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

Problem Set 2

Model Answers

Problem 1

Part (a)

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

(1) We must show that $\% \in Y$. We have that $\% \in \{0,1\}^*$ and $\mathbf{diff}\,\% = 0$. So it remains to show that, for all prefixes v of $\%$, $\mathbf{diff}\,v \leq 0$. Suppose v is a prefix of $\%$. Then $v = \%$, so that $\mathbf{diff}\,v = \mathbf{diff}\,\% = 0 \leq 0$.

(2) Suppose $x, y \in X$, and assume the inductive hypothesis: $x, y \in Y$. We must show that $x0y1 \in Y$. Because $x, y \in Y$, it follows that $x0y1 \in \{0,1\}^*$. And, since $x, y \in Y$, we have that $\mathbf{diff}\,x = \mathbf{diff}\,y = 0$, so that $\mathbf{diff}(x0y1) = \mathbf{diff}\,x + \mathbf{diff}\,0 + \mathbf{diff}\,y + \mathbf{diff}\,1 = 0 + -1 + 0 + 1 = 0$. It remains to show that, for all prefixes v of $x0y1$, $\mathbf{diff}\,v \leq 0$. Suppose v is a prefix of $x0y1$. There are three sub cases to consider.

- Suppose v is a prefix of x. Because $x \in Y$, it follows that $\mathbf{diff}\,v \leq 0$.
- Suppose $v = x0u$, for some prefix u of y. Because $x \in Y$, we have $\mathbf{diff}\,x = 0$. Because $y \in Y$ and u is a prefix of y, it follows that $\mathbf{diff}\,u \leq 0$. Thus $\mathbf{diff}\,v = \mathbf{diff}(x0u) = \mathbf{diff}\,x + \mathbf{diff}\,0 + \mathbf{diff}\,u = 0 + -1 + \mathbf{diff}\,u = \mathbf{diff}\,u - 1$, so that $\mathbf{diff}\,v + 1 = \mathbf{diff}\,u \leq 0$. Hence $\mathbf{diff}\,v \leq -1 \leq 0$.
- Suppose $v = x0y1$. We showed above that $\mathbf{diff}(x0y1) = 0$, and thus $\mathbf{diff}\,v = 0 \leq 0$.

Part (b)

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

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

We proceed by strong string induction. Suppose $w \in \{0,1\}^*$, and assume the inductive hypothesis: for all $x \in \{0,1\}^*$, 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 two cases to consider, depending upon whether w is empty or not. If $w = \%$, then $w = \% \in X$, by Rule (1) of X's definition. So, suppose w is nonempty. Then $w = ta$ for some $t \in \{0,1\}^*$ and $a \in \{0,1\}$.

Suppose, toward a contradiction, that $a = \mathsf{0}$. Because $w \in Y$, we have that $\mathbf{diff}\, t + -1 = \mathbf{diff}\, t + \mathbf{diff}\, \mathsf{0} = \mathbf{diff}\, t + \mathbf{diff}\, a = \mathbf{diff}(ta) = \mathbf{diff}\, w = 0$. But then $1 = \mathbf{diff}\, t \leq 0$, because t is a prefix of $ta = w$ and $w \in Y$—contradiction.

Thus $a = \mathsf{1}$, so that $w = t\mathsf{1}$. Since $\mathbf{diff}\, t + 1 = \mathbf{diff}\, t + \mathbf{diff}\, \mathsf{1} = \mathbf{diff}(t\mathsf{1}) = \mathbf{diff}\, w = 0$, we have that $\mathbf{diff}\, t = -1$. Let u be the shortest suffix of t such that $\mathbf{diff}\, u \leq -1$. ($u$ is well-defined, because t is a suffix of itself, and $\mathbf{diff}\, t = -1$.) Thus $t = xu$ for some $x \in \{\mathsf{0}, \mathsf{1}\}^*$. Because $\mathbf{diff}\, u \leq -1$, we have that $u \neq \%$, so that $u = by$ for some $b \in \{\mathsf{0}, \mathsf{1}\}$ and $y \in \{\mathsf{0}, \mathsf{1}\}^*$. Since $u = by$ and $t = xu = xby$, we have that y is a strictly shorter suffix of t than u. Hence, by the definition of u, it follows that $\mathbf{diff}\, y \geq 0$.

Suppose, toward a contradiction, that $b = \mathsf{1}$. Then $1 + \mathbf{diff}\, y = \mathbf{diff}\, \mathsf{1} + \mathbf{diff}\, y = \mathbf{diff}\, b + \mathbf{diff}\, y = \mathbf{diff}(by) = \mathbf{diff}\, u \leq -1$, so that $\mathbf{diff}\, y \leq -2$—contradiction.

Thus $b = \mathsf{0}$, so that $u = by = \mathsf{0}y$, $t = xu = x\mathsf{0}y$ and $w = t\mathsf{1} = x\mathsf{0}y\mathsf{1}$. Since $-1 + \mathbf{diff} y = \mathbf{diff}\, \mathsf{0} + \mathbf{diff} y = \mathbf{diff}(\mathsf{0}y) = \mathbf{diff} u \leq -1$, it follows that $\mathbf{diff}\, y \leq 0$. But $\mathbf{diff}\, y \geq 0$, and thus $\mathbf{diff}\, y = 0$. Since $\mathbf{diff}\, x = \mathbf{diff}\, x + -1 + 0 + 1 = \mathbf{diff}\, x + \mathbf{diff}\, \mathsf{0} + \mathbf{diff}\, y + \mathbf{diff}\, \mathsf{1} = \mathbf{diff}(x\mathsf{0}y\mathsf{1}) = \mathbf{diff}\, w = 0$, we have $\mathbf{diff}\, x = 0$.

To complete the proof that $x \in Y$, suppose v is a prefix of x. We must show that $\mathbf{diff}\, v \leq 0$. But v is a prefix of $x\mathsf{0}y\mathsf{1} = w$ and $w \in Y$, and thus $\mathbf{diff}\, v \leq 0$.

To complete the proof that $y \in Y$, suppose v is a prefix of y. We must show that $\mathbf{diff}\, v \leq 0$. We have that $y = vz$ for some $z \in \{\mathsf{0}, \mathsf{1}\}^*$. Because $u = \mathsf{0}y = \mathsf{0}vz$, we have that z is strictly shorter than u. Furthermore, z is a suffix of $x\mathsf{0}vz = x\mathsf{0}y = t$. But by its definition, we had that u was the shortest suffix of t such that $\mathbf{diff}\, u \leq -1$. Hence $\mathbf{diff}\, z \geq 0$. Since $\mathbf{diff}\, v + \mathbf{diff}\, z = \mathbf{diff}(vz) = \mathbf{diff}\, y = 0$, we have that $-(\mathbf{diff}\, v) = \mathbf{diff}\, z \geq 0$, so that $\mathbf{diff}\, v \leq 0$—as required.

Summarizing, we have that $w = x\mathsf{0}y\mathsf{1}$ and $x, y \in Y$. Clearly x and y are proper substrings of w, and thus the inductive hypothesis tells us that $x, y \in X$. Consequently, $w = x\mathsf{0}y\mathsf{1} \in X$, by Rule 2 of X's definition.

Problem 2

Here is `ps2-explain.sml`:

```
(* ps2-explain.sml

   Solution to Problem 2 of Problem Set 2 *)

(* val shortestPref : (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 shortestPref f w returns (x, y), where x
   is the shortest such prefix and y is such that x @ y = w *)

fun shortestPref 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 shortestSuff : (int -> bool) -> str -> str * str

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

fun shortestSuff f (w : str) : str * str =
      let val (x, y) = shortestPref f (rev w)
      in (rev x, rev y) end

(* val shortestNegSuff : str -> str * str

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

val shortestNegSuff = shortestSuff(fn n => n <= ~1)

(* 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 (* w ends with one *)
           let val t = Str.allButLast w
               (* w = t @ [one], diff t = ~1 *)
               val (u, x) = shortestNegSuff t
               (* t = x @ u, u begins with zero, diff u = ~1, all proper
                  suffixes of u have non-negative diffs, x is in Y *)
               val y = tl u
               (* w = x @ [zero] @ y @ [one], x, y are in Y *)
           in Rule2(explain x, explain y) 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
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 shortestPref = fn : (int -> bool) -> str -> str * str
val shortestSuff = fn : (int -> bool) -> str -> str * str
val shortestNegSuff = fn : str -> str * str
val explain = fn : str -> expl
val it = () : unit
- val doit = test explain;
val doit = fn : str -> unit
- doit(Str.fromString "%");
% is in X, by rule (1)
val it = () : unit
- doit(Str.fromString "01");
01 = % @ 0 @ % @ 1 is in X, by rule (2)
  % is in X, by rule (1)
  % is in X, by rule (1)
val it = () : unit
- doit(Str.fromString "0101");
0101 = 01 @ 0 @ % @ 1 is in X, by rule (2)
  01 = % @ 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 "010101");
010101 = 0101 @ 0 @ % @ 1 is in X, by rule (2)
  0101 = 01 @ 0 @ % @ 1 is in X, by rule (2)
    01 = % @ 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)
  % is in X, by rule (1)
val it = () : unit
- doit(Str.fromString "000111");
000111 = % @ 0 @ 0011 @ 1 is in X, by rule (2)
  % is in X, by rule (1)
  0011 = % @ 0 @ 01 @ 1 is in X, by rule (2)
    % is in X, by rule (1)
    01 = % @ 0 @ % @ 1 is in X, by rule (2)
      % is in X, by rule (1)
      % is in X, by rule (1)
```

```
val it = () : unit
- doit(Str.fromString "0010100111");
0010100111 = % @ 0 @ 01010011 @ 1 is in X, by rule (2)
  % is in X, by rule (1)
  01010011 = 0101 @ 0 @ 01 @ 1 is in X, by rule (2)
    0101 = 01 @ 0 @ % @ 1 is in X, by rule (2)
      01 = % @ 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)
    01 = % @ 0 @ % @ 1 is in X, by rule (2)
      % is in X, by rule (1)
      % is in X, by rule (1)
val it = () : unit
- doit(Str.fromString "00101100110010100111");
00101100110010100111 = 0010110011 @ 0 @ 01010011 @ 1 is in X, by rule (2)
  0010110011 = 001011 @ 0 @ 01 @ 1 is in X, by rule (2)
    001011 = % @ 0 @ 0101 @ 1 is in X, by rule (2)
      % is in X, by rule (1)
      0101 = 01 @ 0 @ % @ 1 is in X, by rule (2)
        01 = % @ 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)
    01 = % @ 0 @ % @ 1 is in X, by rule (2)
      % is in X, by rule (1)
      % is in X, by rule (1)
  01010011 = 0101 @ 0 @ 01 @ 1 is in X, by rule (2)
    0101 = 01 @ 0 @ % @ 1 is in X, by rule (2)
      01 = % @ 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)
    01 = % @ 0 @ % @ 1 is in X, by rule (2)
      % is in X, by rule (1)
      % is in X, by rule (1)
val it = () : unit
```