COMP 150-EFL—Experimenting with Formal Languages—Spring 2012

Problem Set 2

Model Answers

Problem 1

Part (1)

It will suffice to use induction on X to show that, for all $w \in X$, $w \in Y$. There are five steps to show.

(1) We must show that $\% \in Y$, and this follows since $\% \in \{\mathsf{0},\mathsf{1}\}^*$ and $\mathbf{diff}\,\% = 0$.

(2) Suppose $x, y \in X$, and assume the inductive hypothesis: $x, y \in Y$. We must show that $\mathsf{0}x\mathsf{0}y\mathsf{1} \in Y$. Because $x, y \in Y$, it follows that $\mathsf{0}x\mathsf{0}y\mathsf{1} \in \{\mathsf{0},\mathsf{1}\}^*$. And, since $x, y \in Y$, we have that $\mathbf{diff}\,x = \mathbf{diff}\,y = 0$, so that $\mathbf{diff}(\mathsf{0}x\mathsf{0}y\mathsf{1}) = \mathbf{diff}\,\mathsf{0} + \mathbf{diff}\,x + \mathbf{diff}\,\mathsf{0} + \mathbf{diff}\,y + \mathbf{diff}\,\mathsf{1} = 1 + 0 + 1 + 0 + -2 = 0$, completing the proof that $\mathsf{0}x\mathsf{0}y\mathsf{1} \in Y$.

(3) Suppose $x, y \in X$, and assume the inductive hypothesis: $x, y \in Y$. We must show that $\mathsf{0}x\mathsf{1}y\mathsf{0} \in Y$. Because $x, y \in Y$, it follows that $\mathsf{0}x\mathsf{1}y\mathsf{0} \in \{\mathsf{0},\mathsf{1}\}^*$. And, since $x, y \in Y$, we have that $\mathbf{diff}\,x = \mathbf{diff}\,y = 0$, so that $\mathbf{diff}(\mathsf{0}x\mathsf{1}y\mathsf{0}) = \mathbf{diff}\,\mathsf{0} + \mathbf{diff}\,x + \mathbf{diff}\,\mathsf{1} + \mathbf{diff}\,y + \mathbf{diff}\,\mathsf{0} = 1 + 0 + -2 + 0 + 1 = 0$, completing the proof that $\mathsf{0}x\mathsf{1}y\mathsf{0} \in Y$.

(4) Suppose $x, y \in X$, and assume the inductive hypothesis: $x, y \in Y$. We must show that $\mathsf{1}x\mathsf{0}y\mathsf{0} \in Y$. Because $x, y \in Y$, it follows that $\mathsf{1}x\mathsf{0}y\mathsf{0} \in \{\mathsf{0},\mathsf{1}\}^*$. And, since $x, y \in Y$, we have that $\mathbf{diff}\,x = \mathbf{diff}\,y = 0$, so that $\mathbf{diff}(\mathsf{1}x\mathsf{0}y\mathsf{0}) = \mathbf{diff}\,\mathsf{1} + \mathbf{diff}\,x + \mathbf{diff}\,\mathsf{0} + \mathbf{diff}\,y + \mathbf{diff}\,\mathsf{0} = -2 + 0 + 1 + 0 + 1 = 0$, completing the proof that $\mathsf{1}x\mathsf{0}y\mathsf{0} \in Y$.

(5) Suppose $x, y \in X$, and assume the inductive hypothesis: $x, y \in Y$. We must show that $xy \in Y$. Because $x, y \in Y$, it follows that $xy \in \{\mathsf{0},\mathsf{1}\}^*$. And, since $x, y \in Y$, we have that $\mathbf{diff}\,x = \mathbf{diff}\,y = 0$, so that $\mathbf{diff}(xy) = \mathbf{diff}\,x + \mathbf{diff}\,y = 0 + 0 = 0$, completing the proof that $xy \in Y$.

Part (2)

We begin by proving a useful lemma:

Lemma PS2.1.1
For all $w \in \{\mathsf{0},\mathsf{1}\}^*$, if $\mathbf{diff}\,w \geq 1$, then $w = x\mathsf{0}y$, for some $x, y \in \{\mathsf{0},\mathsf{1}\}^*$ such that $\mathbf{diff}\,x = 0$ and $\mathbf{diff}\,y = \mathbf{diff}\,w - 1$.

Proof. Suppose $w \in \{\mathsf{0},\mathsf{1}\}^*$ and $\mathbf{diff}\,w \geq 1$. Let $u \in \{\mathsf{0},\mathsf{1}\}^*$ be the shortest prefix of w such that $\mathbf{diff}\,u \geq 1$, and let $y \in \{\mathsf{0},\mathsf{1}\}^*$ be such that $w = uy$. Then $u \neq \%$, so that $u = xb$ for some $x \in \{\mathsf{0},\mathsf{1}\}^*$ and $b \in \{\mathsf{0},\mathsf{1}\}$. Thus $w = uy = xby$. Since x is a shorter prefix of w than u, we have that $\mathbf{diff}\,x \leq 0$.

Suppose, toward a contradiction, that $b = \mathsf{1}$. Then $\mathbf{diff}\, x + -2 = \mathbf{diff}(x\mathsf{1}) = \mathbf{diff}(x\mathsf{b}) = \mathbf{diff}\, u \geq 1$, so that $\mathbf{diff}\, x \geq 3$—contradiction. Thus $b = \mathsf{0}$.

Summarizing, we have that $u = x\mathsf{b} = x\mathsf{0}$, $w = uy = x\mathsf{0}y$, $\mathbf{diff}\, u \geq 1$, $\mathbf{diff}\, w \geq 1$ and $\mathbf{diff}\, x \leq 0$. Since $\mathbf{diff}\, x{+}1 = \mathbf{diff}(x\mathsf{0}) = \mathbf{diff}\, u \geq 1$, we have that $\mathbf{diff}\, x \geq 0$. But $\mathbf{diff}\, x \leq 0$, and thus $\mathbf{diff}\, x = 0$. Finally, since $\mathbf{diff}\, w = \mathbf{diff}(x\mathsf{0}y) = 0 + 1 + \mathbf{diff}\, y = 1 + \mathbf{diff}\, y$, we have that $\mathbf{diff}\, y = \mathbf{diff}\, w - 1$. □

Now, we use the lemma to prove that $Y \subseteq X$. Since $Y \subseteq \{\mathsf{0},\mathsf{1}\}^*$, it will suffice to show that, for all $w \in \{\mathsf{0},\mathsf{1}\}^*$,

$$\text{if } w \in Y, \text{ then } w \in X.$$

We proceed by 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 is a proper substring of w, then

$$\text{if } x \in Y, \text{ then } x \in X.$$

We must show that

$$\text{if } w \in Y, \text{ then } w \in X.$$

Suppose $w \in Y$. We must show that $w \in X$. There are three cases to consider.

- Suppose $w = \%$. Then $w = \% \in X$, by part (1) of the definition of X.
- Suppose $w = \mathsf{0}t$, for some $t \in \{\mathsf{0},\mathsf{1}\}^*$. Since $1 + \mathbf{diff}\, t = \mathbf{diff}(\mathsf{0}t) = \mathbf{diff}\, w = 0$, we have that $\mathbf{diff}\, t = -1$. Let $u \in \{\mathsf{0},\mathsf{1}\}^*$ be the shortest prefix of t such that $\mathbf{diff}\, u \leq -1$, and let $v \in \{\mathsf{0},\mathsf{1}\}^*$ be such that $t = uv$. Then $u \neq \%$, so that $u = x\mathsf{b}$ for some $x \in \{\mathsf{0},\mathsf{1}\}^*$ and $\mathsf{b} \in \{\mathsf{0},\mathsf{1}\}$. Hence $t = uv = x\mathsf{b}v$. Since x is a shorter prefix of t than u, we have that $\mathbf{diff}\, x \geq 0$.

  Suppose, toward a contradiction, that $b = \mathsf{0}$. Since $\mathbf{diff}\, x{+}1 = \mathbf{diff}(x\mathsf{0}) = \mathbf{diff}(x\mathsf{b}) = \mathbf{diff}\, u \leq -1$, we have that $\mathbf{diff}\, x \leq -2$. But $\mathbf{diff}\, x \geq 0$—contradiction. Thus $b = \mathsf{1}$.

  Summarizing, we have that $u = x\mathsf{b} = x\mathsf{1}$, $t = uv = x\mathsf{1}v$, $w = \mathsf{0}t = \mathsf{0}x\mathsf{1}v$, $\mathbf{diff}\, t = -1$, $\mathbf{diff}\, u \leq -1$ and $\mathbf{diff}\, x \geq 0$. Since $\mathbf{diff}\, x + -2 = \mathbf{diff}(x\mathsf{1}) = \mathbf{diff}\, u \leq -1$, we have that $\mathbf{diff}\, x \leq 1$. But $\mathbf{diff}\, x \geq 0$, and thus we have that $\mathbf{diff}\, x \in \{0, 1\}$. Hence there are two sub-cases to consider.

  - Suppose $\mathbf{diff}\, x = 0$. Because $-2 + \mathbf{diff}\, v = 0 + -2 + \mathbf{diff}\, v = \mathbf{diff}(x\mathsf{1}v) = \mathbf{diff}\, t = -1$, we have that $\mathbf{diff}\, v = 1$. Since $\mathbf{diff}\, v \geq 1$, Lemma PS2.1.1 tells us that $v = y\mathsf{0}z$, for some $y, z \in \{\mathsf{0},\mathsf{1}\}^*$ such that $\mathbf{diff}\, y = 0$ and $\mathbf{diff}\, z = \mathbf{diff}\, v - 1$. Hence $w = \mathsf{0}x\mathsf{1}v = \mathsf{0}x\mathsf{1}y\mathsf{0}z$ and $\mathbf{diff}\, z = 0$. Since $\mathbf{diff}\, x = \mathbf{diff}\, y = \mathbf{diff}\, z = 0$, we have that $x, y, z \in Y$. Because x, y and z are proper substrings of w, the inductive hypothesis tells us that $x, y, z \in X$. By part (3) of the definition of X, we have that $\mathsf{0}x\mathsf{1}y\mathsf{0} \in X$. Thus, by part (5) of the definition of X, we can conclude that $w = \mathsf{0}x\mathsf{1}y\mathsf{0}z = (\mathsf{0}x\mathsf{1}y\mathsf{0})z \in X$.
  - Suppose $\mathbf{diff}\, x = 1$. Because $-1 + \mathbf{diff}\, v = 1 + -2 + \mathbf{diff}\, v = \mathbf{diff}(x\mathsf{1}v) = \mathbf{diff}\, t = -1$, we have that $\mathbf{diff}\, v = 0$. Since $\mathbf{diff}\, x = 1$, we have that $x \neq \%$, so that $x = cy$ for some $c \in \{\mathsf{0},\mathsf{1}\}$ and $y \in \{\mathsf{0},\mathsf{1}\}^*$. Because c is a prefix of x, it follows that c is a shorter prefix of t than u.

    Suppose, toward a contradiction, that $c = \mathsf{1}$. Then $\mathbf{diff}\, c = -2 \leq -1$, contradicting the definition of u. Thus $c = \mathsf{0}$, so that $x = \mathsf{0}y$.

Because $1 + \mathbf{diff}\, y = \mathbf{diff}(0y) = \mathbf{diff}\, x = 1$, we have that $\mathbf{diff}\, y = 0$. Hence $w = 0x1v = 00y1v = (0\%0y1)v$. Since $\mathbf{diff}\, y = 0$ and $\mathbf{diff}\, v = 0$, we have that $y, v \in Y$. Because y and v are proper substrings of w, the inductive hypothesis tells us that $y, v \in X$. Since $\% \in X$ (by part (1) of the definition of X) and $y \in X$, part (2) of the definition of X tells us that $0\%0y1 \in X$. Thus, by part (5) of the definition of X, we can conclude that $w = (0\%0y1)v \in X$.

- Suppose $w = 1t$, for some $t \in \{0,1\}^*$. Since $-2 + \mathbf{diff}\, t = \mathbf{diff}(1t) = \mathbf{diff}\, w = 0$, we have that $\mathbf{diff}\, t = 2$. Because $\mathbf{diff}\, t \geq 1$, Lemma PS2.1.1 tells us that $t = x0u$, for some $x, u \in \{0,1\}^*$ such that $\mathbf{diff}\, x = 0$ and $\mathbf{diff}\, u = \mathbf{diff}\, t - 1$. Hence $\mathbf{diff}\, u = 1$. Because $\mathbf{diff}\, u \geq 1$, Lemma PS2.1.1 tells us that $u = y0z$, for some $y, z \in \{0,1\}^*$ such that $\mathbf{diff}\, y = 0$ and $\mathbf{diff}\, z = \mathbf{diff}\, u - 1$. Hence $\mathbf{diff}\, z = 0$.

  Summarizing, we have that $w = 1t = 1x0u = 1x0y0z$ and $x, y, z \in Y$. Since x, y and z are proper substrings of w, the inductive hypothesis tells us that $x, y, z \in X$. By part (4) of the definition of X, we have that $1x0y0 \in X$. Thus, by part (5) of the definition of X, we can conclude that $w = 1x0y0z = (1x0y0)z \in X$.

Note that, in the preceding proof, we only use part (2) of X's definition in the case when $x = \%$.

Problem 2

Here is `ps2-explain.sml`:

```
(* ps2-explain.sml

   Solution to Problem 2 of Problem Set 2. *)

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

   If w is an str of zero's and one's, and there is a prefix x of w
   such that f(diff x), then shortest f w returns (x, y), where x is
   the shortest such prefix and y is such that x @ y = w. *)

fun shortest f (w : str) : str * str =
      let fun short(bs, n, nil)     =
                if f n then (bs, nil) else raise Fail "shouldn't happen"
            | short(bs, n, c :: cs) =
                if f n
                then (bs, c :: cs)
                else short(bs @ [c], n + diffSym c, cs)
      in short(nil, 0, w) end

(* val shortestPositive : str -> str * str

   If w is an str of zero's and one's, and there is a prefix x of w
   such that diff x >= 1, then shortestPositive w returns (x, y),
   where x is the shortest such prefix and y is such that x @ y = w. *)
```

```
val shortestPositive = shortest(fn n => n >= 1)

(* val shortestNegative : str -> str * str

   If w is an str of zero's and one's, and there is a prefix x of w
   such that diff x <= ~1, then shortestNegative w returns (x, y),
   where x is the shortest such prefix and y is such that x @ y = w. *)

val shortestNegative = shortest(fn n => n <= ~1)

(* val splitPositive : str -> str * str

   If w is an str of zero's and one's and diff w >= 1, then
   splitPositive w returns a pair (x, y), such that w = x @ [zero] @ y,
   diff x = 0 and diff y = diff w - 1. *)

fun splitPositive w : str * str =
      let val (u, y) = shortestPositive w  (* w = u @ y *)
          (* diff u >= 1 and shorter prefixes of w have non-positive diffs *)
          val x      = Str.allButLast u  (* diff x = 0 *)
          val b      = Str.last u  (* isZero b, diff y = diff w - 1 *)
      in (x, y) end

(* val explain : str -> expl

   If w is in Y, then explain w returns an explanation expl such
   that Str.equal(strExplained expl, w). *)

fun explain (w : str) =
      if null w
        then Rule1
      else if isZero(hd w)
        then let val t      = tl w  (* w = [zero] @ t, diff t = ~1 *)
                 val (u, v) = shortestNegative t  (* t = u @ v *)
                 (* diff u <= ~1 and shorter prefixes of t than u have
                    non-negative diffs *)
                 val x      = Str.allButLast u  (* diff x in {0, 1} *)
                 val b      = Str.last u  (* isOne b *)
             in if diff x = 0
                then (* diff v = 1 *)
                     let val (y, z) = splitPositive v  (* v = y @ [zero] @ z *)
                         (* diff y = 0, diff z = 0,
                            w = ([zero] @ x @ [one] @ y @ [zero]) @ z *)
                     in Rule5(Rule3(explain x, explain y), explain z) end
                else (* diff x = 1, diff v = 0, x begins with 0 *)
                     let val y = tl x  (* diff y = 0 *)
                         (* w = ([zero] @ nil @ [zero] @ y @ [one]) @ v *)
```

```
                    in Rule5(Rule2(Rule1, explain y), explain v) end
              end
        else (* isOne(hd w) *)
             let val t      = tl w  (* w = [one] @ t, diff t = 2 *)
                 val (x, u) = splitPositive t  (* t = x @ [zero] @ u,
                                                  diff x = 0, diff u = 1 *)
                 val (y, z) = splitPositive u  (* u = y @ [zero] @ z,
                                                  diff y = 0, diff z = 0 *)
                 (* w = ([one] @ x @ [zero] @ y @ [zero]) @ z *)
             in Rule5(Rule4(explain x, explain y), explain z) end;
```

And here is how `explain` was tested:

```
- use "ps2-framework.sml";
[opening ps2-framework.sml]
exception Error
val zero = - : sym
val one = - : sym
val isZero = fn : sym -> bool
val isOne = fn : sym -> bool
val diffSym = fn : sym -> int
val diff = fn : str -> int
val validStr = fn : str -> bool
datatype expl
  = Rule1
  | Rule2 of expl * expl
  | Rule3 of expl * expl
  | Rule4 of expl * expl
  | Rule5 of expl * expl
val strExplained = fn : expl -> str
val printExplanation = fn : expl -> unit
val test = fn : (str -> expl) -> str -> unit
val it = () : unit
- use "ps2-explain.sml";
[opening ps2-explain.sml]
val shortest = fn : (int -> bool) -> str -> str * str
val shortestPositive = fn : str -> str * str
val shortestNegative = fn : str -> str * str
val splitPositive = fn : str -> str * str
val explain = fn : str -> expl
val it = () : unit
- val doit = test explain;
val doit = fn : str -> unit
- doit(Str.fromString "012");
str has symbol other than 0/1
val it = () : unit
- doit(Str.fromString "01");
str has non-zero diff : ~1
val it = () : unit
```

```
- doit(Str.fromString "0100");
str has non-zero diff : 1
val it = () : unit
- doit(Str.fromString "%");
% is in X, by rule (1)
val it = () : unit
- doit(Str.fromString "001");
001 = 001 @ % is in X, by rule (5)
  001 = 0 @ % @ 0 @ % @ 1 is in X, by rule (2)
    % is in X, by rule (1)
    % is in X, by rule (1)
  % is in X, by rule (1)
val it = () : unit
- doit(Str.fromString "010");
010 = 010 @ % is in X, by rule (5)
  010 = 0 @ % @ 1 @ % @ 0 is in X, by rule (3)
    % is in X, by rule (1)
    % is in X, by rule (1)
  % is in X, by rule (1)
val it = () : unit
- doit(Str.fromString "100");
100 = 100 @ % is in X, by rule (5)
  100 = 1 @ % @ 0 @ % @ 0 is in X, by rule (4)
    % is in X, by rule (1)
    % is in X, by rule (1)
  % is in X, by rule (1)
val it = () : unit
- doit(Str.fromString "001010");
001010 = 001 @ 010 is in X, by rule (5)
  001 = 0 @ % @ 0 @ % @ 1 is in X, by rule (2)
    % is in X, by rule (1)
    % is in X, by rule (1)
  010 = 010 @ % is in X, by rule (5)
    010 = 0 @ % @ 1 @ % @ 0 is in X, by rule (3)
      % is in X, by rule (1)
      % is in X, by rule (1)
    % is in X, by rule (1)
val it = () : unit
- doit(Str.fromString "010100");
010100 = 010 @ 100 is in X, by rule (5)
  010 = 0 @ % @ 1 @ % @ 0 is in X, by rule (3)
    % is in X, by rule (1)
    % is in X, by rule (1)
  100 = 100 @ % is in X, by rule (5)
    100 = 1 @ % @ 0 @ % @ 0 is in X, by rule (4)
      % is in X, by rule (1)
      % is in X, by rule (1)
    % is in X, by rule (1)
```

```
val it = () : unit
- doit(Str.fromString "100001");
100001 = 100 @ 001 is in X, by rule (5)
  100 = 1 @ % @ 0 @ % @ 0 is in X, by rule (4)
    % is in X, by rule (1)
    % is in X, by rule (1)
  001 = 001 @ % is in X, by rule (5)
    001 = 0 @ % @ 0 @ % @ 1 is in X, by rule (2)
      % is in X, by rule (1)
      % is in X, by rule (1)
    % is in X, by rule (1)
val it = () : unit
- doit(Str.fromString "100000011010");
100000011010 = 100 @ 000011010 is in X, by rule (5)
  100 = 1 @ % @ 0 @ % @ 0 is in X, by rule (4)
    % is in X, by rule (1)
    % is in X, by rule (1)
  000011010 = 000011 @ 010 is in X, by rule (5)
    000011 = 0 @ % @ 0 @ 001 @ 1 is in X, by rule (2)
      % is in X, by rule (1)
      001 = 001 @ % is in X, by rule (5)
        001 = 0 @ % @ 0 @ % @ 1 is in X, by rule (2)
          % is in X, by rule (1)
          % is in X, by rule (1)
        % is in X, by rule (1)
    010 = 010 @ % is in X, by rule (5)
      010 = 0 @ % @ 1 @ % @ 0 is in X, by rule (3)
        % is in X, by rule (1)
        % is in X, by rule (1)
      % is in X, by rule (1)
val it = () : unit
- doit(Str.fromString "110001000001000111000");
110001000001000111000 = 110001000 @ 001000111000 is in X, by rule (5)
  110001000 = 1 @ 100 @ 0 @ 100 @ 0 is in X, by rule (4)
    100 = 100 @ % is in X, by rule (5)
      100 = 1 @ % @ 0 @ % @ 0 is in X, by rule (4)
        % is in X, by rule (1)
        % is in X, by rule (1)
      % is in X, by rule (1)
    100 = 100 @ % is in X, by rule (5)
      100 = 1 @ % @ 0 @ % @ 0 is in X, by rule (4)
        % is in X, by rule (1)
        % is in X, by rule (1)
      % is in X, by rule (1)
  001000111000 = 001 @ 000111000 is in X, by rule (5)
    001 = 0 @ % @ 0 @ % @ 1 is in X, by rule (2)
      % is in X, by rule (1)
      % is in X, by rule (1)
```

```
000111000 = 000111000 @ % is in X, by rule (5)
  000111000 = 0 @ 001 @ 1 @ 100 @ 0 is in X, by rule (3)
    001 = 001 @ % is in X, by rule (5)
      001 = 0 @ % @ 0 @ % @ 1 is in X, by rule (2)
        % is in X, by rule (1)
        % is in X, by rule (1)
      % is in X, by rule (1)
    100 = 100 @ % is in X, by rule (5)
      100 = 1 @ % @ 0 @ % @ 0 is in X, by rule (4)
        % is in X, by rule (1)
        % is in X, by rule (1)
      % is in X, by rule (1)
  % is in X, by rule (1)
val it = () : unit
```

Note that the last two tests produce explanations using all five rules of X's definition.