(x)\). Suppose, toward a contradiction, that \(x\) is a suffix of \(w\). Then \(w \in \text{HasSuf}(x)\) — contradiction. Hence \(x\) is not a suffix of \(w\), completing the proof that \(w \in \text{HasNotSuf}(x)\).

(3) Suppose \(x \in \{0, 1, 2\}^*\). We must show that \(\text{HasPref}(x) = \{x\} \{0, 1, 2\}^*\). It will suffice to show that \(\text{HasPref}(x) \subseteq \{x\} \{0, 1, 2\}^* \subseteq \text{HasPref}(x)\).

Suppose \(w \in \text{HasPref}(x)\). Then \(w \in \{0, 1, 2\}^*\) and \(x\) is a prefix of \(w\). Thus \(w = xu\) for some \(u \in \{0, 1, 2\}^*\), so that \(w = xu \in \{x\} \{0, 1, 2\}^*\).

Suppose \(w \in \{x\} \{0, 1, 2\}^*\). Then \(w = xu\) for some \(u \in \{0, 1, 2\}^*\). Hence \(x\) is a prefix of \(w\), so that \(w \in \text{HasPref}(x)\).

(4) Suppose \(x \in \{0, 1, 2\}^*\). We must show that \(\text{HasNotPref}(x) = \{0, 1, 2\}^* - \text{HasPref}(x)\). It will suffice to show that \(\text{HasNotPref}(x) \subseteq \{0, 1, 2\}^* - \text{HasPref}(x) \subseteq \text{HasNotPref}(x)\).

Suppose \(w \in \text{HasNotPref}(x)\). Then \(w \in \{0, 1, 2\}^*\) and \(x\) is not a prefix of \(w\). Suppose, toward a contradiction, that \(w \in \text{HasPref}(x)\). Then \(x\) is a prefix of \(w\) — contradiction. Thus \(w \notin \text{HasPref}(x)\), completing the proof that \(w \in \{0, 1, 2\}^* - \text{HasPref}(x)\).

Suppose \(w \in \{0, 1, 2\}^* - \text{HasPref}(x)\). Then \(w \in \{0, 1, 2\}^*\) and \(w \notin \text{HasPref}(x)\). Suppose, toward a contradiction, that \(x\) is a prefix of \(w\). Then \(w \in \text{HasPref}(x)\) — contradiction. Hence \(x\) is not a prefix of \(w\), completing the proof that \(w \in \text{HasNotPref}(x)\).

(5) Suppose \(x, y, z \in \{0, 1, 2\}^*\). We must show that

\[
\text{NotSur}(x, y, z) = \text{HasSuf}(x)\{y\}{0, 1, 2}^* \cup {0, 1, 2}^*\{y\}\text{HasNotPref}(z).
\]

It will suffice to show that

\[
\text{NotSur}(x, y, z) \subseteq \text{HasSuf}(x)\{y\}{0, 1, 2}^* \cup {0, 1, 2}^*\{y\}\text{HasNotPref}(z) \\
\subseteq \text{NotSur}(x, y, z).
\]
Suppose \( w \in \text{NotSur}(x, y, z) \). Thus \( w \in \{0, 1, 2\}^* \) and there are \( u, v \in \{0, 1, 2\}^* \) such that \( w = uv \) and either \( x \) is not a suffix of \( u \) or \( z \) is not a prefix of \( v \). There are two cases to consider.

- Suppose \( x \) is not a suffix of \( u \). Thus \( u \in \text{HasNotSuf}(x) \), so that \( w = uv \in \text{HasNotSuf}(x) \{y\} \{0, 1, 2\}^* \). Hence \( w \in \text{HasNotSuf}(x) \{y\} \{0, 1, 2\}^* \cup \{0, 1, 2\}^* \{y\} \text{HasNotPref}(z) \).
- Suppose \( z \) is not a prefix of \( v \). Thus \( v \in \text{HasNotPref}(z) \), so that \( w = uv \in \{0, 1, 2\}^* \{y\} \text{HasNotPref}(z) \). Hence \( w \in \text{HasNotSuf}(x) \{y\} \{0, 1, 2\}^* \cup \{0, 1, 2\}^* \{y\} \text{HasNotPref}(z) \).

Suppose \( w \in \text{HasNotSuf}(x) \{y\} \{0, 1, 2\}^* \cup \{0, 1, 2\}^* \{y\} \text{HasNotPref}(z) \). There are two cases to consider.

- Suppose \( w \in \text{HasNotSuf}(x) \{y\} \{0, 1, 2\}^* \). Then \( w = uv \) for some \( u \in \text{HasNotSuf}(x) \) and \( v \in \{0, 1, 2\}^* \). Hence \( u \in \{0, 1, 2\}^* \) and \( x \) is not a suffix of \( u \), so that \( w \in \text{NotSur}(x, y, z) \).
- Suppose \( w \in \{0, 1, 2\}^* \{y\} \text{HasNotPref}(z) \). Then \( w = uv \) for some \( u \in \{0, 1, 2\}^* \) and \( v \in \text{HasNotPref}(z) \). Hence \( v \in \{0, 1, 2\}^* \) and \( z \) is not a prefix of \( v \), so that \( w \in \{0, 1, 2\}^* \). Thus \( w = uv \) and either \( x \) is not a suffix of \( u \) or \( z \) is not a prefix of \( v \), so that \( w \in \text{NotSur}(x, y, z) \).

(6) Suppose \( x, y, z \in \{0, 1, 2\}^* \). We must show that \( \text{Sur}(x, y, z) = \{0, 1, 2\}^* - \text{NotSur}(x, y, z) \). It will suffice to show that \( \text{Sur}(x, y, z) \subseteq \{0, 1, 2\}^* - \text{NotSur}(x, y, z) \subseteq \text{Sur}(x, y, z) \).

Suppose \( w \in \text{Sur}(x, y, z) \). Then \( w \in \{0, 1, 2\}^* \) and, (\( \emptyset \)) for all \( u, v \in \{0, 1, 2\}^* \) if \( w = uv \), then \( x \) is a suffix of \( u \) and \( z \) is a prefix of \( v \). Suppose, toward a contradiction, that \( w \in \text{NotSur}(x, y, z) \). Then there are \( u, v \in \{0, 1, 2\}^* \) such that \( w = uv \) and either \( x \) is not a suffix of \( u \) or \( z \) is not a prefix of \( v \). But this contradicts (\( \emptyset \)). Thus \( w \notin \text{NotSur}(x, y, z) \), completing the proof that \( w \in \{0, 1, 2\}^* - \text{NotSur}(x, y, z) \).

Suppose \( w \in \{0, 1, 2\}^* - \text{NotSur}(x, y, z) \). Thus \( w \in \{0, 1, 2\}^* \) and \( w \notin \text{NotSur}(x, y, z) \). To see that \( w \in \text{Sur}(x, y, z) \), suppose \( u, v \in \{0, 1, 2\}^* \) and \( w = uv \). We must show that \( x \) is a suffix of \( u \) and \( z \) is a prefix of \( v \). Suppose, toward a contradiction, that \( x \) is not a suffix of \( u \). Then \( w = uv \) and either \( x \) is not a suffix of \( u \) or \( z \) is not a prefix of \( v \), so that \( w \in \text{NotSur}(x, y, z) \)—contradiction. Thus \( x \) is a suffix of \( u \). Suppose, toward a contradiction, that \( z \) is not a prefix of \( v \). Then \( w = uv \) and either \( x \) is not a suffix of \( u \) or \( z \) is not a prefix of \( v \), so that \( w \in \text{NotSur}(x, y, z) \)—contradiction. Thus \( z \) is a prefix of \( v \).

\[\square\]

Next, we define some useful functions. Define \( \text{faToDFA} \in \text{FA} \rightarrow \text{DFA} \) by:

\[
\text{faToDFA} = \text{nfaToDFA} \circ \text{efToNFA} \circ \text{faToEFA}.
\]

Then we have that, for all \( M \in \text{FA} \),

\[
L(\text{faToDFA}(M)) = L(\text{nfaToDFA}(\text{efToNFA}(\text{faToEFA}(M)))) = L(\text{efToNFA}(\text{faToEFA}(M))) = L(\text{faToEFA}(M)) = L(M).
\]
Define \( \text{regToDFA} \in \text{Reg} \rightarrow \text{DFA} \) by:

\[
\text{regToDFA} = \text{faToDFA} \circ \text{regToFA}.
\]

Then we have that, for all \( \alpha \in \text{Reg} \),

\[
L(\text{regToDFA}(\alpha)) = L(\text{faToDFA}(\text{regToFA}(\alpha))) = L(\text{regToFA}(\alpha)) = L(\alpha).
\]

Define \( \text{minAndRen} \in \text{DFA} \rightarrow \text{DFA} \) by: for all \( M \in \text{DFA} \),

\[
\text{minAndRen}(M) = \text{renameStatesCanonically}(\text{minimize}(M)).
\]

Then, for all \( M \in \text{FA} \),

\[
L(\text{minAndRen}(M)) = L(\text{renameStatesCanonically}(\text{minimize}(M)))
= L(\text{minimize}(M)) = L(M).
\]

Define the DFA \( \text{allStrDFA} \) by:

\[
\text{allStrDFA} = \text{minAndRen}(\text{regToDFA}((0 + 1 + 2)^*)).
\]

Then, we have that

\[
L(\text{allStrDFA}) = L(\text{minAndRen}(\text{regToDFA}((0 + 1 + 2)^*)))
= L(\text{regToDFA}((0 + 1 + 2)^*))
= L((0 + 1 + 2)^*) = \{0, 1, 2\}^*.
\]

Let the FA \( \text{allStrFA} \) be \( \text{allStrDFA} \). Thus \( L(\text{allStrFA}) = L(\text{allStrDFA}) = \{0, 1, 2\}^* \).

Define \( \text{hasSufFA} \in \{0, 1, 2\}^* \rightarrow \text{FA} \) by: for all \( x \in \{0, 1, 2\}^* \),

\[
\text{hasSufFA}(x) = \text{concat}(\text{allStrFA}, \text{strToFA}(x)).
\]

Then, we have that, for all \( x \in \{0, 1, 2\}^* \),

\[
L(\text{hasSufFA}(x)) = L(\text{concat}(\text{allStrFA}, \text{strToFA}(x)))
= L(\text{allStrFA}) \, L(\text{strToFA}(x))
= \{0, 1, 2\}^* \{x\}
= \text{HasSuf}(x),
\]

by Lemma ES5.3.1(1).

Define \( \text{hasSufDFA} \in \{0, 1, 2\}^* \rightarrow \text{DFA} \) by: for all \( x \in \{0, 1, 2\}^* \),

\[
\text{hasSufDFA}(x) = \text{minAndRen}(\text{faToDFA}(\text{hasSufFA}(x))).
\]

Then, we have that, for all \( x \in \{0, 1, 2\}^* \),

\[
L(\text{hasSufDFA}(x)) = L(\text{minAndRen}(\text{faToDFA}(\text{hasSufFA}(x))))
= L(\text{faToDFA}(\text{hasSufFA}(x)))
= L(\text{hasSufFA}(x))
= \text{HasSuf}(x).
\]
Define $\text{hasNotSufDFA} \in \{0, 1, 2\}^* \rightarrow \text{DFA}$ by: for all $x \in \{0, 1, 2\}^*$, 

$$\text{hasNotSufDFA}(x) = \text{minus}(\text{allStrDFA}, \text{hasSufDFA}(x)).$$

Then, we have that, for all $x \in \{0, 1, 2\}^*$,

$$L(\text{hasNotSufDFA}(x)) = L(\text{minus}(\text{allStrDFA}, \text{hasSufDFA}(x)))$$

$$= L(\text{allStrDFA}) - L(\text{hasSufDFA}(x))$$

$$= \{0, 1, 2\}^* - \text{HasSuf}(x)$$

$$= \text{HasNotSuf}(x),$$

by Lemma ES5.3.1(2).

Define $\text{hasNotSufFA} \in \{0, 1, 2\}^* \rightarrow \text{FA}$ by: for all $x \in \{0, 1, 2\}^*$, $\text{hasNotSufFA}(x) = \text{hasNotSufDFA}(x)$. Then, for all $x \in \{0, 1, 2\}^*$,

$$L(\text{hasNotSufFA}(x)) = L(\text{hasNotSufDFA}(x)) = \text{HasNotSuf}(x).$$

Define $\text{hasPrefFA} \in \{0, 1, 2\}^* \rightarrow \text{FA}$ by: for all $x \in \{0, 1, 2\}^*$,

$$\text{hasPrefFA}(x) = \text{concat}(\text{strToFA}(x), \text{allStrFA}).$$

Then, we have that, for all $x \in \{0, 1, 2\}^*$,

$$L(\text{hasPrefFA}(x)) = L(\text{concat}(\text{strToFA}(x), \text{allStrFA}))$$

$$= L(\text{strToFA}(x)) L(\text{allStrFA})$$

$$= \{x\} \{0, 1, 2\}^*$$

$$= \text{HasPref}(x),$$

by Lemma ES5.3.1(3).

Define $\text{hasPrefDFA} \in \{0, 1, 2\}^* \rightarrow \text{DFA}$ by: for all $x \in \{0, 1, 2\}^*$,

$$\text{hasPrefDFA}(x) = \text{minAndRen}(\text{faToDFA}(\text{hasPrefFA}(x))).$$

Then, we have that, for all $x \in \{0, 1, 2\}^*$,

$$L(\text{hasPrefDFA}(x)) = L(\text{minAndRen}(\text{faToDFA}(\text{hasPrefFA}(x))))$$

$$= L(\text{faToDFA}(\text{hasPrefFA}(x)))$$

$$= L(\text{hasPrefFA}(x))$$

$$= \text{HasPref}(x).$$

Define $\text{hasNotPrefDFA} \in \{0, 1, 2\}^* \rightarrow \text{DFA}$ by: for all $x \in \{0, 1, 2\}^*$,

$$\text{hasNotPrefDFA}(x) = \text{minus}(\text{allStrDFA}, \text{hasPrefDFA}(x)).$$

Then, we have that, for all $x \in \{0, 1, 2\}^*$,

$$L(\text{hasNotPrefDFA}(x)) = L(\text{minus}(\text{allStrDFA}, \text{hasPrefDFA}(x)))$$

$$= L(\text{allStrDFA}) - L(\text{hasPrefDFA}(x))$$

$$= \{0, 1, 2\}^* - \text{HasPref}(x)$$

$$= \text{HasNotPref}(x),$$
by Lemma ES5.3.1(4).

Define \( \text{hasNotPrefFA} \in \{0, 1, 2\}^* \rightarrow \text{FA} \) by: for all \( x \in \{0, 1, 2\}^* \), \( \text{hasNotPrefFA}(x) = \text{hasNotPrefDFA}(x) \). Then, for all \( x \in \{0, 1, 2\}^* \),

\[
L(\text{hasNotPrefFA}(x)) = L(\text{hasNotPrefDFA}(x)) = \text{HasNotPref}(x).
\]

Define \( \text{notSurFA} \in \{0, 1, 2\}^* \times \{0, 1, 2\}^* \times \{0, 1, 2\}^* \rightarrow \text{FA} \) by: for all \( x, y, z \in \{0, 1, 2\}^* \),

\[
\text{notSurFA}(x, y, z) = \text{union}(\text{concat}(\text{hasNotSufFA}(x), \text{concat}(\text{strToFA}(y), \text{allStrFA})),
\text{concat}(\text{allStrFA}, \text{concat}(\text{strToFA}(y), \text{hasNotPrefFA}(z))))).
\]

Then, we have that, for all \( x, y, z \in \{0, 1, 2\}^* \),

\[
L(\text{notSurFA}(x, y, z)) = L(\text{union}(\text{concat}(\text{hasNotSufFA}(x), \text{concat}(\text{strToFA}(y), \text{allStrFA})),
\text{concat}(\text{allStrFA}, \text{concat}(\text{strToFA}(y), \text{hasNotPrefFA}(z))))))
\]

\[
= L(\text{concat}(\text{hasNotSufFA}(x), \text{concat}(\text{strToFA}(y), \text{allStrFA})))) \cup
L(\text{concat}(\text{allStrFA}, \text{concat}(\text{strToFA}(y), \text{hasNotPrefFA}(z))))
\]

\[
= L(\text{hasNotSufFA}(x)) L(\text{strToFA}(y)) L(\text{allStrFA}) \cup
L(\text{allStrFA}) L(\text{strToFA}(y)) L(\text{hasNotPrefFA}(z))
\]

\[
= \text{HasNotSuf}(x) \{y\} \{0, 1, 2\}^* \cup \{0, 1, 2\}^* \{y\} \text{HasNotPref}(z)
\]

\[
= \text{NotSur}(x, y, z).
\]

by Lemma ES5.3.1(5).

Define \( \text{notSurDFA} \in \{0, 1, 2\}^* \times \{0, 1, 2\}^* \times \{0, 1, 2\}^* \rightarrow \text{DFA} \) by: for all \( x, y, z \in \{0, 1, 2\}^* \),

\[
\text{notSurDFA}(x, y, z) = \text{minAndRen}(\text{faToDFA}(\text{notSurFA}(x, y, z))).
\]

Then, we have that, for all \( x, y, z \in \{0, 1, 2\}^* \),

\[
L(\text{notSurDFA}(x, y, z)) = L(\text{minAndRen}(\text{faToDFA}(\text{notSurFA}(x, y, z))))
\]

\[
= L(\text{faToDFA}(\text{notSurFA}(x, y, z)))
\]

\[
= L(\text{notSurFA}(x, y, z))
\]

\[
= \text{NotSur}(x, y, z).
\]

Finally define \( \text{surDFA} \in \{0, 1, 2\}^* \times \{0, 1, 2\}^* \times \{0, 1, 2\}^* \rightarrow \text{DFA} \) by: for all \( x, y, z \in \{0, 1, 2\}^* \),

\[
\text{surDFA}(x, y, z) = \text{minAndRen}(\text{minAndRen}(\text{allStrDFA}, \text{notSurDFA}(x, y, z))).
\]

Then we have that, for all \( x, y, z \in \{0, 1, 2\}^* \),

\[
L(\text{surDFA}(x, y, z)) = L(\text{minAndRen}(\text{minAndRen}(\text{allStrDFA}, \text{notSurDFA}(x, y, z))))
\]

\[
= L(\text{minAndRen}(\text{allStrDFA}, \text{notSurDFA}(x, y, z)))
\]

\[
= L(\text{allStrDFA}) - L(\text{notSurDFA}(x, y, z))
\]

\[
= \{0, 1, 2\}^* - \text{NotSur}(x, y, z)
\]

\[
= \text{Sur}(x, y, z),
\]
by Lemma ES5.3.1(6), and \( \text{surDFA}(x, y, z) \) has as few states as possible, because the last step in its definition is \( \text{minAndRen} \).

(b) First, we put the text

\[
\begin{align*}
\text{val faToDFA} &= \text{nfaToDFA} \circ \text{efaNFA} \circ \text{faToEFA}; \\
\text{val regToDFA} &= \text{faToDFA} \circ \text{regToFA}; \\
\text{val minAndRen} &= \text{DFA.renameStatesCanonically} \circ \text{DFA.minimize}; \\
\text{val allStrDFA} &= \text{minAndRen}(\text{regToDFA}(\text{Reg.fromString } "(0 + 1 + 2)\star")); \\
\text{val allStrFA} &= \text{injDFAToFA} \circ \text{allStrDFA}; \\
\text{fun hasSufFA} \ x &= \text{FA.concat}(\text{allStrFA}, \text{strToFA} \ x); \\
\text{val hasSufDFA} &= \text{minAndRen} \circ \text{faToDFA} \circ \text{hasSufFA}; \\
\text{fun hasNotSufDFA} \ x &= \text{DFA.minus}(\text{allStrDFA}, \text{hasSufDFA} \ x); \\
\text{val hasNotSufFA} &= \text{injDFAToFA} \circ \text{hasNotSufDFA}; \\
\text{fun hasPrefFA} \ x &= \text{FA.concat}(\text{strToFA} \ x, \text{allStrFA}); \\
\text{val hasPrefDFA} &= \text{minAndRen} \circ \text{faToDFA} \circ \text{hasPrefFA}; \\
\text{fun hasNotPrefDFA} \ x &= \text{DFA.minus}(\text{allStrDFA}, \text{hasPrefDFA} \ x); \\
\text{val hasNotPrefFA} &= \text{injDFAToFA} \circ \text{hasNotPrefDFA}; \\
\text{fun notSurFA}(x,y,z) &= \\
\text{FA.union}(\text{FA.concat}(\text{hasNotSufFA} \ x, \\
\text{FA.concat}(\text{strToFA} \ y, \\
\text{FA.concat}(\text{allStrFA}, \\
\text{FA.concat}(\text{strToFA} y, \\
\text{hasNotPrefFA} z))), \\
\text{FA.concat}(\text{allStrFA}, \\
\text{FA.concat}(\text{strToFA} y, \\
\text{hasNotPrefFA} z)))); \\
\text{val notSurDFA} &= \text{minAndRen} \circ \text{faToDFA} \circ \text{notSurFA}; \\
\text{fun surDFA}(x, y, z) &= \text{minAndRen}(\text{DFA.minus}(\text{allStrDFA}, \text{notSurDFA}(x, y, z))); \\
\end{align*}
\]

in the file \texttt{sur.sml}. Then we invoke Forlan and proceed as follows:

\[
\begin{align*}
- \text{use } "\text{sur.sml}"; \\
[\text{opening \texttt{sur.sml}}] \\
\text{val faToDFA} = fn : fa \rightarrow dfa \\
\text{val regToDFA} = fn : reg \rightarrow dfa \\
\text{val minAndRen} = fn : dfa \rightarrow dfa \\
\text{val allStrDFA} = - : dfa \\
\text{val allStrFA} = - : fa \\
\text{val hasSufFA} = fn : str \rightarrow fa \\
\text{val hasSufDFA} = fn : str \rightarrow dfa \\
\text{val hasNotSufDFA} = fn : str \rightarrow dfa \\
\text{val hasNotSufFA} = fn : str \rightarrow fa \\
\text{val hasPrefFA} = fn : str \rightarrow fa \\
\text{val hasPrefDFA} = fn : str \rightarrow dfa \\
\text{val hasNotPrefDFA} = fn : str \rightarrow dfa \\
\text{val hasNotPrefFA} = fn : str \rightarrow fa \\
\end{align*}
\]
val notSurFA = fn : str * str * str -> fa
val notSurDFA = fn : str * str * str -> dfa
val surDFA = fn : str * str * str -> dfa
val it = () : unit

val dfa = surDFA(Str.fromString "00", Str.fromString "11", Str.fromString "22");

val dfa = dfa
- DFA.output("", dfa);

{states}
A, B, C, D, E, F, G, H
{start state}
C
{accepting states}
A, B, C, E, F
{transitions}
A, 0 -> A; A, 1 -> E; A, 2 -> C; B, 0 -> A; B, 1 -> F; B, 2 -> C; C, 0 -> B;
C, 1 -> F; C, 2 -> C; D, 0 -> H; D, 1 -> H; D, 2 -> C; E, 0 -> B; E, 1 -> G;
E, 2 -> C; F, 0 -> B; F, 1 -> H; F, 2 -> C; G, 0 -> H; G, 1 -> H; G, 2 -> D;
H, 0 -> H; H, 1 -> H; H, 2 -> H
val it = () : unit

Here is a drawing of dfa: