

Problem Set 3

Model Answers

Problem 1

To begin with, we put the following declarations in the file `ps3-prob1.sml`:

```

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

fun diff (nil : str) = 0
  | diff (b :: bs)   =
    if Sym.equal(b, zero)
    then ~1 + diff bs
    else 1 + diff bs;

fun equal n =
  Set.filter
    (fn x => diff x = 0)
    (StrSet.power(StrSet.fromString "0, 1", n));

fun upto 0 = equal 0
  | upto n = StrSet.union(equal n, upto(n - 1));

fun locSimp n = Reg.locallySimplify(SOME n, Reg.obviousSubset);

fun assess reg =
  (length(Reg.ccToList(Reg.cc reg)), Reg.size reg, Reg.numConcats reg,
   Reg.numSyms reg, Reg.standardized reg);

```

We then load this file into Forlan:

```

- use "ps3-prob1.sml";
[opening ps3-prob1.sml]
val zero = - : sym
val one  = - : sym
val diff = fn : str -> int
val equal = fn : int -> str set
val upto  = fn : int -> str set
val locSimp = fn : int -> reg -> bool * reg
val assess = fn : reg -> int * int * int * int * bool
val it = () : unit

```

Given a natural number n :

- `equal` returns $\{w \in \{0, 1\}^* \mid |w| = n \text{ and } \mathbf{diff} w = 0\}$; and

- `upto` returns $\{w \in \{0,1\}^* \mid |w| \leq n \text{ and } \mathbf{diff} w = 0\}$.

The function `locSimp` locally simplifies a regular expression using `Reg.obviousSubset` as the approximation to subset testing, and considering up to n structural reorganizations at each recursive call. And the function `assess` assesses the complexity of a regular expression. (Because our regular expressions won't contain closures, their closure complexities will be lists of zeroes.) Thus `upto 6` returns X , and we bind `xs` to X :

```
- val xs = upto 6;
val xs = - : str set
- StrSet.output("", xs);
%, 01, 10, 0011, 0101, 0110, 1001, 1010, 1100, 000111, 001011, 001101, 001110,
010011, 010101, 010110, 011001, 011010, 011100, 100011, 100101, 100110, 101001,
101010, 101100, 110001, 110010, 110100, 111000
val it = () : unit
```

To begin our first attempt at finding a simple regular expression generating X , we create a regular expression, `reg`, consisting of the union of all the elements of X :

```
- val reg = Reg.fromStrSet xs;
val reg = - : reg
- Reg.output("", reg);
% + 01 + 10 + 0011 + 0101 + 0110 + 1001 + 1010 + 1100 + 000111 + 001011 +
001101 + 001110 + 010011 + 010101 + 010110 + 011001 + 011010 + 011100 + 100011 +
100101 + 100110 + 101001 + 101010 + 101100 + 110001 + 110010 + 110100 + 111000
val it = () : unit
```

Then, we can try locally simplifying `reg` with increasing values of n : 10, 1000, 1500:

```
- val (b, reg10) = locSimp 10 reg;
val b = false : bool
val reg10 = - : reg
- assess reg10;
val it = (97,193,68,96,true) : int * int * int * int * bool
- Reg.output("", reg10);
% +
0
(1 + 0(0111 + 1(1 + 011 + 1(01 + 10))) +
  1(01 + 0(011 + 1(01 + 10)) + 1(0 + 0(01 + 10) + 100))) +
1
(0 + 001 + 00011 + 00101 + 00110 + 010 + 01001 + 01010 + 01100 +
  1(00 + 0001 + 0010 + 0100 + 1000))
val it = () : unit
- val (b, reg1000) = locSimp 1000 reg;
val b = false : bool
val reg1000 = - : reg
- assess reg1000;
val it = (77,153,48,68,true) : int * int * int * int * bool
- Reg.output("", reg1000);
% +
```

```

0
(0(0111 + 1(011 + 1(% + 01 + 10))) +
 1(% + 0(011 + 1(% + 01 + 10)) + 1(0(% + 01 + 10) + 100))) +
1
(0(% + 0(1(01 + 10) + (% + 01)1) + 1(0(% + 01 + 10) + 100)) +
 1(0(0(% + 01 + 10) + 100) + 1000))
val it = () : unit
- val (b, reg1500) = locSimp 1500 reg;
val b = false : bool
val reg1500 = - : reg
- assess reg1500;
val it = (77,153,48,68,true) : int * int * int * int * bool
- Reg.output("", reg1500);
% +
0
(0(0111 + 1(011 + 1(% + 01 + 10))) +
 1(% + 0(011 + 1(% + 01 + 10)) + 1(0(% + 01 + 10) + 100))) +
1
(0(% + 0(011 + 1(% + 01 + 10)) + 1(0(% + 01 + 10) + 100)) +
 1(0(0(% + 01 + 10) + 100) + 1000))
val it = () : unit

```

(It took about 18 minutes to carry out these simplifications on my laptop.) Note that `reg1000` and `reg1500` have the same complexity:

```

- Reg.compareComplexity(reg1000, reg1500);
val it = EQUAL : order

```

But `reg1500` is more symmetric than 1000, as can be seen by manually reordering its unions:

```

% +
0
(0(1(1(% + 01 + 10) + 011) + 0111) +
 1(% + 0(1(% + 01 + 10) + 011) + 1(0(% + 01 + 10) + 100))) +
1
(1(0(0(% + 10 + 01) + 100) + 1000) +
 0(% + 1(0(% + 10 + 01) + 100) + 0(1(% + 10 + 01) + 011)))

```

Although `reg1500` is nicely symmetric, it seemed unlikely to be optimally simple, so I tried several approaches to guiding Forlan to a better result. The approach that worked best is detailed below.

First, we bind `fours` to the result of evaluating `equal 4`, i.e., to $\{w \in \{0,1\}^* \mid |w| = 4 \text{ and } \text{diff } w = 0\}$:

```

- val fours = equal 4;
val fours = - : str set
- StrSet.output("", fours);
0011, 0101, 0110, 1001, 1010, 1100
val it = () : unit

```

Recall the elements of X :

```
- StrSet.output("", xs);
%, 01, 10, 0011, 0101, 0110, 1001, 1010, 1100, 000111, 001011, 001101, 001110,
010011, 010101, 010110, 011001, 011010, 011100, 100011, 100101, 100110, 101001,
101010, 101100, 110001, 110010, 110100, 111000
val it = () : unit
```

Because a majority of the elements of X end with one of the elements of `fours`, we will partition X into 8 sets: the elements of X ending in each of the 6 elements of `fours`, the elements of X with length no more than 2, and the length 6 elements of X that don't end with an element of `fours`:

```
- fun ends(x, ys) = Set.filter (fn y => Str.suffix(x, y)) ys
= val parts =
=     let val ps = Set.mapToList (fn y => ends(y, xs)) fours
=         val ws = StrSet.minus(xs, StrSet.genUnion ps)
=         val us = Set.filter (fn w => length w <= 2) ws
=         val vs = StrSet.minus(ws, us)
=     in vs :: us :: ps end;
val ends = fn : str * str set -> str set
val parts = [-,-,-,-,-,-,-] : str set list
- app (fn part => StrSet.output("", part)) parts;
000111, 001011, 001101, 001110, 110001, 110010, 110100, 111000
%, 01, 10
0011, 010011, 100011
0101, 010101, 100101
0110, 010110, 100110
1001, 011001, 101001
1010, 011010, 101010
1100, 011100, 101100
val it = () : unit
- StrSet.equal(StrSet.genUnion parts, xs);
val it = true : bool
```

So, the first element of `parts` consists of the length 6 elements of X that don't end with an element of `fours`, the next element is the elements of X with length no more than 2, and the remaining six elements are the elements of X ending in 0011, 0101, 0110, 1001, 1010 and 1100, respectively.

Next, we convert each element of `parts` into a regular expression that's the union of its elements, and simplify those regular expressions:

```
- val regs = map (fn ys => #2(locSimp 1000 (Reg.fromStrSet ys))) parts;
val regs = [-,-,-,-,-,-,-] : reg list
- app (fn reg => Reg.output("", reg)) regs;
00(11(01 + 10) + (01 + 10)11) + 11(00(01 + 10) + (01 + 10)00)
% + 01 + 10
(% + 01 + 10)0011
(% + 01 + 10)0101
(% + 01 + 10)0110
```

```

(% + 01 + 10)1001
(% + 01 + 10)1010
(% + 01 + 10)1100
val it = () : unit

```

Because all but the first of our regular expressions have a common subtree, we simplify the result of unioning those regular expressions together, resulting in `reg'`:

```

- val reg' = #2(locSimp 1000 (Reg.genUnion(tl regs)));
val reg' = - : reg
- Reg.output("", reg');
(% + 01 + 10)(% + 0(011 + 1(01 + 10)) + 1(001 + (01 + 10)0))
val it = () : unit

```

Finally, we simplify the union the first element of `regs` and `reg'`, calling the result `reg''`:

```

- val reg'' = #2(locSimp 1000 (Reg.union(hd regs, reg')));
val reg'' = - : reg
- assess reg'';
val it = (52,103,35,50,true) : int * int * int * int * bool
- Reg.output("", reg'');
00(11(01 + 10) + (01 + 10)11) + 11(00(01 + 10) + (01 + 10)00) +
(% + 01 + 10)(% + 0(011 + 1(01 + 10)) + 1(001 + (01 + 10)0))
val it = () : unit

```

Note that `reg''` is correct by construction, but we can also directly verify its correctness:

```

- StrSet.equal(Reg.toStrSet reg'', xs);
val it = true : bool

```

Problem 2

First, we define a function `locSimpTr` for locally simplifying a regular expression, with tracing turned on, using `Reg.obviousSubset` as the approximation to subset testing, and considering up to n structural reorganizations at each recursive call.

```

- fun locSimpTr n =
=       Reg.locallySimplifyTrace(SOME n, Reg.obviousSubset);
val locSimpTr = fn : int -> reg -> bool * reg

```

Then we use this function to illustrate how reduction rule (20) works:

```

- locSimpTr 100 (Reg.fromString "(11 + 111 + 11111 + 111111111)*");
exploration of structural reorganizations of (11 + 111 + 11111 + 111111111)*
curtailed
(11 + 111 + 11111 + 111111111)* transformed by reduction rule 20 at position []
to % + (11)1* weakly simplifies to % + 111*
considered all 12 structural reorganizations of % + 111*
% + 111* is locally simplified
val it = (true,-) : bool * reg
- locSimpTr 100

```

```

= (Reg.fromString "(111 + 1111 + 11111 + 1111111 + 111111111)*");
exploration of structural reorganizations of
(111 + 1111 + 11111 + 1111111 + 111111111)* curtailed
(111 + 1111 + 11111 + 1111111 + 111111111)* transformed by reduction rule 20 at
position [] to % + (111)1* weakly simplifies to % + 1111*
considered all 40 structural reorganizations of % + 1111*
% + 1111* is locally simplified
val it = (true,-) : bool * reg
- locSimpTr 100
= (Reg.fromString
= "(1111 + 11111 + 111111 + 1111111 + 111111111 + 11111111111)*");
exploration of structural reorganizations of
(1111 + 11111 + 111111 + 1111111 + 111111111 + 11111111111)* curtailed
(1111 + 11111 + 111111 + 1111111 + 111111111 + 11111111111)* transformed by
reduction rule 20 at position [] to % + (1111)1* weakly simplifies to % + 11111*
exploration of structural reorganizations of % + 11111* curtailed
% + 11111* may not be locally simplified
val it = (false,-) : bool * reg
- locSimpTr 100
= (Reg.fromString
= ("((0+1)(0+1)(0+1) + (0+1)(0+1)(0+1)(0+1) + (0+1)(0+1)(0+1)(0+1)(0+1))*");
((0 + 1)(0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1)(0 + 1) +
(0 + 1)(0 + 1)(0 + 1)(0 + 1)(0 + 1))*
weakly simplifies to
((0 + 1)(0 + 1)(0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1)(0 + 1) +
(0 + 1)(0 + 1)(0 + 1))*
exploration of structural reorganizations of
((0 + 1)(0 + 1)(0 + 1)(0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1)(0 + 1) +
(0 + 1)(0 + 1)(0 + 1))*
curtailed
((0 + 1)(0 + 1)(0 + 1)(0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1)(0 + 1) +
(0 + 1)(0 + 1)(0 + 1))*
transformed by structural rule 2 at position [1] to
(((0 + 1)(0 + 1)(0 + 1)(0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1)(0 + 1)) +
(0 + 1)(0 + 1)(0 + 1))*
transformed by structural rule 5 at position [1, 1] to
(((0 + 1)(0 + 1)(0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1)(0 + 1)(0 + 1)) +
(0 + 1)(0 + 1)(0 + 1))*
transformed by reduction rule 22 at position [1, 1] to
((% + 0 + 1)(0 + 1)(0 + 1)(0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1))*
exploration of structural reorganizations of
((% + 0 + 1)(0 + 1)(0 + 1)(0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1))* curtailed
((% + 0 + 1)(0 + 1)(0 + 1)(0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1))* transformed
by structural rule 5 at position [1] to
((0 + 1)(0 + 1)(0 + 1) + (% + 0 + 1)(0 + 1)(0 + 1)(0 + 1))* transformed
by structural rule 4 at position [1, 2] to
((0 + 1)(0 + 1)(0 + 1) + ((% + 0 + 1)(0 + 1))(0 + 1)(0 + 1)(0 + 1))* transformed
by reduction rule 22 at position [1] to

```

```

((% + (% + 0 + 1)(0 + 1))(0 + 1)(0 + 1)(0 + 1))*
exploration of structural reorganizations of
((% + (% + 0 + 1)(0 + 1))(0 + 1)(0 + 1)(0 + 1))* curtailed
((% + (% + 0 + 1)(0 + 1))(0 + 1)(0 + 1)(0 + 1))* may not be locally simplified
val it = (false,-) : bool * reg
- locSimpTr 1000
= (Reg.fromString
= ("((0+1)(0+1)(0+1)(0+1) + (0+1)(0+1)(0+1)(0+1)(0+1) + (0+1)(0+1)(0+1))*");
((0 + 1)(0 + 1)(0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1)(0 + 1) +
(0 + 1)(0 + 1)(0 + 1))*
weakly simplifies to
((0 + 1)(0 + 1)(0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1)(0 + 1) +
(0 + 1)(0 + 1)(0 + 1))*
exploration of structural reorganizations of
((0 + 1)(0 + 1)(0 + 1)(0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1)(0 + 1) +
(0 + 1)(0 + 1)(0 + 1))*
curtailed
((0 + 1)(0 + 1)(0 + 1)(0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1)(0 + 1) +
(0 + 1)(0 + 1)(0 + 1))*
transformed by structural rule 2 at position [1] to
(((0 + 1)(0 + 1)(0 + 1)(0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1)(0 + 1)) +
(0 + 1)(0 + 1)(0 + 1))*
transformed by structural rule 5 at position [1] to
((0 + 1)(0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1)(0 + 1)(0 + 1) +
(0 + 1)(0 + 1)(0 + 1)(0 + 1))*
transformed by structural rule 5 at position [1, 2] to
((0 + 1)(0 + 1)(0 + 1) + (0 + 1)(0 + 1)(0 + 1)(0 + 1) +
(0 + 1)(0 + 1)(0 + 1)(0 + 1)(0 + 1))*
transformed by reduction rule 20 at position [] to
% + ((0 + 1)(0 + 1)(0 + 1)(0 + 1))* weakly simplifies to
% + (0 + 1)(0 + 1)(0 + 1)(0 + 1)*
considered all 640 structural reorganizations of
% + (0 + 1)(0 + 1)(0 + 1)(0 + 1)*
% + (0 + 1)(0 + 1)(0 + 1)(0 + 1)* is locally simplified
val it = (true,-) : bool * reg

```

The last two examples show how a large number of structural reorganizations must sometimes be considered before one to which rule (20) applies is found.

Problem 3

(a)

Our α is

$$(0(01)^*1 + 1(10)^*0)^*(\% + 0(01)^*(\% + 0) + 1(10)^*(\% + 1)).$$

(b)

Let

$$\begin{aligned} A_0 &= \{0\}\{01\}^*, \\ A_1 &= \{1\}\{10\}^*, \text{ and} \\ B &= (A_0\{1\} \cup A_1\{0\})^* (\{\% \} \cup A_0\{\%, 0\} \cup A_1\{\%, 1\}). \end{aligned}$$

Then $L(\alpha) = B$, so it will suffice to show $B = Y$. We show that $B \subseteq Y \subseteq B$.

For $l, m, n \in \mathbb{Z}$ such that $l \leq 0$, $m \geq 0$ and $l \leq n \leq m$, define:

$$\begin{aligned} Y^{l,m} &= \{w \in \{0, 1\}^* \mid \text{for all prefixes } v \text{ of } w, l \leq \mathbf{diff} v \leq m\}, \text{ and} \\ Y_n^{l,m} &= \{w \in \{0, 1\}^* \mid w \in Y^{l,m} \text{ and } \mathbf{diff} w = n\} \end{aligned}$$

Thus:

- for all $l, m, n \in \mathbb{Z}$ such that $l \leq 0$, $m \geq 0$ and $l \leq n \leq m$, $Y_n^{l,m} \subseteq Y^{l,m}$;
- for all $l, l', m, m' \in \mathbb{Z}$, if $l' \leq l \leq 0$ and $0 \leq m \leq m'$, then $Y^{l,m} \subseteq Y^{l',m'}$; and
- for all $l, l', m, m', n \in \mathbb{Z}$, if $l' \leq l \leq 0$, $0 \leq m \leq m'$ and $l \leq n \leq m$, then $Y_n^{l,m} \subseteq Y_n^{l',m'}$.

Lemma PS3.3.1

(1) $\% \in Y_0^{0,0}$.

(2) $0 \in Y_{-1}^{-1,0}$.

(3) $1 \in Y_1^{0,1}$.

(4) For all $l, l', m, m' \in \mathbb{Z}$, if $l, l' \leq 0$ and $m, m' \geq 0$ then

$$Y^{l,m} \cup Y^{l',m'} \subseteq Y^{\min(l,l'), \max(m,m')}.$$

(5) For all $l, l', m, m', n \in \mathbb{Z}$, if $l, l' \leq 0$, $m, m' \geq 0$, $l \leq n \leq m$ and $l' \leq n \leq m'$, then

$$Y_n^{l,m} \cup Y_n^{l',m'} \subseteq Y_n^{\min(l,l'), \max(m,m')}.$$

(6) For all $l, l', m, m', n \in \mathbb{Z}$, if $l, l' \leq 0$, $m, m' \geq 0$ and $l \leq n \leq m$, then

$$Y_n^{l,m} Y^{l',m'} \subseteq Y^{\min(l, n+l'), \max(m, n+m')}.$$

(7) For all $l, l', m, m', n, n' \in \mathbb{Z}$, if $l, l' \leq 0$, $m, m' \geq 0$, $l \leq n \leq m$ and $l' \leq n' \leq m'$, then

$$Y_n^{l,m} Y_{n'}^{l',m'} \subseteq Y_{n+n'}^{\min(l, n+l'), \max(m, n+m')}.$$

(8) For all $l, m \in \mathbb{Z}$, if $l \leq 0$ and $m \geq 0$, then $(Y_0^{l,m})^* \subseteq Y_0^{l,m}$.

Proof.

- (1) Follows since $\mathbf{diff} \% = 0$, and $\%$ is the only prefix of itself.
- (2) Follows since $\mathbf{diff} \% = 0$, $\mathbf{diff} 0 = -1$ and the only prefixes of 0 are $\%$ and 0 .
- (3) Follows since $\mathbf{diff} \% = 0$, $\mathbf{diff} 1 = 1$ and the only prefixes of 1 are $\%$ and 1 .
- (4) Suppose $w \in Y^{l,m} \cup Y^{l',m'}$. There are two cases to consider.
 - Suppose $w \in Y^{l,m}$. To see that $w \in Y^{\mathbf{min}(l,l'), \mathbf{max}(m,m')}$, suppose v is a prefix of w . Then $\mathbf{min}(l, l') \leq l \leq \mathbf{diff} v \leq m \leq \mathbf{max}(m, m')$.
 - Suppose $w \in Y^{l',m'}$. The proof is similar to the other case.
- (5) Follows immediately from part (4).
- (6) Suppose $w \in Y_n^{l,m} Y_{n'}^{l',m'}$, so that $w = xy$ for some $x \in Y_n^{l,m}$ and $y \in Y_{n'}^{l',m'}$. To see that $w \in Y^{\mathbf{min}(l, n+l'), \mathbf{max}(m, n+m')}$, suppose v is a prefix of w . There are two cases to consider.
 - Suppose v is a prefix of x . Then $\mathbf{min}(l, n+l') \leq l \leq \mathbf{diff} v \leq m \leq \mathbf{max}(m, n+m')$.
 - Suppose $v = xu$ for a prefix u of y . Hence $l' \leq \mathbf{diff} u \leq m'$, so that $\mathbf{min}(l, n+l') \leq n+l' \leq n+\mathbf{diff} u \leq n+m' \leq \mathbf{max}(m, n+m')$. But $\mathbf{diff} v = \mathbf{diff}(xu) = \mathbf{diff} x + \mathbf{diff} u = n + \mathbf{diff} u$, so that $\mathbf{min}(l, n+l') \leq \mathbf{diff} v \leq \mathbf{max}(m, n+m')$.
- (7) Suppose $w \in Y_n^{l,m} Y_{n'}^{l',m'}$, so that $w = xy$ for some $x \in Y_n^{l,m}$ and $y \in Y_{n'}^{l',m'}$. Thus $\mathbf{diff} w = \mathbf{diff}(xy) = \mathbf{diff} x + \mathbf{diff} y = n + n'$. And the rest follows by part (6).
- (8) We use mathematical induction to show that, for all $n \in \mathbb{N}$, $(Y_0^{l,m})^n \subseteq Y_0^{l,m}$.

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

(Inductive Step) Suppose $n \in \mathbb{N}$, and assume the inductive hypothesis: $(Y_0^{l,m})^n \subseteq Y_0^{l,m}$. Then $(Y_0^{l,m})^{n+1} = Y_0^{l,m} (Y_0^{l,m})^n \subseteq Y_0^{l,m} Y_0^{l,m} \subseteq Y_0^{\mathbf{min}(l, 0+l), \mathbf{max}(m, 0+m)} = Y_0^{l,m}$, by the inductive hypothesis and part (7).

Now, suppose $w \in (Y_0^{l,m})^*$. Then $w \in (Y_0^{l,m})^n$, for some $n \in \mathbb{N}$. Hence $w \in (Y_0^{l,m})^n \subseteq Y_0^{l,m}$.

□

Lemma PS3.3.2

- (1) $\{01\}^* \subseteq Y_0^{-1,0}$.
- (2) $\{10\}^* \subseteq Y_0^{0,1}$.
- (3) $A_0 \subseteq Y_{-1}^{-2,0}$.
- (4) $A_1 \subseteq Y_1^{0,2}$.
- (5) $A_0\{1\} \subseteq Y_0^{-2,0}$.
- (6) $A_1\{0\} \subseteq Y_0^{0,2}$.
- (7) $A_0\{1\} \cup A_1\{0\} \subseteq Y_0^{-2,2}$.

- (8) $(A_0\{1\} \cup A_1\{0\})^* \subseteq Y_0^{-2,2}$.
- (9) $\{\%, 0\} \subseteq Y^{-1,0}$.
- (10) $\{\%, 1\} \subseteq Y^{0,1}$.
- (11) $A_0\{\%, 0\} \subseteq Y^{-2,0}$.
- (12) $A_1\{\%, 1\} \subseteq Y^{0,2}$.
- (13) $\{\%\} \cup A_0\{\%, 0\} \cup A_1\{\%, 1\} \subseteq Y^{-2,2}$.
- (14) $B \subseteq Y^{-2,2}$.

We use Lemma PS3.3.1 repeatedly, without reference, in the following proof.

Proof.

- (1) We have that $\{01\} = \{0\}\{1\} \subseteq Y_{-1}^{-1,0}Y_1^{0,1} \subseteq Y_{-1+1}^{\min(-1,-1+0),\max(0,-1+1)} = Y_0^{-1,0}$. Thus $\{01\}^* \subseteq (Y_0^{-1,0})^* \subseteq Y_0^{-1,0}$.
- (2) We have that $\{10\} = \{1\}\{0\} \subseteq Y_1^{0,1}Y_{-1}^{-1,0} \subseteq Y_{1+-1}^{\min(0,1+-1),\max(1,1+0)} = Y_0^{0,1}$. Thus $\{10\}^* \subseteq (Y_0^{0,1})^* \subseteq Y_0^{0,1}$.
- (3) Since $\{0\} \subseteq Y_{-1}^{-1,0}$, we have that $A_0 = \{0\}\{01\}^* \subseteq Y_{-1}^{-1,0}Y_0^{-1,0} \subseteq Y_{-1+0}^{\min(-1,-1+-1),\max(0,-1+0)} = Y_{-1}^{-2,0}$, by part (1).
- (4) Since $\{1\} \subseteq Y_1^{0,1}$, we have that $A_1 = \{1\}\{10\}^* \subseteq Y_1^{0,1}Y_0^{0,1} \subseteq Y_{1+0}^{\min(0,1+0),\max(1,1+1)} = Y_1^{0,2}$, by part (2).
- (5) $A_0\{1\} \subseteq Y_{-1}^{-2,0}Y_1^{0,1} \subseteq Y_{-1+1}^{\min(-2,-1+0),\max(0,-1+1)} = Y_0^{-2,0}$, by part (3).
- (6) $A_1\{0\} \subseteq Y_1^{0,2}Y_{-1}^{-1,0} \subseteq Y_{1+-1}^{\min(0,1+-1),\max(2,1+0)} = Y_0^{0,2}$, by part (4).
- (7) $A_0\{1\} \cup A_1\{0\} \subseteq Y_0^{-2,0} \cup Y_0^{0,2} \subseteq Y_0^{\min(-2,0),\max(0,2)} = Y_0^{-2,2}$, by parts (5) and (6).
- (8) Since $A_0\{1\} \cup A_1\{0\} \subseteq Y_0^{-2,2}$, by part (7), we have that $(A_0\{1\} \cup A_1\{0\})^* \subseteq (Y_0^{-2,2})^* \subseteq Y_0^{-2,2}$.
- (9) $\{\%, 0\} = \{\%\} \cup \{0\} \subseteq Y^{0,0} \cup Y^{-1,0} \subseteq Y^{\min(0,-1),\max(0,0)} = Y^{-1,0}$.
- (10) $\{\%, 1\} = \{\%\} \cup \{1\} \subseteq Y^{0,0} \cup Y^{0,1} \subseteq Y^{\min(0,0),\max(0,1)} = Y^{0,1}$.
- (11) $A_0\{\%, 0\} \subseteq Y_{-1}^{-2,0}Y^{-1,0} \subseteq Y^{\min(-2,-1+-1),\max(0,-1+0)} = Y^{-2,0}$, by parts (3) and (9).
- (12) $A_1\{\%, 1\} \subseteq Y_1^{0,2}Y^{0,1} \subseteq Y^{\min(0,1+0),\max(2,1+1)} = Y^{0,2}$, by parts (4) and (10).
- (13) $\{\%\} \cup A_0\{\%, 0\} \cup A_1\{\%, 1\} \subseteq Y^{0,0} \cup Y^{-2,0} \cup Y^{0,2} \subseteq Y^{\min(0,-2),\max(0,0)} \cup Y^{0,2} = Y^{-2,0} \cup Y^{0,2} \subseteq Y^{\min(-2,0),\max(0,2)} = Y^{-2,2}$, by parts (11) and (12).
- (14) $B = (A_0\{1\} \cup A_1\{0\})^* (\{\%\} \cup A_0\{\%, 0\} \cup A_1\{\%, 1\}) \subseteq Y_0^{-2,2} Y^{-2,2} \subseteq Y^{\min(-2,0+-2),\max(2,0+2)} = Y^{-2,2}$, by parts (8) and (13).

□

By Lemma PS3.3.2(14), we have that $B \subseteq Y^{-2,2} = Y$. So, it remains to show that $Y \subseteq B$.

Lemma PS3.3.3

For all $x, y \in \{0, 1\}^*$, if $xy \in Y$ and $\mathbf{diff} x = 0$, then $y \in Y$.

Proof. Suppose $x, y \in \{0, 1\}^*$, $xy \in Y$ and $\mathbf{diff} x = 0$. To show that $y \in Y$, suppose v is a prefix of y . Hence xv is a prefix of xy , so that $-2 \leq \mathbf{diff}(xv) \leq 2$. But $\mathbf{diff}(xv) = \mathbf{diff} x + \mathbf{diff} v = 0 + \mathbf{diff} v = \mathbf{diff} v$, so that $-2 \leq \mathbf{diff} v \leq 2$, as required. \square

Lemma PS3.3.4

$Y \subseteq B$.

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

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

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,

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

We must show that,

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

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

- Suppose $w = \%$. Then

$$w = \% = \% \% \in (A_0\{1\} \cup A_1\{0\})^* (\{\%\} \cup A_0\{\%, 0\} \cup A_1\{\%, 1\}) = B.$$

- Suppose $w = 0x$, for some $x \in \{0, 1\}^*$. Let y be the longest element of $\{01\}^*$ such that y is a prefix of x (y is well-defined, because it could be $\%$), and $z \in \{0, 1\}^*$ be such that $x = yz$. Thus $w = 0x = 0yz$ and $0y \in \{0\}\{01\}^* = A_0$. There are three subcases to consider.

- Suppose $z = \%$. Then

$$\begin{aligned} w = 0yz = 0y\% = 0y = \% (0y)\% &\in (A_0\{1\} \cup A_1\{0\})^* A_0\{\%, 0\} \\ &\subseteq (A_0\{1\} \cup A_1\{0\})^* (\{\%\} \cup A_0\{\%, 0\} \cup A_1\{\%, 1\}) = B. \end{aligned}$$

- Suppose $z = 0u$, for some $u \in \{0, 1\}^*$. Thus $x = yz = y0u$ and $w = 0x = 0y0u$.

Suppose, toward a contradiction, that $u \neq \%$. There are two cases to consider.

- * Suppose $u = 0v$, for some $v \in \{0, 1\}^*$. Then $w = 0y0u = 0y00v$. By Lemma PS3.3.2(1), we have that $y \in \{01\}^* \subseteq Y_0^{-1,0}$, so that $\mathbf{diff} y = 0$. Hence $\mathbf{diff}(0y00) = \mathbf{diff} 0 + \mathbf{diff} y + \mathbf{diff} 0 + \mathbf{diff} 0 = -1 + 0 + -1 + -1 = -3$. But $0y00$ is a prefix of $w \in Y$ —contradiction.
- * Suppose $u = 1v$, for some $v \in \{0, 1\}^*$. Then $x = y0u = y01v$. Since $y \in \{01\}^*$, it follows that $y01 \in \{01\}^*\{01\} \subseteq \{01\}^*$. But $y01$ is a longer prefix of x than y , contradicting the definition of y .

Since we obtained a contradiction in both cases, we have an overall contradiction. Thus $u = \%$.

Since $u = \%$, we have that

$$\begin{aligned} w &= 0y0u = 0y0\% = 0y0 = \%(0y)0 \in (A_0\{1\} \cup A_1\{0\})^* A_0\{\%, 0\} \\ &\subseteq (A_0\{1\} \cup A_1\{0\})^* (\{\%\} \cup A_0\{\%, 0\} \cup A_1\{\%, 1\}) = B. \end{aligned}$$

- Suppose $z = 1u$, for some $u \in \{0, 1\}^*$. Thus $w = 0yz = 0y1u$. By Lemma PS3.3.2(5), $0y1 \in A_0\{1\} \subseteq Y_0^{-2,0}$, so that $\mathbf{diff}(0y1) = 0$. Thus, since $0y1u = w \in Y$, Lemma PS3.3.3 tells us that $u \in Y$. Since u is a proper substring of w , the inductive hypothesis tells us that $u \in B$. Hence

$$\begin{aligned} w &= 0y1u \in A_0\{1\}B \subseteq (A_0\{1\} \cup A_1\{0\})B \subseteq (A_0\{1\} \cup A_1\{0\})^* B \\ &= (A_0\{1\} \cup A_1\{0\})^* (A_0\{1\} \cup A_1\{0\})^* (\{\%\} \cup A_0\{\%, 0\} \cup A_1\{\%, 1\}) \\ &= (A_0\{1\} \cup A_1\{0\})^* (\{\%\} \cup A_0\{\%, 0\} \cup A_1\{\%, 1\}) = B. \end{aligned}$$

- Suppose $w = 1x$, for some $x \in \{0, 1\}^*$. Let y be the longest element of $\{10\}^*$ such that y is a prefix of x (y is well-defined, because it could be $\%$), and $z \in \{0, 1\}^*$ be such that $x = yz$. Thus $w = 1x = 1yz$ and $1y \in \{1\}\{10\}^* = A_1$. There are three subcases to consider.

- Suppose $z = \%$. Then

$$\begin{aligned} w &= 1yz = 1y\% = 1y = \%(1y)\% \in (A_0\{1\} \cup A_1\{0\})^* A_1\{\%, 1\} \\ &\subseteq (A_0\{1\} \cup A_1\{0\})^* (\{\%\} \cup A_0\{\%, 0\} \cup A_1\{\%, 1\}) = B. \end{aligned}$$

- Suppose $z = 1u$, for some $u \in \{0, 1\}^*$. Thus $x = yz = y1u$ and $w = 1x = 1y1u$.

Suppose, toward a contradiction, that $u \neq \%$. There are two cases to consider.

- * Suppose $u = 1v$, for some $v \in \{0, 1\}^*$. Then $w = 1y1u = 1y11v$. By Lemma PS3.3.2(2), we have that $y \in \{10\}^* \subseteq Y_0^{0,1}$, so that $\mathbf{diff} y = 0$. Hence $\mathbf{diff}(1y11) = \mathbf{diff} 1 + \mathbf{diff} y + \mathbf{diff} 1 + \mathbf{diff} 1 = 1 + 0 + 1 + 1 = 3$. But $1y11$ is a prefix of $w \in Y$ —contradiction.
- * Suppose $u = 0v$, for some $v \in \{0, 1\}^*$. Then $x = y1u = y10v$. Since $y \in \{10\}^*$, it follows that $y10 \in \{10\}^*\{10\} \subseteq \{10\}^*$. But $y10$ is a longer prefix of x than y , contradicting the definition of y .

Since we obtained a contradiction in both cases, we have an overall contradiction. Thus $u = \%$.

Since $u = \%$, we have that

$$\begin{aligned} w &= 1y1u = 1y1\% = 1y1 = \%(1y)1 \in (A_0\{1\} \cup A_1\{0\})^* A_1\{\%, 1\} \\ &\subseteq (A_0\{1\} \cup A_1\{0\})^* (\{\%\} \cup A_0\{\%, 0\} \cup A_1\{\%, 1\}) = B. \end{aligned}$$

- Suppose $z = 0u$, for some $u \in \{0, 1\}^*$. Thus $w = 1yz = 1y0u$. By Lemma PS3.3.2(6), $1y0 \in A_1\{0\} \subseteq Y_0^{0,2}$, so that $\mathbf{diff}(1y0) = 0$. Thus, since $1y0u = w \in Y$, Lemma PS3.3.3

tells us that $u \in Y$. Since u is a proper substring of w , the inductive hypothesis tells us that $u \in B$. Hence

$$\begin{aligned}
 w &= 1y0u \in A_1\{0\}B \subseteq (A_0\{1\} \cup A_1\{0\})B \subseteq (A_0\{1\} \cup A_1\{0\})^*B \\
 &= (A_0\{1\} \cup A_1\{0\})^*(A_0\{1\} \cup A_1\{0\})^*({\%} \cup A_0\{%, 0\} \cup A_1\{%, 1\}) \\
 &= (A_0\{1\} \cup A_1\{0\})^*({\%} \cup A_0\{%, 0\} \cup A_1\{%, 1\}) = B.
 \end{aligned}$$

□