CIS 570 — Introduction to Formal Language Theory — Fall 2007

Exercise Set 2

Model Answers

Exercise 1

First, we put the following text in the file `es2-ex1.sml`:

```
(* written so that the string length upper bound is a parameter; this
   generality was not required *)

(* val zeroOne : str set

   zeroOne = {0, 1} *)

val zeroOne = StrSet.fromString "0, 1";

(* val unwanted : int -> str set

   if n >= 0, then unwanted n returns all length-n elements of {0, 1}*
   with an occurrence of either 000/111 *)

fun unwanted n =
      if n < 3
      then Set.empty
      else let (* val bad : str set

                  bad = {000, 111} *)

               val bad = StrSet.fromString "000, 111"

               (* val unwant : int -> str set

                  if 1 <= m <= n - 2, then unwant m is all length n
                  elements of {0, 1}* with an occurrence of 000/111
                  beginning at one of the positions 1, ..., m *)

               fun unwant 1 = StrSet.concat(bad, StrSet.power(zeroOne, n - 3))
                 | unwant m =
                     StrSet.union
                     (StrSet.concat
                      (StrSet.power(zeroOne, m - 1),
                       StrSet.concat(bad,
                                     StrSet.power(zeroOne, n - 3 - (m - 1)))),
                      unwant(m - 1))
           in unwant(n - 2) end
```

```
(* val atMost : int -> str set

   if n >= 0, then atMost n returns all elements of {0, 1}* that have
   length at most n and have no occurrences of 000/111 *)

fun atMost 0 = StrSet.minus(StrSet.power(zeroOne, 0), unwanted 0)
  | atMost n =
      StrSet.union(StrSet.minus(StrSet.power(zeroOne, n), unwanted n),
                   atMost(n - 1));

(* val atMostReg : int -> reg

   if n >= 0, then atMostReg n returns a Reg.weakSubset-simplified
   regular expression whose meaning is the set of all elements of {0,
   1}* that have length at most n and have no occurrences of 000/111 *)

fun atMostReg n = Reg.simplify Reg.weakSubset (Reg.fromStrSet(atMost n));
```

Next, we start Forlan, and then proceed as follows:

```
- use "es2-ex1.sml";
[opening es2-ex1.sml]
val zeroOne = - : str set
val unwanted = fn : int -> str set
val atMost = fn : int -> str set
val atMostReg = fn : int -> reg
val it = () : unit
- val reg = atMostReg 4;
val reg = - : reg
- Reg.size reg;
val it = 69 : int
- Reg.output("", reg);
% + 0 + 1 + 0(0 + 1 + 0(1 + 1(0 + 1)) + 1(0 + 1 + 0(0 + 1) + 10)) +
1(0 + 1 + 0(0 + 1 + 01 + 1(0 + 1)) + 1(0 + 0(0 + 1)))
val it = () : unit
```

For fun, we also solve the version of the exercise where the upper bound on string length is 5:

```
- val reg = atMostReg 5;
val reg = - : reg
- Reg.size reg;
val it = 121 : int
- Reg.output("", reg);
% + 0 + 1 +
0
(0 + 1 + 0(1 + 1(0 + 1 + 0(0 + 1) + 10)) +
 1(0 + 1 + 0(0 + 1 + 01 + 1(0 + 1)) + 1(0 + 0(0 + 1)))) +
1
(0 + 1 + 0(0 + 1 + 0(1 + 1(0 + 1)) + 1(0 + 1 + 0(0 + 1) + 10)) +
```

```
 1(0 + 0(0 + 1 + 01 + 1(0 + 1))))
val it = () : unit
```

Exercise 2

(a) Suppose that $n \in \mathbb{N}$, $A, B \in \mathbf{Lan}$, $n \geq 1$ and $A^n \subseteq B$. To show that $A^{n+1}A^* \cup B = A^nA^* \cup B$, it will suffice to show that

$$A^{n+1}A^* \cup B \subseteq A^nA^* \cup B \subseteq A^{n+1}A^* \cup B.$$

First, we show that $A^{n+1}A^* \cup B \subseteq A^nA^* \cup B$. We have that $A^{n+1}A^* = (A^nA)A^* = A^n(AA^*) \subseteq A^nA^*$. Hence $A^{n+1}A^* \cup B \subseteq A^nA^* \cup B$.

Second, we show that $A^nA^* \cup B \subseteq A^{n+1}A^* \cup B$. Suppose $w \in A^nA^* \cup B$. We must show that $w \in A^{n+1}A^* \cup B$. There are two cases to consider.

- Suppose $w \in A^nA^*$. Hence $w = xy$ for some $x \in A^n$ and $y \in A^*$, so that $y \in A^m$ for some $m \in \mathbb{N}$. There are two subcases to consider.
  - Suppose $m = 0$. Because $y \in A^m = A^0 = \{\%\}$, we have that $y = \%$. And $A^n \subseteq B$, so that $w = xy = x\% = x \in A^n \subseteq B \subseteq A^{n+1}A^* \cup B$.
  - Suppose $m \geq 1$. Thus $w = xy \in A^nA^m = A^n(AA^{m-1}) = (A^nA)A^{m-1} = A^{n+1}A^{m-1} \subseteq A^{n+1}A^* \subseteq A^{n+1}A^* \cup B$.
- Suppose $w \in B$. Then $w \in A^{n+1}A^* \cup B$.

(b) Suppose $n \in \mathbb{N}$, $\alpha, \beta \in \mathbf{Reg}$, $n \geq 1$ and $L(\alpha^n) \subseteq L(\beta)$. Thus $L(\alpha)^n \subseteq L(\beta)$, so that

$$\begin{aligned}
L(\alpha^{n+1}\alpha^* + \beta) &= L(\alpha^{n+1}\alpha^*) \cup L(\beta) \\
&= L(\alpha^{n+1})L(\alpha^*) \cup L(\beta) \\
&= L(\alpha)^{n+1}L(\alpha)^* \cup L(\beta) \\
&= L(\alpha)^nL(\alpha)^* \cup L(\beta) \qquad \text{(by Part (a) as } n \geq 1 \text{ and } L(\alpha)^n \subseteq L(\beta)) \\
&= L(\alpha^n)L(\alpha^*) \cup L(\beta) \\
&= L(\alpha^n\alpha^*) \cup L(\beta) \\
&= L(\alpha^n\alpha^* + \beta).
\end{aligned}$$

Hence $\alpha^{n+1}\alpha^* + \beta \approx \alpha^n\alpha^* + \beta$.

Exercise 3

(a)

$$(\mathsf{1}(\mathsf{10})^*\mathsf{0})^*(\% + \mathsf{1}(\mathsf{10})^*(\% + \mathsf{1}))$$

(b) Define languages

$$A = \{\mathsf{1}\}\{\mathsf{10}\}^*\{\mathsf{0}\} \qquad \text{and} \qquad B = \{\mathsf{1}\}\{\mathsf{10}\}^*\{\%, \mathsf{1}\}.$$

Thus $L(\mathsf{1}(\mathsf{10})^*\mathsf{0}) = A$ and $L(\mathsf{1}(\mathsf{10})^*(\% + \mathsf{1})) = B$, so that $L(\alpha) = A^*(\{\%\} \cup B)$. Thus it will suffice to show that $A^*(\{\%\} \cup B) = X$. For $i \in \{0, 1, 2\}$, let the language Y_i be $\{ w \in \{\mathsf{0}, \mathsf{1}\}^* \mid w \in X$ and $\mathbf{diff}(w) = i \}$.

Lemma ES2.3.1

(1) $\% \in Y_0$.

(2) $\mathsf{1} \in Y_1$.

(3) *For all* $i \in \{0, 1, 2\}$, $Y_0 Y_i \subseteq Y_i$.

(4) $Y_0 X \subseteq X$.

(5) *For all* $n \in \mathbb{N}$, ${Y_0}^n \subseteq Y_0$.

(6) ${Y_0}^* \subseteq Y_0$.

(7) $Y_1\{\mathsf{0}\} \subseteq Y_0$.

(8) $Y_1\{\mathsf{10}\} \subseteq Y_1$.

(9) *For all* $n \in \mathbb{N}$, $Y_1\{\mathsf{10}\}^n \subseteq Y_1$.

(10) $Y_1\{\mathsf{10}\}^* \subseteq Y_1$.

(11) $Y_1\{\mathsf{1}\} \subseteq Y_2$

(12) $\{\mathsf{1}\}\{\mathsf{10}\}^* \subseteq Y_1$.

(13) $A \subseteq Y_0$.

(14) $A^* \subseteq Y_0$.

(15) $B \subseteq X$.

(16) $\{\%\} \cup B \subseteq X$.

(17) $A^*(\{\%\} \cup B) \subseteq X$.

Proof.

(1) Because $\%$ is the only prefix of itself, $\mathbf{diff}(\%) = 0$ and $0 \leq 0 \leq 2$, it follows that $\% \in X$. And $\mathbf{diff}(\%) = 0$, completing the proof that $\% \in Y_0$.

(2) Because $\%$ and $\mathsf{1}$ are the only prefixes of $\mathsf{1}$, $\mathbf{diff}(\%) = 0$, $\mathbf{diff}(\mathsf{1}) = 1$, $0 \leq 0 \leq 2$ and $0 \leq 1 \leq 2$, it follows that $\mathsf{1} \in X$. And $\mathbf{diff}(\mathsf{1}) = 1$, completing the proof that $\mathsf{1} \in Y_1$.

(3) Suppose $i \in \{0, 1, 2\}$ and $w \in Y_0 Y_i$. Thus $w = xy$ for some $x \in Y_0$ and $y \in Y_i$. Hence $\mathbf{diff}(w) = \mathbf{diff}(x) + \mathbf{diff}(y) = 0 + i = i$. So, to conclude that $w \in Y_i$, it remains to show that $w \in X$. Suppose v is a prefix of w. We must show that $0 \leq \mathbf{diff}(v) \leq 2$. Because $w = xy$, there are two cases to consider.

- Suppose v is a prefix of x. Because $x \in Y_0 \subseteq X$, we have that $0 \leq \mathbf{diff}(v) \leq 2$.

- Suppose $v = xu$, for some prefix u of y. Because $y \in Y_i \subseteq X$, we have that $0 \leq \mathbf{diff}(u) \leq 2$. Thus, since $\mathbf{diff}(v) = \mathbf{diff}(x) + \mathbf{diff}(u) = 0 + \mathbf{diff}(u) = \mathbf{diff}(u)$, it follows that $0 \leq \mathbf{diff}(v) \leq 2$.

(4) Suppose $w \in Y_0X$. Thus $w = xy$ for some $x \in Y_0$ and $y \in X$. Because $y \in X$ and y is a prefix of itself, it follows that $0 \leq \mathbf{diff}(y) \leq 2$, i.e., $\mathbf{diff}(y) \in \{0, 1, 2\}$. And, because $y \in X$ and $\mathbf{diff}(y) = \mathbf{diff}(y)$, we have that $y \in Y_{\mathbf{diff}(y)}$. Hence, by Part (3), it follows that $w = xy \in Y_0Y_{\mathbf{diff}(y)} \subseteq Y_{\mathbf{diff}(y)} \subseteq X$.

(5) We proceed by mathematical induction.

(Basis Step) We have that ${Y_0}^0 = \{\%\} \subseteq Y_0$, by Part (1).

(Inductive Step) Suppose $n \in \mathbb{N}$, and assume the inductive hypothesis: ${Y_0}^n \subseteq Y_0$. Then ${Y_0}^{n+1} = {Y_0}^n Y_0 \subseteq Y_0Y_0 \subseteq Y_0$, by the inductive hypothesis and Part (3).

(6) Suppose $w \in {Y_0}^*$. Thus $w \in {Y_0}^n$ for some $n \in \mathbb{N}$. Hence, by Part (5), we have that $w \in {Y_0}^n \subseteq Y_0$.

(7) Suppose $w \in Y_1\{\mathsf{0}\}$. Thus $w = x\mathsf{0}$ for some $x \in Y_1$. Because $\mathbf{diff}(w) = \mathbf{diff}(x) + -1 = 1 + -1 = 0$, it remains to show that $w \in X$. Suppose v is a prefix of w. We must show that $0 \leq \mathbf{diff}(v) \leq 2$. Because $w = x\mathsf{0}$, there are two cases to consider.

- Suppose v is a prefix of x. Because $x \in Y_1 \subseteq X$, it follows that $0 \leq \mathbf{diff}(v) \leq 2$.
- Suppose $v = x\mathsf{0}$. Thus $v = x\mathsf{0} = w$, so that $\mathbf{diff}(v) = \mathbf{diff}(w) = 0$. Hence $0 \leq \mathbf{diff}(v) \leq 2$.

(8) Suppose $w \in Y_1\{\mathsf{10}\}$. Thus $w = x\mathsf{10}$ for some $x \in Y_1$. Because $\mathbf{diff}(w) = \mathbf{diff}(x) + 1 + -1 = \mathbf{diff}(x) = 1$, it remains to show that $w \in X$. Suppose v is a prefix of w. We must show that $0 \leq \mathbf{diff}(v) \leq 2$. Because $w = x\mathsf{10}$, there are three cases to consider.

- Suppose v is a prefix of x. Because $x \in Y_1 \subseteq X$, it follows that $0 \leq \mathbf{diff}(v) \leq 2$.
- Suppose $v = x\mathsf{1}$. Thus $\mathbf{diff}(v) = \mathbf{diff}(x) + 1 = 1 + 1 = 2$, so that $0 \leq \mathbf{diff}(v) \leq 2$.
- Suppose $v = x\mathsf{10}$. Thus $v = x\mathsf{10} = w$, so that $\mathbf{diff}(v) = \mathbf{diff}(w) = 1$. Hence $0 \leq \mathbf{diff}(v) \leq 2$.

(9) We proceed by mathematical induction.

(Basis Step) We have that $Y_1\{\mathsf{10}\}^0 = Y_1\{\%\} = Y_1 \subseteq Y_1$.

(Inductive Step) Suppose $n \in \mathbb{N}$, and assume the inductive hypothesis: $Y_1\{\mathsf{10}\}^n \subseteq Y_1$. Then $Y_1\{\mathsf{10}\}^{n+1} = Y_1\{\mathsf{10}\}^n\{\mathsf{10}\} \subseteq Y_1\{\mathsf{10}\} \subseteq Y_1$, by the inductive hypothesis and Part (8).

(10) Suppose $w \in Y_1\{\mathsf{10}\}^*$. Thus $w = xy$ for some $x \in Y_1$ and $y \in \{\mathsf{10}\}^*$. Thus $y \in \{\mathsf{10}\}^n$ for some $n \in \mathbb{N}$. Hence $w = xy \in Y_1\{\mathsf{10}\}^n \subseteq Y_1$, by Part (9).

(11) Suppose $w \in Y_1\{\mathsf{1}\}$. Thus $w = x\mathsf{1}$ for some $x \in Y_1$. Because $\mathbf{diff}(w) = \mathbf{diff}(x)+1 = 1+1 = 2$, it remains to show that $w \in X$. Suppose v is a prefix of w. We must show that $0 \leq \mathbf{diff}(v) \leq 2$. Because $w = x\mathsf{1}$, there are two cases to consider.

- Suppose v is a prefix of x. Because $x \in Y_1 \subseteq X$, it follows that $0 \leq \mathbf{diff}(v) \leq 2$.

- Suppose $v = x\mathsf{1}$. Thus $v = x\mathsf{1} = w$, so that $\mathbf{diff}(v) = \mathbf{diff}(w) = 2$. Hence $0 \leq \mathbf{diff}(v) \leq 2$.

(12) We have that $\{\mathsf{1}\}\{\mathsf{10}\}^* \subseteq Y_1\{\mathsf{10}\}^* \subseteq Y_1$, by Parts (2) and (10).

(13) By Parts (2), (10) and (7), we have that $A = \{\mathsf{1}\}\{\mathsf{10}\}^*\{\mathsf{0}\} \subseteq Y_1\{\mathsf{10}\}^*\{\mathsf{0}\} \subseteq Y_1\{\mathsf{0}\} \subseteq Y_0$.

(14) We have that $A^* \subseteq {Y_0}^* \subseteq Y_0$, by Parts (13) and (6).

(15) We have that

$$\begin{aligned}
B &= \{\mathsf{1}\}\{\mathsf{10}\}^*\{\%, \mathsf{1}\} \\
&\subseteq Y_1\{\mathsf{10}\}^*\{\%, \mathsf{1}\} && \text{(Part (2))} \\
&\subseteq Y_1\{\%, \mathsf{1}\} && \text{(Part (10))} \\
&= Y_1(\{\%\} \cup \{\mathsf{1}\}) \\
&= Y_1\{\%\} \cup Y_1\{\mathsf{1}\} && \text{(Proposition 3.2.6(1))} \\
&= Y_1 \cup Y_1\{\mathsf{1}\} \\
&\subseteq Y_1 \cup Y_2 && \text{(Part (11))} \\
&\subseteq X \cup X \\
&= X.
\end{aligned}$$

(16) We have that $\{\%\} \cup B \subseteq Y_0 \cup X \subseteq X \cup X = X$, by Parts (1) and (15).

(17) We have that $A^*(\{\%\} \cup B) \subseteq Y_0X \subseteq X$, by Parts (14), (16) and (4).

□

Lemma ES2.3.2
$X \subseteq A^*(\{\%\} \cup B)$.

Proof. Since $X \subseteq \{\mathsf{0}, \mathsf{1}\}^*$, it will suffice to show that, for all $w \in \{\mathsf{0}, \mathsf{1}\}^*$,

$$\text{if } w \in X, \text{ then } w \in A^*(\{\%\} \cup B).$$

We prove this using strong string induction. Suppose $w \in \{\mathsf{0}, \mathsf{1}\}^*$, and assume the inductive hypothesis: for all $x \in \{\mathsf{0}, \mathsf{1}\}^*$, if $|x| < |w|$, then

$$\text{if } x \in X, \text{ then } x \in A^*(\{\%\} \cup B).$$

We must show that

$$\text{if } w \in X, \text{ then } w \in A^*(\{\%\} \cup B).$$

Suppose $w \in X$. We must show that $w \in A^*(\{\%\} \cup B)$. There are three cases to consider.

- Suppose $w = \%$. Then $w = \% = \%\% \in A^*(\{\%\} \cup B)$.
- Suppose $w = \mathsf{0}t$ for some $t \in \{\mathsf{0}, \mathsf{1}\}^*$. Because $w \in X$ and $\mathsf{0}$ is a prefix of w, it follows that $0 \leq \mathbf{diff}(\mathsf{0}) = -1$—contradiction. Thus $w \in A^*(\{\%\} \cup B)$.

- Suppose $w = 1t$ for some $t \in \{0,1\}^*$. Let x be the longest prefix of t such that $x \in \{10\}^*$. (Such an x exists, because % is a prefix of t that is in $\{10\}^*$.) Let $u \in \{0,1\}^*$ be such that $t = xu$. Thus $w = 1t = 1xu$. There are three subcases to consider.
  - Suppose $u = \%$. Since $1x = 1x\% \in \{1\}\{10\}^*\{\%, 1\} = B \subseteq \{\%\} \cup B$, we have that $w = 1xu = 1x\% = \%(1x) \in A^*(\{\%\} \cup B)$.
  - Suppose $u = 0y$ for some $y \in \{0,1\}^*$. Thus $w = 1xu = 1x0y$. We have that $1x0 \in \{1\}\{10\}^*\{0\} = A$. By Lemma ES2.3.1(13), it follows that $A \subseteq Y_0$. Hence $1x0 \in Y_0$, so that $\mathbf{diff}(1x0) = 0$.

    To see that $y \in X$, suppose v is a prefix of y. We must show that $0 \leq \mathbf{diff}(v) \leq 2$. Because v is a prefix of y, it follows that $1x0v$ is a prefix of $1x0y = w$. Thus, since $w \in X$, we have that $0 \leq \mathbf{diff}(1x0v) \leq 2$. But $\mathbf{diff}(v) = 0 + \mathbf{diff}(v) = \mathbf{diff}(1x0) + \mathbf{diff}(v) = \mathbf{diff}(1x0v)$, and thus $0 \leq \mathbf{diff}(v) \leq 2$.

    Because $y \in X$ and $|y| < |w|$, the inductive hypothesis tells us that $y \in A^*(\{\%\} \cup B)$. Hence $w = (1x0)y \in AA^*(\{\%\} \cup B) \subseteq A^*(\{\%\} \cup B)$.
  - Suppose $u = 1y$ for some $y \in \{0,1\}^*$. Thus $t = xu = x1y$ and $w = 1xu = 1x1y$. There are three sub-subcases to consider.
    - Suppose $y = \%$. Then $w = 1x1y = 1x1\% = 1x1$. We have that $1x1 \in \{1\}\{10\}^*\{\%, 1\} = B \subseteq \{\%\} \cup B$. Hence $w = \%(1x1) \in A^*(\{\%\} \cup B)$.
    - Suppose $y = 0z$, for some $z \in \{0,1\}^*$. Thus $t = x1y = x10z$. But $x10 \in \{10\}^*\{10\} \subseteq \{10\}^*$ and $x10$ is a longer prefix of t than x, contradicting the definition of x. Thus $w \in A^*(\{\%\} \cup B)$.
    - Suppose $y = 1z$, for some $z \in \{0,1\}^*$. Thus $w = 1x1y = 1x11z$. By Parts (2) and (10) of Lemma ES2.3.1, we have that $1x \in Y_1\{10\}^* \subseteq Y_1$. Thus $\mathbf{diff}(1x11) = \mathbf{diff}(1x) + \mathbf{diff}(11) = 1 + 2 = 3$. But $1x11$ is a prefix of $1x11z = w \in X$, and thus $3 = \mathbf{diff}(1x11) \leq 2$—contradiction. Thus $w \in A^*(\% \cup B)$.

□

Proposition ES2.3.3
$A^*(\{\%\} \cup B) = X$.

Proof. By Lemma ES2.3.1(17) and Lemma ES2.3.2, we have that $A^*(\{\%\} \cup B) \subseteq X \subseteq A^*(\{\%\} \cup B)$. Thus $A^*(\{\%\} \cup B) = X$. □

Exercise 4

Here is the text of the file `es2-ex4.sml`:

```
structure ES2Ex4 =
struct

(* val zero : sym is the symbol 0
   val one  : sym is the symbol 1 *)
```

```
val zero = Sym.fromString "0"
val one  = Sym.fromString "1"

(* val isZero : sym -> bool

   isZero a tests whether a is zero *)

fun isZero a = Sym.equal(a, zero)

(* val isOne : sym -> bool

   isOne a tests whether a is one *)

fun isOne a = Sym.equal(a, one)

(* val diffSym : sym -> int

   if diffSym is called with zero or one, then it returns the diff of
   its argument; otherwise, it returns 0 *)

fun diffSym a =
      if isZero a
        then ~1
      else if isOne a
        then 1
      else 0

(* val validStr : str -> bool

   validStr w tests whether w is in Y, returning true if it is, and
   false otherwise; when w is not in Y, it prints an explanation of
   why w fails to be in Y on the standard output *)

fun validStr w =
      let (* val valid : str * int * str -> bool

             in a call valid(ds, n, cs), diff ds = n and w = ds @ cs

             if every nonempty prefix of cs ends with a zero or one,
             and has a diff that when added to n is <= 1, then

               if n + diff cs = 1, then valid silently returns true

               otherwise, valid complains that w has a diff of n
               instead of 1, and returns false

             otherwise, let es be the shortest nonempty prefix of cs
             such that either es doesn't end with a zero or one, or
```

```
          the n + diff es > 1

          if the last symbol of es isn't a zero or one, then
          valid complains about this symbol of w

          otherwise, valid compains that ds @ es is a prefix of w
          with a diff that is > 1 *)

        fun valid(_, n, nil)      =
              if n = 1
              then true
              else (print "diff of string is ";
                    print(Int.toString n); print " not 1\n";
                    false)
          | valid(ds, n, c :: cs) =
              let val m  = n + diffSym c
                  val ds = ds @ [c]
              in if m = n
                   then (print "string has symbol other than 0/1 : ";
                         print(Sym.toString c); print "\n";
                         false)
                 else if m > 1
                   then (print "prefix "; print(Str.toString ds);
                         print " of string has diff "; print(Int.toString m);
                         print " which is greater-than 1\n";
                         false)
                 else valid(ds, m, cs)
              end
      in valid(nil, 0, w) end

(* val shortestPrefix : (int -> bool) -> str -> str * str

   if w is an str of zeros and ones, and there is a prefix x of w such
   that f(diff x), then shortestPrefix f w returns (x, y), where x
   is the shortest such prefix, and y is such that x @ y = w *)

fun shortestPrefix f w =
      let (* val short : str * int * str -> str * str

          if cs is a list of zeros and ones, and there is a
          prefix ds of cs such that f(n + diff ds), then
          short(bs, n, cs) returns (bs @ ds, es), where ds is the
          shortest such prefix, and es is such that cs = ds @ es *)

        fun short(bs, n, nil)             =
              if f n then (bs, nil) else raise Fail "shouldn't happen"
          | short(bs, n, c_cs as c :: cs) =
              if f n
```

```
                then (bs, c_cs)
                else short(bs @ [c], n + diffSym c, cs)
      in short(nil, 0, w) end

(* val splitPositive : str -> str * str

   if w is an str of zeros and ones such that diff w >= 1, then
   splitPositive w returns a pair (x, y) such that w = x @ y,
   x is in Y and diff y = diff w - 1 *)

val splitPositive = shortestPrefix(fn n => n >= 1)

(* val indent : int -> unit

   indent n prints n spaces *)

fun indent n = print(StringCvt.padLeft #" " n "")

(* val text1 : int -> unit

   print explanation based on rule (1) of X's definition at given indentation *)

fun text1 ind = (indent ind; print "1 is in X, by (1)\n")

(* val text2 : int * str * str * str -> unit

   print explanation based on rule (2) of X's definition at given indentation *)

fun text2(ind, w, x, y) =
      (indent ind;
       print(Str.toString w ^ " = " ^
             Str.toString x ^ " @ 0 @ " ^
             Str.toString y ^ " is in X, by (2)\n"))

(* val text3 : int * str * str * str -> unit

   print explanation based on rule (3) of X's definition at given indentation *)

fun text3(ind, w, x, y) =
      (indent ind;
       print(Str.toString w ^ " = 0 @ " ^
             Str.toString x ^ " @ " ^
             Str.toString y ^ " is in X, by (3)\n"))

(* val expl : int * str -> unit

   if ind >= 0 and w is in Y, then expl(ind, w) prints an explanation
   of why w is in X, indented ind spaces; in recursive calls, the size
```

```
   of the str goes down *)

fun expl(ind, w) =
      if Str.isEmpty w
        then raise Fail "cannot happen"
      else if isZero(hd w)
        then let val t      = tl w  (* w = [zero] @ t, diff t = 2 *)
                 val (x, y) = splitPositive t  (* t = x @ y *)
                 (* x, y are in Y, w = [zero] @ x @ y *)
             in text3(ind, w, x, y); expl(ind + 2, x); expl(ind + 2, y)
             end
      else (* isOne(hd w) *)
           let val t = tl w  (* w = [one] @ t, diff t = 0 *)
           in if Str.isEmpty t  (* w = [one] *)
                then text1 ind
              else if isZero(hd t)
                then let val y = tl t  (* t = [zero] @ y, w = [one, zero] @ y *)
                         (* y is in Y *)
                     in text2(ind, w, [one], y); text1(ind + 2);
                        expl(ind + 2, y)
                     end
              else (* isOne(hd t), w begins with [one, one] *)
                   raise Fail "cannot happen"
           end

(* val explain : unit -> unit

   explain() reads an str from the standard input; if the str is not
   in Y, it prints an error message; otherwise, it prints an
   explanation of why the str is in X *)

fun explain() =
      let val w = Str.input ""
      in if validStr w then expl(0, w) else () end

end;
```

And, here is how the program was tested:

```
- use "es2-ex4.sml";
[opening es2-ex4.sml]
[autoloading]
[autoloading done]
structure ES2Ex4 :
  sig
    val zero : sym
    val one : sym
    val isZero : sym -> bool
    val isOne : sym -> bool
```

```
    val diffSym : sym -> int
    val validStr : sym list -> bool
    val shortestPrefix : (int -> bool) -> sym list -> sym list * sym list
    val splitPositive : sym list -> sym list * sym list
    val indent : int -> unit
    val text1 : int -> unit
    val text2 : int * str * str * str -> unit
    val text3 : int * str * str * str -> unit
    val expl : int * str -> unit
    val explain : unit -> unit
  end
val it = () : unit
- ES2Ex4.explain();
@ 100011101
@ .
100011101 = 1 @ 0 @ 0011101 is in X, by (2)
  1 is in X, by (1)
  0011101 = 0 @ 011 @ 101 is in X, by (3)
    011 = 0 @ 1 @ 1 is in X, by (3)
      1 is in X, by (1)
      1 is in X, by (1)
    101 = 1 @ 0 @ 1 is in X, by (2)
      1 is in X, by (1)
      1 is in X, by (1)
val it = () : unit
- ES2Ex4.explain();
@ 0110
@ .
diff of string is 0 not 1
val it = () : unit
- ES2Ex4.explain();
@ 01110
@ .
prefix 0111 of string has diff 2 which is greater-than 1
val it = () : unit
- ES2Ex4.explain();
@ 1020
@ .
string has symbol other than 0/1 : 2
val it = () : unit
- ES2Ex4.explain();
@ %
@ .
diff of string is 0 not 1
val it = () : unit
- ES2Ex4.explain();
@ 000111101
@ .
```

```
000111101 = 0 @ 00111 @ 101 is in X, by (3)
  00111 = 0 @ 011 @ 1 is in X, by (3)
    011 = 0 @ 1 @ 1 is in X, by (3)
      1 is in X, by (1)
      1 is in X, by (1)
    1 is in X, by (1)
  101 = 1 @ 0 @ 1 is in X, by (2)
    1 is in X, by (1)
    1 is in X, by (1)
val it = () : unit
```