Exercise Set 6

Model Answers

Exercise 1

(a)

\[
\begin{align*}
A & \rightarrow 0A3 \mid B \mid C \\
B & \rightarrow 1B3 \mid D \\
C & \rightarrow 0C2 \mid D \\
D & \rightarrow 1D2 \mid \%
\end{align*}
\]

(b) Let \(pt_1\) and \(pt_2\) be the following parse trees:

\[
\begin{align*}
\text{(pt}_1\text{)} & \\
A & \\
0 & A & 3 \\
C & \\
0 & C & 2 \\
D & \\
1 & D & 2 \\
\% & \\
\text{(pt}_2\text{)} & \\
A & \\
0 & A & 3 \\
B & \\
1 & B & 3 \\
D & \\
1 & D & 2 \\
\%
\end{align*}
\]

Then \(pt_1\) is a parse of 001223, and \(pt_2\) is a parse of 00112333.

To check that our answers are correct, we place the description

\[
\begin{align*}
\{\text{variables}\} & \\
A, B, C, D & \\
\{\text{start variable}\} & \\
A & \\
\{\text{productions}\} & \\
A & \rightarrow 0A3 \mid B \mid C; \\
B & \rightarrow 1B3 \mid D; \\
C & \rightarrow 0C2 \mid D; \\
D & \rightarrow 1D2 \mid \%
\end{align*}
\]
of $G$ in the file es6-ex1-gram, and then load $G$ into Forlan:

```ml
- val gram = Gram.input "es6-ex1-gram";
val gram = - : gram
```

Next, we put the description

```ml
A(0,
 A(C(0,
  C(D(1,
   D(%),
    2)),
     2)),
  3),
3)
```

of $pt_1$ in the file es6-ex1-pt1, load $pt_1$ into Forlan, and check that it has the required properties:

```ml
- val pt1 = PT.input "es6-ex1-pt1";
val pt1 = - : pt
- Gram.validPT gram pt1;
val it = true : bool
- Sym.output("", PT.rootLabel pt1);
  A
val it = () : unit
- Str.output("", PT.yield pt1);
  001223
val it = () : unit
```

Finally, we put the description

```ml
A(0,
 A(0,
  A(B(1,
   B(D(1,
    D(%),
     2)),
      2)),
   3),
3),
3)
```

of $pt_2$ in the file es6-ex1-pt2, load $pt_2$ into Forlan, and check that it has the required properties:

```ml
- val pt2 = PT.input "es6-ex1-pt2";
val pt2 = - : pt
- Gram.validPT gram pt2;
val it = true : bool
- Sym.output("", PT.rootLabel pt2);
  A
val it = () : unit
```
(c) First, we define a testing function, and check that some strings in $X$ are generated by $G$:

```ml
val generated = fn : string -> bool

- map generated
  = ["%", "12", "03", "02", "13", "1122", "0033", "0123", "001233", "00012233", "011333", "0111122333"];

val it = [true,true,true,true,true,true,true,true,true,true,true,true] : bool list
```

Second, we check that some strings that are not in $X$ are not generated by $G$:

```ml
val generated = fn : string -> bool

- map generated

val it = [false,false,false,false,false,false,false,false,false,false,false,false,false,false,false] : bool list
```

(d) Let

$$
Y = \{1^j 2^k 3^l \mid j, k, l \in \mathbb{N} \text{ and } j = k + l\},
$$

$$
Z = \{0^i 1^j 2^k \mid i, j, k \in \mathbb{N} \text{ and } i + j = k\},
$$

$$
W = \{1^n 2^n \mid n \in \mathbb{N}\}.
$$

We will show that $\Pi_A = X$, $\Pi_B = Y$, $\Pi_C = Z$, $\Pi_D = W$.

**Lemma ES6.1.1**

$W \subseteq \Pi_D$.

**Proof.** It will suffice to show that, for all $n \in \mathbb{N}$, $1^n 2^n \in \Pi_D$. We proceed by mathematical induction.

(Basis Step) Because $D \rightarrow \% \in P$, we have that $1^0 2^0 = \%\% = \% \in \Pi_D$.

(Inductive Step) Suppose $n \in \mathbb{N}$, and assume the inductive hypothesis: $1^n 2^n \in \Pi_D$. Because $D \rightarrow 1D2 \in P$, it follows that $1^{n+1} 2^{n+1} = 1(1^n 2^n)2 \in \Pi_D$. $\square$

**Lemma ES6.1.2**

$Y \subseteq \Pi_B$.

**Proof.** First, we show a fact that we call (†) below: for all $x \in W$ and $n \in \mathbb{N}$, $1^n x 3^n \in \Pi_B$. Let $x \in W$. We use mathematical induction to show that, for all $n \in \mathbb{N}$, $1^n x 3^n \in \Pi_B$.

(Basis Step) By Lemma ES6.1.1, we have that $x \in W \subseteq \Pi_D$. Thus, since $B \rightarrow D \in P$, it follows that $1^0 x 3^0 = \%x\% = x \in \Pi_B$. 

3
Lemma ES6.1.3

\[ Z \subseteq \Pi_C. \]

**Proof.** First, we show a fact that we call \((\dagger)\) below: for all \(x \in W\) and \(n \in \mathbb{N}\), \(0^n x 2^n \in \Pi_C\). Let \(x \in W\). We use mathematical induction to show that, for all \(n \in \mathbb{N}\), \(0^n x 3^n \in \Pi_C\).

1. **Basis Step** By Lemma ES6.1.1, we have that \(x \in W \subseteq \Pi_D\). Thus, since \(C \rightarrow D \in P\), it follows that \(0^0 x 2^0 = %x\% = x \in \Pi_C\).

2. **Inductive Step** Suppose \(n \in \mathbb{N}\), and assume the inductive hypothesis: \(0^n x 2^n \in \Pi_C\). Thus, since \(C \rightarrow 0C2 \in P\), it follows that \(0^{n+1} x 2^{n+1} = 0(0^n x 2^n)2 \in \Pi_C\).

Now, we use \((\dagger)\) to show that \(Z \subseteq \Pi_C\). Suppose \(w \in Z\), so that \(w = 0^i 1^j 2^k\), for some \(i, j, k \in \mathbb{N}\) such that \(i + j = k\). Let \(x = 1^j 2^k\). Since \(x \in W\), we have that \(w = 0^i 1^j 2^k = 0^i 1^j 2^{i+j} = 0^i 1^j 2^{i+j} = 0^i 1^j 2^{i+j} = 0^i x 2^i \in \Pi_C\), by \((\dagger)\). \(\square\)

**Lemma ES6.1.4**

\(X \subseteq \Pi_A\).

**Proof.** First, we show a fact that we call \((\ddagger)\) below: for all \(x \in Y \cup Z\) and \(n \in \mathbb{N}\), \(0^n x 3^n \in \Pi_A\). Let \(x \in Y \cup Z\). We use mathematical induction to show that, for all \(n \in \mathbb{N}\), \(0^n x 3^n \in \Pi_A\).

1. **Basis Step** Since \(x \in Y \cup Z\), there are two cases to consider.
   - Suppose \(x \in Y\). By Lemma ES6.1.2, we have that \(x \in Y \subseteq \Pi_B\). Thus, since \(A \rightarrow B \in P\), it follows that \(0^0 x 3^0 = %x\% = x \in \Pi_A\).
   - Suppose \(x \in Z\). By Lemma ES6.1.3, we have that \(x \in Z \subseteq \Pi_C\). Thus, since \(A \rightarrow C \in P\), it follows that \(0^0 x 3^0 = %x\% = x \in \Pi_A\).

2. **Inductive Step** Suppose \(n \in \mathbb{N}\), and assume the inductive hypothesis: \(0^n x 3^n \in \Pi_A\). Thus, since \(A \rightarrow 0A3 \in P\), it follows that \(0^{n+1} x 3^{n+1} = 0(0^n x 3^n)3 \in \Pi_A\).

Now, we use \((\ddagger)\) to show that \(X \subseteq \Pi_A\). Suppose \(w \in X\), so that \(w = 0^i 1^j 2^k\), for some \(i, j, k \in \mathbb{N}\) such that \(i + j = k + l\). There are two cases to consider.
   - Suppose \(i \leq l\). Thus \(l = i + m\), for some \(m \in \mathbb{N}\). Let \(x = 1^j 2^k 3^m\). Since \(i + j = k + l = k + i + m = i + k + m\), we have that \(j = k + m\), and thus that \(x \in Y \subseteq Y \cup Z\). Hence, by \((\ddagger)\), we have that \(w = 0^i 1^j 2^k 3^m = 0^i 1^j 2^k 3^m 3^i = 0^i 1^j 2^k 3^m 3^i = 0^i x 3^i \in \Pi_A\).
   - Suppose \(i > l\). Thus \(i = l + m\), for some \(m \in \mathbb{N} \setminus \{0\}\). Let \(x = 0^m 1^j 2^k\). Since \(l + m + j = i + j = k + l\), we have that \(m + j = k\), and thus that \(x \in Z \subseteq Y \cup Z\). Hence, by \((\ddagger)\), we have that \(w = 0^m 1^j 2^k 3^l = 0^m 1^j 2^k 3^l = 0^m 0^m 1^j 2^k 3^l = 0^i x 3^i \in \Pi_A\).

\(\square\)

**Lemma ES6.1.5**

(A) \(\Pi_A \subseteq X\).
We proceed by induction on $\Pi$. There are nine productions to consider.

**Proof.** It will suffice to show that:

(A) For all $w \in \Pi_A$, $w \in X$.
(B) For all $w \in \Pi_B$, $w \in Y$.
(C) For all $w \in \Pi_C$, $w \in Z$.
(D) For all $w \in \Pi_D$, $w \in W$.

We proceed by induction on $\Pi$. There are nine productions to consider.

- **(A $\rightarrow$ 0A3)** Suppose $w \in \Pi_A$, and assume the inductive hypothesis: $w \in X$. Then $w = 0^i1^j2^k3^l$, for some $i, j, k, l \in \mathbb{N}$ such that $i+j = k+l$. Hence $0w3 = 0^i1^j2^k3^l3 = 0^{i+1}1^j2^k3^{l+1} \in X$, since $(i+1)+j = (i+j) + 1 = (k+l) + 1 = k + (l+1)$.

- **(A $\rightarrow$ B)** Suppose $w \in \Pi_B$, and assume the inductive hypothesis: $w \in Y$. Then $w = 1^i2^j3^l$, for some $j, k, l \in \mathbb{N}$ such that $j = k + l$. Hence $w = W = 1^i2^j3^l = 0^j1^j2^k3^l \in X$, since $0 + j = j = k + l$.

- **(A $\rightarrow$ C)** Suppose $w \in \Pi_C$, and assume the inductive hypothesis: $w \in Z$. Then $w = 0^i1^j2^k$, for some $i, j, k \in \mathbb{N}$ such that $i+j = k$. Hence $w \in Z = 0^i1^j2^k3^l \in X$, since $i + j = k + 0$.

- **(B $\rightarrow$ 1B3)** Suppose $w \in \Pi_B$, and assume the inductive hypothesis: $w \in Y$. Then $w = 1^i2^k3^l$, for some $j, k, l \in \mathbb{N}$ such that $j = k + l$. Hence $1w3 = 1^i2^k3^l3 = 1^{i+1}2^k3^{l+1} \in Y$, since $j + 1 = (k+l) + 1 = k + (l+1)$.

- **(B $\rightarrow$ D)** Suppose $w \in \Pi_D$, and assume the inductive hypothesis: $w \in W$. Then $w = 1^n2^n$, for some $n \in \mathbb{N}$. Hence $w = w \in Z = 0^i1^j2^k3^l \in Y$, since $n = n + 0$.

- **(C $\rightarrow$ 0C2)** Suppose $w \in \Pi_C$, and assume the inductive hypothesis: $w \in Z$. Then $w = 0^i1^j2^k$, for some $i, j, k \in \mathbb{N}$ such that $i+j = k$. Hence $0w2 = 0^i1^j2^k2 = 0^{i+1}1^j2^k \in Z$, since $(i+1) + j = (i+j) + 1 = k + 1$.

- **(C $\rightarrow$ D)** Suppose $w \in \Pi_D$, and assume the inductive hypothesis: $w \in W$. Then $w = 1^n2^n$, for some $n \in \mathbb{N}$. Hence $w = w \in Z = 0^i1^j2^k3^l \in Y$, since $0 + n = n$.

- **(D $\rightarrow$ 1D2)** Suppose $w \in \Pi_D$, and assume the inductive hypothesis: $w \in W$. Then $w = 1^n2^n$, for some $n \in \mathbb{N}$. Hence $1w2 = 1^i2^i2^2 = 1^{i+1}2^n \in W$.

- **(D $\rightarrow$ %)** We have that $% = % = 1^n2^n \in W$.
By Lemmas ES6.1.4 and ES6.1.5(A), we have that $X \subseteq \Pi_A \subseteq X$, so that $L(G) = \Pi_A = X$. 

(e) Suppose, toward a contradiction, that $X$ is regular. Thus there is an $n \in \mathbb{N}$ with the property of the Pumping Lemma, where $X$ has been substituted for $L$. Let $z = 0^n3^n$. Since $z = 0^n3^n = 0^n1^03^n$ and $n + 0 = 0 + n$, we have that $z \in X$. And, $|z| = 2n \geq n$. Thus, by the property of the Pumping Lemma, we have that there are $u, v, w \in \text{Str}$ such that $z = uvw$ and

1. $|uv| \leq n$;
2. $v \neq \varepsilon$; and
3. $uv^iw \in X$, for all $i \in \mathbb{N}$.

Since $0^n3^n = z = uvw$, (1) tells us that there are $i, j, k \in \mathbb{N}$ such that

$$u = 0^i, \quad v = 0^j, \quad w = 0^k3^n, \quad i + j + k = n.$$ 

By (2), we have that $j \geq 1$, and thus that $i + k = n - j < n$. By (3), we have that

$$0^{i+k}3^n = 0^i0^k3^n = uw = u\varepsilon w = uw0w \in X.$$ 

Hence $0^{i+k}3^n = 0^{i'}1^j2^k3^{l'}$ for some $i', j', k', l' \in \mathbb{N}$ such that $i' + j' = k' + l'$, so that $i + k = i'$, $0 = j'$, $0 = k'$ and $n = l'$. Thus $i + k = i + k + 0 = i' + j' = k' + l' = 0 + n = n$—contradiction. Thus $X$ is not regular.

Exercise 2

(a)

(* val zero : sym *)

val zero = Sym.fromString "0";

(* val one : sym *)

val one = Sym.fromString "1";

(* val minAndRen : dfa -> dfa *)

val minAndRen = DFA.renameStatesCanonically o DFA.minimize o DFA.renameStatesCanonically;

(* val efatoDFA : efa -> dfa *)

val efatoDFA = nfaToDFA o efatoNFA;

(* val allStrDFA : dfa *)

allStrDFA accepts {0, 1}* * *)
val allStrDFA = 
  minAndRen(efaToDFA(EFA.closure(EFA.union(EFA.fromSym zero, 
      EFA.fromSym one))));

(* val swapRel : sym_rel *)

val swapRel = SymRel.fromString "((0, 1), (1, 0))";

(* val zeroAllPrefPosEFA : int -> efa
  if n >= 0, then zeroAllPrefPosEFA n returns an efa that accepts all
  w in \{0, 1\}^* such that diff w = 0 and all prefixes of w have diff's
  between 0 and n *)

fun zeroAllPrefPosEFA 0 = EFA.emptyStr 
  | zeroAllPrefPosEFA n = 
      EFA.closure(EFA.concat(EFA.fromSym one, 
        EFA.concat(zeroAllPrefPosEFA(n - 1), 
          EFA.fromSym zero)));

(* val zeroAllPrefPosDFA : int -> dfa
  if n >= 0, then zeroAllPrefPosDFA n returns a dfa that accepts
  all w in \{0, 1\}^* such that diff w = 0 and all prefixes of w
  have diff's between 0 and n *)

fun zeroAllPrefPosDFA n = minAndRen(efaToDFA(zeroAllPrefPosEFA n));

(* val justBadPosEFA : int -> efa
  if n >= 0, then justBadPosEFA n returns an efa that accepts all w
  in \{0, 1\}^* such that diff w = n + 1 and all proper prefixes of w
  have diff's between 0 and n *)

fun justBadPosEFA 0 = EFA.fromSym one 
  | justBadPosEFA n = 
      EFA.concat(injDFAToEFA(zeroAllPrefPosDFA n), 
        EFA.concat(EFA.fromSym one, justBadPosEFA(n - 1)));

(* val justBadPosDFA : int -> dfa
  if n >= 0, then justBadPosDFA n returns a dfa that accepts all w in
  \{0, 1\}^* such that diff w = n + 1 and all proper prefixes of w have
  diff's between 0 and n *)

fun justBadPosDFA n = minAndRen(efaToDFA(justBadPosEFA n));
(* val justBadNegDFA : int -> dfa

if n >= 0, then justBadNegDFA n returns a dfa that accepts all w in
\{0, 1\}* such that \text{diff } w = \neg(n + 1) and all proper prefixes of w
have diff’s between 0 and \neg n *)

fun justBadNegDFA n = DFA.renameAlphabet(justBadPosDFA n, swapRel);

(* val justBadPosOrNegDFA : int -> dfa

if n >= 0, then justBadPosOrNegDFA n returns a dfa accepting all w
in \{0, 1\}* such that either:

- \text{diff } w = n + 1 and all proper prefixes of w have diff’s between 0
  and n; or

- \text{diff } w = \neg(n + 1) and all proper prefixes of w have diff’s
  between 0 and \neg n *)

fun justBadPosOrNegDFA n =
  minAndRen(efaToDFA(EFA.union(injDFAToEFA(justBadPosDFA n),
  injDFAToEFA(justBadNegDFA n))));

(* val someSubBadEFA : int -> efa

if n >= 0, then someSubBadEFA n returns an efa that accepts all w in
\{0, 1\}* such that some substring of w has a diff that is < \neg n or > n *)

fun someSubBadEFA n =
  EFA.concat(injDFAToEFA allStrDFA,
  EFA.concat(injDFAToEFA(justBadPosOrNegDFA n),
  injDFAToEFA allStrDFA));

(* val someSubBadDFA : int -> dfa

if n >= 0, then someSubBadDFA n returns a dfa that accepts all w in
\{0, 1\}* such that some substring of w has a diff that is < \neg n or > n *)

fun someSubBadDFA n =
  minAndRen(efaToDFA(someSubBadEFA n));

(* val allSubGoodDFA : int -> dfa

if n >= 0, then allSubGoodDFA n returns a minimized dfa that
accepts all w in \{0, 1\}* such that all substrings of w have diff’s
between ‘n and n *)

fun allSubGoodDFA n =
  minAndRen(DFA.minus(allStrDFA, someSubBadDFA n));
(b) First, we put the solution to Part (a) in the file `all-sub-good.sml`. Then we invoke Forlan and load this file:

```ml
val zero = - : sym
val one = - : sym
val minAndRen = fn : dfa -> dfa
val efaToDFA = fn : efa -> dfa
val allStrDFA = - : dfa
val swapRel = - : sym_rel
val zeroAllPrefPosEFA = fn : int -> efa
val zeroAllPrefPosDFA = fn : int -> dfa
val justBadPosEFA = fn : int -> efa
val justBadPosDFA = fn : int -> dfa
val justBadNegDFA = fn : int -> dfa
val justBadPosOrNegDFA = fn : int -> dfa
val someSubBadEFA = fn : int -> efa
val someSubBadDFA = fn : int -> dfa
val allSubGoodDFA = fn : int -> dfa
val it = () : unit
```

Next, we generate and display a DFA accepting `AllSubGood(2)`:

```ml
val dfa1 = allSubGoodDFA 2;
val dfa1 = - : dfa
val it = () : unit
```

Here is a drawing of `dfa1`:
Finally, we generate and display a DFA accepting \texttt{AllSubGood}(3):

```ml
val dfa2 = allSubGoodDFA 3;
val dfa2 = - : dfa
- DFA.output("", dfa2);
{states}
A, B, C, D, E, F, G, H, I, J, K
{start state}
A
{accepting states}
A, B, C, D, E, F, G, H, I, J
{transitions}
A, 0 \rightarrow B; A, 1 \rightarrow C; B, 0 \rightarrow F; B, 1 \rightarrow C; C, 0 \rightarrow B; C, 1 \rightarrow G; D, 0 \rightarrow E;
D, 1 \rightarrow J; E, 0 \rightarrow I; E, 1 \rightarrow D; F, 0 \rightarrow I; F, 1 \rightarrow H; G, 0 \rightarrow H; G, 1 \rightarrow J;
H, 0 \rightarrow F; H, 1 \rightarrow G; I, 0 \rightarrow K; I, 1 \rightarrow E; J, 0 \rightarrow D; J, 1 \rightarrow K; K, 0 \rightarrow K;
K, 1 \rightarrow K
val it = () : unit
```

Here is a drawing of \texttt{dfa2}: