

# CS 591 S2—Formal Language Theory: Integrating Experimentation and Proof—Fall 2018

## Problem Set 5

### Model Answers

#### Problem 1

First, we let the DFA  $M'$  be **determSimplify**( $M, \emptyset$ ).  $M'$  is the same as  $M$ , except that the dead state  $A$  of  $M$  has been renamed to  $\langle \text{dead} \rangle$ .

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:  $(B, \langle \text{dead} \rangle)$ ,  $(\langle \text{dead} \rangle, B)$ ,  $(B, C)$ ,  $(C, B)$ ,  $(D, \langle \text{dead} \rangle)$ ,  $(\langle \text{dead} \rangle, D)$ ,  $(D, C)$  and  $(C, D)$ . Now, we must handle each of these 8 pairs.

Since there are no transitions leading into  $B$ , no pairs can be added using the pairs  $(B, \langle \text{dead} \rangle)$ ,  $(\langle \text{dead} \rangle, B)$ ,  $(B, C)$  and  $(C, B)$ .

Next, we handle the pairs  $(D, \langle \text{dead} \rangle)$  and  $(\langle \text{dead} \rangle, D)$ . Since there are no 0-transitions leading into  $D$ , nothing can be added to  $X$  using  $(D, \langle \text{dead} \rangle)$ ,  $(\langle \text{dead} \rangle, D)$  and 0-transitions. Since  $(C, 1, D)$ ,  $(B, 1, \langle \text{dead} \rangle)$ ,  $(D, 1, \langle \text{dead} \rangle)$  and  $(\langle \text{dead} \rangle, 1, \langle \text{dead} \rangle)$ , are the 1-transitions leading into  $D$  and  $\langle \text{dead} \rangle$ , we add the pairs  $(C, \langle \text{dead} \rangle)$  and  $(\langle \text{dead} \rangle, C)$  to  $X$ . We would also add the pairs  $(C, B)$ ,  $(B, C)$ ,  $(C, D)$  and  $(D, C)$  to  $X$ , if they were not already there.

Since there are no 0-transitions leading into  $D$ , nothing can be added to  $X$  using  $(D, C)$ ,  $(C, D)$  and 0-transitions. And, since there are no 1-transitions leading into  $C$ , nothing can be added to  $X$  using  $(D, C)$ ,  $(C, D)$  and 1-transitions.

We have now handled all of the original elements of  $X$ . Next, we must handle the elements of  $X$  that we subsequently added:  $(C, \langle \text{dead} \rangle)$  and  $(\langle \text{dead} \rangle, C)$ . Since  $(B, 0, C)$ ,  $(D, 0, C)$ ,  $(C, 0, \langle \text{dead} \rangle)$  and  $(\langle \text{dead} \rangle, 0, \langle \text{dead} \rangle)$  are the 0-transitions leading into  $C$  and  $\langle \text{dead} \rangle$ , we would add the pairs  $(B, C)$ ,  $(C, B)$ ,  $(B, \langle \text{dead} \rangle)$ ,  $(\langle \text{dead} \rangle, B)$ ,  $(D, C)$ ,  $(C, D)$ ,  $(D, \langle \text{dead} \rangle)$  and  $(\langle \text{dead} \rangle, D)$  to  $X$ , if they were not already there. Since there are no 1-transitions leading into  $C$ , nothing can be added to  $X$  using  $(C, \langle \text{dead} \rangle)$ ,  $(\langle \text{dead} \rangle, C)$  and 1-transitions.

We have now handled all of the pairs that we have added to  $X$ . Thus,  $X$  consists of the following 10 pairs:  $(B, C)$ ,  $(B, \langle \text{dead} \rangle)$ ,  $(C, B)$ ,  $(C, D)$ ,  $(C, \langle \text{dead} \rangle)$ ,  $(D, C)$ ,  $(D, \langle \text{dead} \rangle)$ ,  $(\langle \text{dead} \rangle, B)$ ,  $(\langle \text{dead} \rangle, C)$  and  $(\langle \text{dead} \rangle, D)$ .

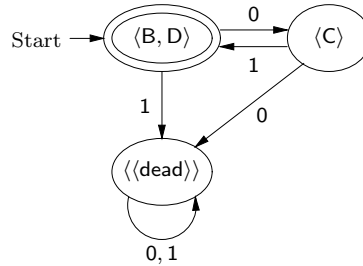
Thus the set  $Y$  consists of the following 6 pairs:  $(B, B)$ ,  $(C, C)$ ,  $(D, D)$ ,  $(\langle \text{dead} \rangle, \langle \text{dead} \rangle)$ ,  $(B, D)$  and  $(D, B)$ . Thus the set  $Z$  consists of the following equivalence classes:  $\{B, D\}$ ,  $\{C\}$  and  $\{\langle \text{dead} \rangle\}$ . Hence the DFA  $N$  has the following states:  $\langle B, D \rangle$ ,  $\langle C \rangle$  and  $\langle \langle \text{dead} \rangle \rangle$ . Since  $B$  is the start state of  $M'$  and  $[B] = \{B, D\}$ , we have that the start state of  $N$  is  $\langle B, D \rangle$ . Since  $\{B, D\} \in Z$  and  $B \in A_{M'}$ , we have that  $\langle B, D \rangle \in A_N$ . And, since  $C$  and  $\langle \text{dead} \rangle$  are not accepting states of  $M'$ , there are no more accepting states of  $N$ . It remains to compute the transitions of  $N$ .

Since  $\{B, D\} \in Z$  and  $[\delta_{M'}(B, 0)] = [C] = \{C\}$ , we have that  $(\langle B, D \rangle, 0, \langle C \rangle) \in T_N$ . Since  $\{B, D\} \in Z$  and  $[\delta_{M'}(B, 1)] = [\langle \text{dead} \rangle] = \{\langle \text{dead} \rangle\}$ , we have that  $(\langle B, D \rangle, 1, \langle \langle \text{dead} \rangle \rangle) \in T_N$ .

Since  $\{C\} \in Z$  and  $[\delta_{M'}(C, 0)] = [\langle \text{dead} \rangle] = \{\langle \text{dead} \rangle\}$ , we have that  $(\langle C \rangle, 0, \langle \langle \text{dead} \rangle \rangle) \in T_N$ . Since  $\{C\} \in Z$  and  $[\delta_{M'}(C, 1)] = [D] = \{B, D\}$ , we have that  $(\langle C \rangle, 1, \langle B, D \rangle) \in T_N$ .

Since  $\{\langle\text{dead}\rangle\} \in Z$  and  $[\delta_{M'}(\langle\text{dead}\rangle, 0)] = [\langle\text{dead}\rangle] = \{\langle\text{dead}\rangle\}$ , we have that  $(\langle\langle\text{dead}\rangle\rangle, 0, \langle\langle\text{dead}\rangle\rangle) \in T_N$ . Since  $\{\langle\text{dead}\rangle\} \in Z$  and  $[\delta_{M'}(\langle\text{dead}\rangle, 1)] = [\langle\text{dead}\rangle] = \{\langle\text{dead}\rangle\}$ , we have that  $(\langle\langle\text{dead}\rangle\rangle, 1, \langle\langle\text{dead}\rangle\rangle) \in T_N$ .

Here is a drawing of  $N$ :



To check that our final answer is correct, we put the text

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

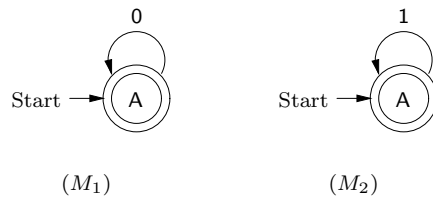
in the file `ps5-p1-dfa.txt`. Then we invoke Forlan and proceed as follows:

```
- val dfa = DFA.input "ps5-p1-dfa.txt";
val dfa = - : dfa
- val dfa' = DFA.minimize dfa;
val dfa' = - : dfa
- DFA.output("", dfa');
{states} <C>, <B,D>, <<dead>> {start state} <B,D> {accepting states} <B,D>
{transitions}
<C>, 0 -> <<dead>>; <C>, 1 -> <B,D>; <B,D>, 0 -> <C>; <B,D>, 1 -> <<dead>>;
<<dead>>, 0 -> <<dead>>; <<dead>>, 1 -> <<dead>>
val it = () : unit
```

It is easy to see that the outputted DFA is  $N$ .

## Problem 2

Let the DFAs  $M_1$  and  $M_2$  be:



Then

$$\begin{aligned}
L(N_1) &= L(\mathbf{minus}(M_1, M_2)) = L(M_1) - L(M_2) = \{0\}^* - \{1\}^* = \{0\}\{0\}^*, \\
L(N_2) &= L(\mathbf{inter}(M_1, \mathbf{complement}(M_2, \emptyset))) = L(M_1) \cap L(\mathbf{complement}(M_2, \emptyset)) \\
&= L(M_1) \cap ((\mathbf{alphabet}(L(M_2)) \cup \emptyset)^* - L(M_2)) = \{0\}^* \cap (\{1\}^* - \{1\}^*) \\
&= \{0\}^* \cap \emptyset = \emptyset.
\end{aligned}$$

Thus  $N_1$  and  $N_2$  are not equivalent.

We can use Forlan to check that our answer is correct, as follows:

```

- val dfa1 = DFA.input "";
@ {states} A {start state} A {accepting states} A {transitions} A, 0 -> A
@ .
val dfa1 = - : dfa
- val dfa2 = DFA.input "";
@ {states} A {start state} A {accepting states} A {transitions} A, 1 -> A
@ .
val dfa2 = - : dfa
- val dfa'1 = DFA.minus(dfa1, dfa2);
val dfa'1 = - : dfa
- val dfa'2 = DFA.inter(dfa1, DFA.complement(dfa2, SymSet.fromString ""));
val dfa'2 = - : dfa
- DFA.relationship(dfa'1, dfa'2);
first language is a proper superset of second language: "0" is in first language
but is not in second language
val it = () : unit

```

### Problem 3

(a) Define functions  $\mathbf{HasSub} \in \{0, 1\}^* \rightarrow \mathbf{Lan}$ ,  $\mathbf{HasNotSub} \in \{0, 1\}^* \rightarrow \mathbf{Lan}$ ,  $\mathbf{HasSubs} \in \{0, 1\}^* \times \{0, 1\}^* \rightarrow \mathbf{Lan}$  and  $\mathbf{HasNotSubs} \in \{0, 1\}^* \times \{0, 1\}^* \rightarrow \mathbf{Lan}$  by:

- for all  $x \in \{0, 1\}^*$ ,  $\mathbf{HasSub} x = \{w \in \{0, 1\}^* \mid x \text{ is a substring of } w\}$ ;
- for all  $x \in \{0, 1\}^*$ ,  $\mathbf{HasNotSub} x = \{w \in \{0, 1\}^* \mid x \text{ is not a substring of } w\}$ ;
- for all  $x, y \in \{0, 1\}^*$ ,

$$\mathbf{HasSubs}(x, y) = \{w \in \{0, 1\}^* \mid x \text{ is a substring of } w \text{ and } y \text{ is a substring of } w\};$$

- for all  $x, y \in \{0, 1\}^*$ ,

$$\mathbf{HasNotSubs}(x, y) = \{w \in \{0, 1\}^* \mid x \text{ is not a substring of } w \text{ and } y \text{ is not a substring of } w\}.$$

#### Lemma PS5.3.1

(1) For all  $x \in \{0, 1\}^*$ ,  $\mathbf{HasSub} x = \{0, 1\}^* \{x\} \{0, 1\}^*$ .

(2) For all  $x \in \{0, 1\}^*$ ,  $\mathbf{HasNotSub} x = \{0, 1\}^* - \mathbf{HasSub} x$ .

- (3) For all  $x, y \in \{0, 1\}^*$ ,  $\mathbf{HasSubs}(x, y) = \mathbf{HasSub} x \cap \mathbf{HasSub} y$ .
- (4) For all  $x, y \in \{0, 1\}^*$ ,  $\mathbf{HasNotSubs}(x, y) = \mathbf{HasNotSub} x \cap \mathbf{HasNotSub} y$ .
- (5) For all  $x, y \in \{0, 1\}^*$ ,  $\mathbf{IFF}(x, y) = \mathbf{HasSubs}(x, y) \cup \mathbf{HasNotSubs}(x, y)$ .

**Proof.**

- (1) Suppose  $x \in \{0, 1\}^*$ . We must show that  $\mathbf{HasSub} x = \{0, 1\}^* \{x\} \{0, 1\}^*$ . It will suffice to show that  $\mathbf{HasSub} x \subseteq \{0, 1\}^* \{x\} \{0, 1\}^* \subseteq \mathbf{HasSub} x$ .
- Suppose  $w \in \mathbf{HasSub} x$ . Then  $w \in \{0, 1\}^*$  and  $x$  is a substring of  $w$ . Thus  $w = u xv$  for some  $u, v \in \{0, 1\}^*$ , so that  $w = u xv \in \{0, 1\}^* \{x\} \{0, 1\}^*$ .
  - Suppose  $w \in \{0, 1\}^* \{x\} \{0, 1\}^*$ . Then  $w = u xv$  for some  $u, v \in \{0, 1\}^*$ . Hence  $w \in \{0, 1\}^*$  and  $x$  is a substring of  $w$ , so that  $w \in \mathbf{HasSub} x$ .
- (2) Suppose  $x \in \{0, 1\}^*$ . We must show that  $\mathbf{HasNotSub} x = \{0, 1\}^* - \mathbf{HasSub} x$ . It will suffice to show that  $\mathbf{HasNotSub} x \subseteq \{0, 1\}^* - \mathbf{HasSub} x \subseteq \mathbf{HasNotSub} x$ .
- Suppose  $w \in \mathbf{HasNotSub} x$ . Then  $w \in \{0, 1\}^*$  and  $x$  is not a substring of  $w$ . Suppose, toward a contradiction, that  $w \in \mathbf{HasSub} x$ . Then  $x$  is a substring of  $w$ —contradiction. Thus  $w \notin \mathbf{HasSub} x$ , completing the proof that  $w \in \{0, 1\}^* - \mathbf{HasSub} x$ .
  - Suppose  $w \in \{0, 1\}^* - \mathbf{HasSub} x$ . Then  $w \in \{0, 1\}^*$  and  $w \notin \mathbf{HasSub} x$ . Suppose, toward a contradiction, that  $x$  is a substring of  $w$ . Then  $w \in \mathbf{HasSub} x$ —contradiction. Hence  $x$  is not a substring of  $w$ , completing the proof that  $w \in \mathbf{HasNotSub} x$ .
- (3) Suppose  $x, y \in \{0, 1\}^*$ . We must show that  $\mathbf{HasSubs}(x, y) = \mathbf{HasSub} x \cap \mathbf{HasSub} y$ . It will suffice to show that  $\mathbf{HasSubs}(x, y) \subseteq \mathbf{HasSub} x \cap \mathbf{HasSub} y \subseteq \mathbf{HasSubs}(x, y)$ .
- Suppose  $w \in \mathbf{HasSubs}(x, y)$ . Then  $w \in \{0, 1\}^*$ ,  $x$  is a substring of  $w$ , and  $y$  is a substring of  $w$ . Thus  $w \in \mathbf{HasSub} x$  and  $w \in \mathbf{HasSub} y$ , so that  $w \in \mathbf{HasSub} x \cap \mathbf{HasSub} y$ .
  - Suppose  $w \in \mathbf{HasSub} x \cap \mathbf{HasSub} y$ . Thus  $w \in \mathbf{HasSub} x$  and  $w \in \mathbf{HasSub} y$ , so that  $w \in \{0, 1\}^*$ ,  $x$  is a substring of  $w$ , and  $y$  is a substring of  $w$ . Thus  $w \in \mathbf{HasSubs}(x, y)$ .
- (4) Suppose  $x, y \in \{0, 1\}^*$ . We must show that  $\mathbf{HasNotSubs}(x, y) = \mathbf{HasNotSub} x \cap \mathbf{HasNotSub} y$ . It will suffice to show that  $\mathbf{HasNotSubs}(x, y) \subseteq \mathbf{HasNotSub} x \cap \mathbf{HasNotSub} y \subseteq \mathbf{HasNotSubs}(x, y)$ .
- Suppose  $w \in \mathbf{HasNotSubs}(x, y)$ . Then  $w \in \{0, 1\}^*$ ,  $x$  is not a substring of  $w$ , and  $y$  is not a substring of  $w$ . Thus  $w \in \mathbf{HasNotSub} x$  and  $w \in \mathbf{HasNotSub} y$ , so that  $w \in \mathbf{HasNotSub} x \cap \mathbf{HasNotSub} y$ .
  - Suppose  $w \in \mathbf{HasNotSub} x \cap \mathbf{HasNotSub} y$ . Thus  $w \in \mathbf{HasNotSub} x$  and  $w \in \mathbf{HasNotSub} y$ , so that  $w \in \{0, 1\}^*$ ,  $x$  is not a substring of  $w$ , and  $y$  is not a substring of  $w$ . Thus  $w \in \mathbf{HasNotSubs}(x, y)$ .
- (5) Suppose  $x, y \in \{0, 1\}^*$ . We must show that  $\mathbf{IFF}(x, y) = \mathbf{HasSubs}(x, y) \cup \mathbf{HasNotSubs}(x, y)$ . It will suffice to show that  $\mathbf{IFF}(x, y) \subseteq \mathbf{HasSubs}(x, y) \cup \mathbf{HasNotSubs}(x, y) \subseteq \mathbf{IFF}(x, y)$ .

- Suppose  $w \in \mathbf{IFF}(x, y)$ . Then  $w \in \{0, 1\}^*$  and  $x$  is a substring of  $w$  iff  $y$  is a substring of  $w$ . There are two cases to consider.
  - Suppose  $x$  is a substring of  $w$ . Then  $y$  is a substring of  $w$ . Hence  $w \in \mathbf{HasSubs}(x, y) \subseteq \mathbf{HasSubs}(x, y) \cup \mathbf{HasNotSubs}(x, y)$ .
  - Suppose  $x$  is not a substring of  $w$ . Thus  $y$  is not a substring of  $w$ . Hence  $w \in \mathbf{HasNotSubs}(x, y) \subseteq \mathbf{HasSubs}(x, y) \cup \mathbf{HasNotSubs}(x, y)$ .
- Suppose  $w \in \mathbf{HasSubs}(x, y) \cup \mathbf{HasNotSubs}(x, y)$ . There are two cases to consider.
  - Suppose  $w \in \mathbf{HasSubs}(x, y)$ . Then  $w \in \{0, 1\}^*$ ,  $x$  is a substring of  $w$ , and  $y$  is a substring of  $w$ . Thus  $x$  is a substring of  $w$  iff  $y$  is a substring of  $w$ , so that  $w \in \mathbf{IFF}(x, y)$ .
  - Suppose  $w \in \mathbf{HasNotSubs}(x, y)$ . Then  $w \in \{0, 1\}^*$ ,  $x$  is not a substring of  $w$ , and  $y$  is not a substring of  $w$ . Thus  $x$  is a substring of  $w$  iff  $y$  is a substring of  $w$ , so that  $w \in \mathbf{IFF}(x, y)$ .

□

Next, we define some useful functions. Define  $\mathbf{efaToDFA} \in \mathbf{EFA} \rightarrow \mathbf{DFA}$  by:  $\mathbf{efaToDFA} = \mathbf{nfaToDFA} \circ \mathbf{efaToNFA}$ . Then we have that, for all  $M \in \mathbf{EFA}$ ,

$$L(\mathbf{efaToDFA} M) = L(\mathbf{nfaToDFA}(\mathbf{efaToNFA} M)) = L(\mathbf{efaToNFA} M) = L(M).$$

Define  $\mathbf{regToEFA} \in \mathbf{Reg} \rightarrow \mathbf{EFA}$  by:  $\mathbf{regToEFA} = \mathbf{faToEFA} \circ \mathbf{regToFA}$ . Then we have that, for all  $\alpha \in \mathbf{Reg}$ ,

$$L(\mathbf{regToEFA} \alpha) = L(\mathbf{faToEFA}(\mathbf{regToFA} \alpha)) = L(\mathbf{regToFA} \alpha) = L(\alpha).$$

Define  $\mathbf{regToDFA} \in \mathbf{Reg} \rightarrow \mathbf{DFA}$  by:  $\mathbf{regToDFA} = \mathbf{efaToDFA} \circ \mathbf{regToEFA}$ . Then we have that, for all  $\alpha \in \mathbf{Reg}$ ,

$$L(\mathbf{regToDFA} \alpha) = L(\mathbf{efaToDFA}(\mathbf{regToEFA} \alpha)) = L(\mathbf{regToEFA} \alpha) = L(\alpha).$$

Define  $\mathbf{faToDFA} \in \mathbf{FA} \rightarrow \mathbf{DFA}$  by:  $\mathbf{faToDFA} = \mathbf{efaToDFA} \circ \mathbf{faToEFA}$ . Then we have that, for all  $M \in \mathbf{FA}$ ,

$$L(\mathbf{faToDFA} M) = L(\mathbf{efaToDFA}(\mathbf{faToEFA} M)) = L(\mathbf{faToEFA} M) = L(M).$$

Define  $\mathbf{minAndRen} \in \mathbf{DFA} \rightarrow \mathbf{DFA}$  by: for all  $M \in \mathbf{DFA}$ ,

$$\mathbf{minAndRen} M = \mathbf{renameStatesCanonically}(\mathbf{minimize} M).$$

Then, for all  $M \in \mathbf{DFA}$ ,

$$\begin{aligned} L(\mathbf{minAndRen} M) &= L(\mathbf{renameStatesCanonically}(\mathbf{minimize} M)) \\ &= L(\mathbf{minimize} M) = L(M), \end{aligned}$$

and  $\mathbf{minAndRen} M$  has no more states than any DFA that is equivalent to  $M$ .

Define the DFA **allStrDFA** by:

$$\mathbf{allStrDFA} = \mathbf{minAndRen}(\mathbf{regToDFA}((0 + 1)^*)).$$

Then, we have that

$$\begin{aligned} L(\mathbf{allStrDFA}) &= L(\mathbf{minAndRen}(\mathbf{regToDFA}((0 + 1)^*))) \\ &= L(\mathbf{regToDFA}((0 + 1)^*)) \\ &= L((0 + 1)^*) = \{0, 1\}^*. \end{aligned}$$

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

Define **hasSubFA**  $\in \{0, 1\}^* \rightarrow \mathbf{FA}$  by: for all  $x \in \{0, 1\}^*$ ,

$$\mathbf{hasSubFA} x = \mathbf{concat}(\mathbf{allStrFA}, \mathbf{concat}(\mathbf{strToFA} x, \mathbf{allStrFA})).$$

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

$$\begin{aligned} L(\mathbf{hasSubFA} x) &= L(\mathbf{concat}(\mathbf{allStrFA}, \mathbf{concat}(\mathbf{strToFA} x, \mathbf{allStrFA}))) \\ &= L(\mathbf{allStrFA}) L(\mathbf{strToFA} x) L(\mathbf{allStrFA}) \\ &= \{0, 1\}^* \{x\} \{0, 1\}^* \\ &= \mathbf{HasSub} x, \end{aligned}$$

by Lemma PS5.3.1(1).

Define **hasSubDFA**  $\in \{0, 1\}^* \rightarrow \mathbf{DFA}$  by:

$$\mathbf{hasSubDFA} = \mathbf{minAndRen} \circ \mathbf{faToDFA} \circ \mathbf{hasSubFA}.$$

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

$$\begin{aligned} L(\mathbf{hasSubDFA} x) &= L(\mathbf{minAndRen}(\mathbf{faToDFA}(\mathbf{hasSubFA} x))) \\ &= L(\mathbf{faToDFA}(\mathbf{hasSubFA} x)) \\ &= L(\mathbf{hasSubFA} x) \\ &= \mathbf{HasSub} x. \end{aligned}$$

Define **hasSubEFA**  $\in \{0, 1\}^* \rightarrow \mathbf{EFA}$  by: **hasSubEFA** = **hasSubDFA**. Then, for all  $x \in \{0, 1\}^*$ ,

$$L(\mathbf{hasSubEFA} x) = L(\mathbf{hasSubDFA} x) = \mathbf{HasSub} x.$$

Define **hasNotSubDFA**  $\in \{0, 1\}^* \rightarrow \mathbf{DFA}$  by: for all  $x \in \{0, 1\}^*$ ,

$$\mathbf{hasNotSubDFA} x = \mathbf{minus}(\mathbf{allStrDFA}, \mathbf{hasSubDFA} x).$$

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

$$\begin{aligned} L(\mathbf{hasNotSubDFA} x) &= L(\mathbf{minus}(\mathbf{allStrDFA}, \mathbf{hasSubDFA} x)) \\ &= L(\mathbf{allStrDFA}) - L(\mathbf{hasSubDFA} x) \\ &= \{0, 1\}^* - \mathbf{HasSub} x \\ &= \mathbf{HasNotSub} x, \end{aligned}$$

by Lemma PS5.3.1(2).

Define  $\mathbf{hasNotSubEFA} \in \{0, 1\}^* \rightarrow \mathbf{EFA}$  by:  $\mathbf{hasNotSubEFA} = \mathbf{hasNotSubDFA}$ . Then, for all  $x \in \{0, 1\}^*$ ,

$$L(\mathbf{hasNotSubEFA} x) = L(\mathbf{hasNotSubDFA} x) = \mathbf{HasNotSub} x.$$

Define  $\mathbf{hasSubsEFA} \in \{0, 1\}^* \times \{0, 1\}^* \rightarrow \mathbf{EFA}$  by: for all  $x, y \in \{0, 1\}^*$ ,

$$\mathbf{hasSubsEFA}(x, y) = \mathbf{inter}(\mathbf{hasSubEFA} x, \mathbf{hasSubEFA} y).$$

Then, for all  $x, y \in \{0, 1\}^*$ ,

$$\begin{aligned} L(\mathbf{hasSubsEFA}(x, y)) &= L(\mathbf{inter}(\mathbf{hasSubEFA} x, \mathbf{hasSubEFA} y)) \\ &= L(\mathbf{hasSubEFA} x) \cap L(\mathbf{hasSubEFA} y) \\ &= \mathbf{HasSub} x \cap \mathbf{HasSub} y \\ &= \mathbf{HasSubs}(x, y), \end{aligned}$$

by Lemma PS5.3.1(3).

Define  $\mathbf{hasNotSubsEFA} \in \{0, 1\}^* \times \{0, 1\}^* \rightarrow \mathbf{EFA}$  by: for all  $x, y \in \{0, 1\}^*$ ,

$$\mathbf{hasNotSubsEFA}(x, y) = \mathbf{inter}(\mathbf{hasNotSubEFA} x, \mathbf{hasNotSubEFA} y).$$

Then, for all  $x, y \in \{0, 1\}^*$ ,

$$\begin{aligned} L(\mathbf{hasNotSubsEFA}(x, y)) &= L(\mathbf{inter}(\mathbf{hasNotSubEFA} x, \mathbf{hasNotSubEFA} y)) \\ &= L(\mathbf{hasNotSubEFA} x) \cap L(\mathbf{hasNotSubEFA} y) \\ &= \mathbf{HasNotSub} x \cap \mathbf{HasNotSub} y \\ &= \mathbf{HasNotSubs}(x, y), \end{aligned}$$

by Lemma PS5.3.1(4).

Define  $\mathbf{iffEFA} \in \{0, 1\}^* \times \{0, 1\}^* \rightarrow \mathbf{EFA}$  by: for all  $x, y \in \{0, 1\}^*$ ,

$$\mathbf{iffEFA}(x, y) = \mathbf{union}(\mathbf{hasSubsEFA}(x, y), \mathbf{hasNotSubsEFA}(x, y)).$$

Then, for all  $x \in \{0, 1\}^*$ ,

$$\begin{aligned} L(\mathbf{iffEFA}(x, y)) &= L(\mathbf{union}(\mathbf{hasSubsEFA}(x, y), \mathbf{hasNotSubsEFA}(x, y))) \\ &= L(\mathbf{hasSubsEFA}(x, y)) \cup L(\mathbf{hasNotSubsEFA}(x, y)) \\ &= \mathbf{HasSubs}(x, y) \cup \mathbf{HasNotSubs}(x, y) \\ &= \mathbf{IFF}(x, y), \end{aligned}$$

by Lemma PS5.3.1(5).

Finally, define  $\mathbf{iffDFA} \in \{0, 1\}^* \times \{0, 1\}^* \rightarrow \mathbf{DFA}$  by:

$$\mathbf{iffDFA} = \mathbf{minAndRen} \circ \mathbf{efaToDFA} \circ \mathbf{iffEFA}.$$

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

$$\begin{aligned}
 L(\mathbf{iffDFA}(x, y)) &= L(\mathbf{minAndRen}(\mathbf{efaToDFA}(\mathbf{iffEFA}(x, y)))) \\
 &= L(\mathbf{efaToDFA}(\mathbf{iffEFA}(x, y))) \\
 &= L(\mathbf{iffEFA}(x, y)) \\
 &= \mathbf{IFF}(x, y),
 \end{aligned}$$

and  $\mathbf{iffDFA}(x, y)$  has as few states as possible, since its last step is  $\mathbf{minAndRen}$ .

(b) First, we put the text

```

val efaToDFA = nfaToDFA o efaToNFA;
val regToEFA = faToEFA o regToFA;
val regToDFA = efaToDFA o regToEFA;
val faToDFA  = efaToDFA o faToEFA;
val minAndRen = DFA.renameStatesCanonically o DFA.minimize;

val allStrDFA = minAndRen(regToDFA(Reg.fromString "(0 + 1)*"));
val allStrFA  = injDFAToFA allStrDFA;

fun hasSubFA x = FA.concat(allStrFA, FA.concat(strToFA x, allStrFA));

val hasSubDFA = minAndRen o faToDFA o hasSubFA;
val hasSubEFA = injDFAToEFA o hasSubDFA;

fun hasNotSubDFA x = DFA.minus(allStrDFA, hasSubDFA x);

val hasNotSubEFA = injDFAToEFA o hasNotSubDFA;

fun hasSubsEFA(x, y) = EFA.inter(hasSubEFA x, hasSubEFA y);

fun hasNotSubsEFA(x, y) = EFA.inter(hasNotSubEFA x, hasNotSubEFA y);

fun iffEFA(x, y) = EFA.union(hasSubsEFA(x, y), hasNotSubsEFA(x, y));

val iffDFA = minAndRen o efaToDFA o iffEFA;

```

in the file `ps5-p3.sml`. Then we invoke Forlan and proceed as follows:

```

- use "ps5-p3.sml";
[opening ps5-p3.sml]
val efaToDFA = fn : efa -> dfa
val regToEFA = fn : reg -> efa
val regToDFA = fn : reg -> dfa
val faToDFA  = fn : fa  -> dfa
val minAndRen = fn : dfa -> dfa
val allStrDFA = - : dfa
val allStrFA  = - : fa

```



```

val hasSubFA = fn : str -> fa
val hasSubDFA = fn : str -> dfa
val hasSubEFA = fn : str -> efa
val hasNotSubDFA = fn : str -> dfa
val hasNotSubEFA = fn : str -> efa
val hasSubsEFA = fn : str * str -> efa
val hasNotSubsEFA = fn : str * str -> efa
val iffEFA = fn : str * str -> efa
val iffDFA = fn : str * str -> dfa
val it = () : unit
- val dfa = iffDFA(Str.fromString "0011", Str.fromString "1100");
val dfa = - : dfa
- DFA.output("", dfa);
{states} A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P {start state} A
{accepting states} A, B, C, E, F, H, J, P
{transitions}
A, 0 -> E; A, 1 -> B; B, 0 -> E; B, 1 -> C; C, 0 -> F; C, 1 -> C; D, 0 -> G;
D, 1 -> D; E, 0 -> J; E, 1 -> B; F, 0 -> K; F, 1 -> B; G, 0 -> K; G, 1 -> D;
H, 0 -> E; H, 1 -> O; I, 0 -> G; I, 1 -> P; J, 0 -> J; J, 1 -> H; K, 0 -> K;
K, 1 -> I; L, 0 -> L; L, 1 -> M; M, 0 -> L; M, 1 -> O; N, 0 -> P; N, 1 -> M;
O, 0 -> N; O, 1 -> O; P, 0 -> P; P, 1 -> P
val it = () : unit

```

Here is a drawing of dfa:

