CS 516—Software Foundations via Formal Languages—Spring 2022

Problem Set 3

Model Answers

Problem 1

First we define the functions `locTr` and `globTr` for carrying out local and global simplification of a regular expression parsed from an SML string, with tracing turned on, and with no limit on the number of structural reorganizations considered, in the case of local simplification, and with no limit on the number of candidates considered, in the case of global simplification:

```
- fun locTr s =
=       Reg.locallySimplifyTrace (NONE, Reg.obviousSubset) (Reg.fromString s);
val locTr = fn : string -> bool * reg
- fun globTr s =
=       Reg.globallySimplifyTrace (NONE, Reg.obviousSubset) (Reg.fromString s);
val globTr = fn : string -> bool * reg
```

Next we define: a function `globPr` for carrying out the global simplification of a regular expression in string form, and then printing out the result; a function `weakSimpPr` for weakly simplifying a regular expression in string form, and printing out the result; a function `cmpCompl` for comparing the complexity of two regular expressions in string form; and a function `cmpTot` for comparing two regular expressions in string form in their total ordering:

```
- fun globPr s =
=       Reg.output
=       ("",
=        #2(Reg.globallySimplify (NONE, Reg.obviousSubset) (Reg.fromString s)));
val globPr = fn : string -> unit
- fun weakSimpPr s =
=       Reg.output("", Reg.weaklySimplify(Reg.fromString s));
val weakSimpPr = fn : string -> unit
- fun cmpCompl(s1, s2) =
=       Reg.compareComplexity(Reg.fromString s1, Reg.fromString s2);
val cmpCompl = fn : string * string -> order
- fun cmpTot(s1, s2) =
=       Reg.compare(Reg.fromString s1, Reg.fromString s2);
val cmpTot = fn : string * string -> order
```

Next, we apply `locTr` and `globPr` to $(00^*11^*)^*$:

```
- locTr "(00*11*)*";
considered all 20 structural reorganizations of (00*11*)*
(00*11*)* transformed by structural rule 4 at position [1] to ((00*)11*)*
transformed by structural rule 4 at position [1] to (((00*)1)1*)* transformed by
reduction rule 14 at position [] to % + ((00*)1)((00*)1 + 1)* weakly simplifies
```

```
to % + 00*1(1 + 00*1)*
considered all 160 structural reorganizations of % + 00*1(1 + 00*1)*
% + 00*1(1 + 00*1)* transformed by structural rule 4 at position
[2, 2, 2, 2, 1, 2] to % + 00*1(1 + (00*)1)* transformed by reduction rule 22 at
position [2, 2, 2, 2, 1] to % + 00*1((% + 00*)1)*
considered all 160 structural reorganizations of % + 00*1((% + 00*)1)*
% + 00*1((% + 00*)1)* transformed by reduction rule 25 at position
[2, 2, 2, 2, 1, 1] to % + 00*1(0*1)*
considered all 50 structural reorganizations of % + 00*1(0*1)*
% + 00*1(0*1)* transformed by reduction rule 15 at position [2, 2, 2, 2] to
% + 00*1(% + (0 + 1)*1)
considered all 80 structural reorganizations of % + 00*1(% + (0 + 1)*1)
% + 00*1(% + (0 + 1)*1) is locally simplified
val it = (true,-) : bool * reg
- globPr "(00*11*)*";
% + 0(0 + 1)*1
val it = () : unit
```

Thus the result, α, of global simplification is $\% + 0(0+1)^*1$, whereas the result of local simplification is $\% + 00^*1(\% + (0+1)^*1)$.

At the end of the first stage of local simplification, we have the intermediate result $\% + 00^*1(1 + 00^*1)^*$. And this globally simplifies to α:

```
- globPr "% + 00*1(1 + 00*1)*";
% + 0(0 + 1)*1
val it = () : unit
```

At the end of the second stage of local simplification, we have the intermediate result $\% + 00^*1((\% + 00^*)1)^*$. And this globally simplifies to α:

```
- globPr "% + 00*1((% + 00*)1)*";
% + 0(0 + 1)*1
val it = () : unit
```

At the end of the third stage of local simplification, we have the intermediate result $\% + 00^*1(0^*1)^*$. And this globally simplifies to α:

```
- globTr "% + 00*1(0*1)*";
considering candidates with explanations of length 0
simplest result now: % + 00*1(0*1)*
considering candidates with explanations of length 1
simplest result now: % + 00*1(0*1)* transformed by reduction rule 15 at position
[2, 2, 2, 2] to % + 00*1(% + (0 + 1)*1)
considering candidates with explanations of length 2
considering candidates with explanations of length 3
considering candidates with explanations of length 4
considering candidates with explanations of length 5
simplest result now: % + 00*1(0*1)* transformed by structural rule 8 at position
[2, 2, 2] to % + 00*(10*)*1 transformed by structural rule 4 at position [2, 2]
to % + 0(0*(10*)*)1 transformed by reduction rule 14 at position [2, 2, 1, 2] to
```

```
% + 0(0*(% + 1(1 + 0)*))1 transformed by weak simplification at position
[2, 2, 1] to % + 0(0*(% + 1(0 + 1)*))1 transformed by reduction rule 23 at
position [2, 2, 1] to % + 0(0 + 1)*1
considering candidates with explanations of length 6
considering candidates with explanations of length 7
considering candidates with explanations of length 8
considering candidates with explanations of length 9
search completed after considering 258 candidates with maximum size 17
% + 00*1(0*1)* transformed by structural rule 8 at position [2, 2, 2] to
% + 00*(10*)*1 transformed by structural rule 4 at position [2, 2] to
% + 0(0*(10*)*)1 transformed by reduction rule 14 at position [2, 2, 1, 2] to
% + 0(0*(% + 1(1 + 0)*))1 transformed by weak simplification at position
[2, 2, 1] to % + 0(0*(% + 1(0 + 1)*))1 transformed by reduction rule 23 at
position [2, 2, 1] to % + 0(0 + 1)*1 is globally simplified
val it = (true,-) : bool * reg
```

From this trace we can see that global simplification initially finds a way of using rule (15), resulting in $\% + 00^*1(\% + (0+1)^*1)$. This is what local simplification does. But then it finds that using rule (14) (after some structural reorganization) takes us to $\% + 0(0^*(\% + 1(1+0)^*))1$, which weakly simplifies at position $[2, 2, 1]$ to $\% + 0(0^*(\% + 1(0+1)^*))1$. This is a non-optimal *local* result, as shown by:

```
- cmpCompl("% + 0(0*(% + 1(0 + 1)*))1", "% + 00*1(% + (0 + 1)*1)");
val it = GREATER : order
```

But an application of rule (23) transforms this regular expression to the better final result, α. When local simplification considered using rule (14), it followed it with weak simplification of the *whole* expression:

```
- weakSimpPr "% + 0(0*(% + 1(1 + 0)*))1";
% + 00*(% + 1(0 + 1)*)1
val it = () : unit
```

Interestingly, this regular expression has the *same* complexity as the result of rule (15), but it is *larger* in our total ordering on regular expressions:

```
- cmpCompl("% + 00*(% + 1(0 + 1)*)1", "% + 00*1(% + (0 + 1)*1)");
val it = EQUAL : order
- cmpTot("% + 00*(% + 1(0 + 1)*)1", "% + 00*1(% + (0 + 1)*1)");
val it = GREATER : order
```

So $\% + 00^*(\% + 1(0+1)^*)1$ was rejected as the stage's result not because it had greater complexity but because the total ordering on regular expressions was used to break ties when judging local optimality. We can check that if local simplification had selected $\% + 00^*(\% + 1(0+1)^*)1$ at this state, it would have terminated with α:

```
- locTr "% + 00*(% + 1(0 + 1)*)1";
considered all 80 structural reorganizations of % + 00*(% + 1(0 + 1)*)1
% + 00*(% + 1(0 + 1)*)1 transformed by structural rule 4 at position [2, 2] to
% + 0(0*(% + 1(0 + 1)*))1 transformed by reduction rule 23 at position [2, 2, 1]
```

```
to % + 0(0 + 1)*1
considered all 8 structural reorganizations of % + 0(0 + 1)*1
% + 0(0 + 1)*1 is locally simplified
val it = (true,-) : bool * reg
```

Finally, note that $\% + 00^*1(\% + (0 + 1)^*1)$ is globally simplified—and so is also locally simplified:

```
- globPr "% + 00*1(% + (0 + 1)*1)";
% + 00*1(% + (0 + 1)*1)
val it = () : unit
```

We can also compare local and global simplification for this problem in another way. Consider the entire global simplification trace:

```
- globTr "(00*11*)*";
considering candidates with explanations of length 0
simplest result now: (00*11*)*
considering candidates with explanations of length 1
simplest result now: (00*11*)* transformed by reduction rule 16 at position []
to % + 0(0 + (11*)0)*11*
considering candidates with explanations of length 2
simplest result now: (00*11*)* transformed by reduction rule 16 at position []
to % + 0(0 + (11*)0)*11* weakly simplifies to % + 0(0 + 11*0)*11*
simplest result now: (00*11*)* transformed by reduction rule 16 at position []
to % + 0(0 + (11*)0)*11* transformed by reduction rule 22 at position
[2, 2, 1, 1] to % + 0((% + 11*)0)*11*
considering candidates with explanations of length 3
simplest result now: (00*11*)* transformed by reduction rule 16 at position []
to % + 0(0 + (11*)0)*11* transformed by reduction rule 22 at position
[2, 2, 1, 1] to % + 0((% + 11*)0)*11* transformed by reduction rule 25 at
position [2, 2, 1, 1, 1] to % + 0(1*0)*11*
considering candidates with explanations of length 4
simplest result now: (00*11*)* transformed by reduction rule 16 at position []
to % + 0(0 + (11*)0)*11* transformed by reduction rule 22 at position
[2, 2, 1, 1] to % + 0((% + 11*)0)*11* transformed by reduction rule 25 at
position [2, 2, 1, 1, 1] to % + 0(1*0)*11* transformed by reduction rule 15 at
position [2, 2, 1] to % + 0(% + (1 + 0)*0)11*
considering candidates with explanations of length 5
simplest result now: (00*11*)* transformed by reduction rule 16 at position []
to % + 0(0 + (11*)0)*11* transformed by structural rule 4 at position [2, 2] to
% + 0((0 + (11*)0)*1)1* transformed by reduction rule 22 at position
[2, 2, 1, 1, 1] to % + 0(((% + 11*)0)*1)1* transformed by reduction rule 25 at
position [2, 2, 1, 1, 1, 1] to % + 0((1*0)*1)1* transformed by reduction rule 15
at position [2, 2, 1, 1] to % + 0((% + (1 + 0)*0)1)1*
simplest result now: (00*11*)* transformed by reduction rule 16 at position []
to % + 0(0 + (11*)0)*11* transformed by reduction rule 22 at position
[2, 2, 1, 1] to % + 0((% + 11*)0)*11* transformed by reduction rule 25 at
position [2, 2, 1, 1, 1] to % + 0(1*0)*11* transformed by reduction rule 15 at
position [2, 2, 1] to % + 0(% + (1 + 0)*0)11* weakly simplifies to
% + 0(% + (0 + 1)*0)11*
considering candidates with explanations of length 6
considering candidates with explanations of length 7
simplest result now: (00*11*)* transformed by reduction rule 16 at position []
to % + 0(0 + (11*)0)*11* transformed by reduction rule 22 at position
[2, 2, 1, 1] to % + 0((% + 11*)0)*11* transformed by reduction rule 25 at
position [2, 2, 1, 1, 1] to % + 0(1*0)*11* transformed by structural rule 7 at
position [2, 2, 2] to % + 0(1*0)*1*1 transformed by structural rule 4 at
position [2, 2] to % + 0((1*0)*1*)1 weakly simplifies to % + 01*(01*)*1
transformed by reduction rule 14 at position [2, 2, 2, 1] to
% + 01*(% + 0(0 + 1)*)1
simplest result now: (00*11*)* transformed by structural rule 4 at position [1]
to ((00*)11*)* transformed by structural rule 7 at position [1, 2] to
((00*)1*1)* transformed by reduction rule 16 at position [] to
% + (00*)(1 + 100*)*1 transformed by reduction rule 21 at position [2, 2, 1, 1]
to % + (00*)(1(% + 00*))*1 weakly simplifies to % + 00*1((% + 00*)1)*
transformed by reduction rule 25 at position [2, 2, 2, 2, 1, 1] to
% + 00*1(0*1)* transformed by reduction rule 15 at position [2, 2, 2, 2] to
% + 00*1(% + (0 + 1)*1)
considering candidates with explanations of length 8
simplest result now: (00*11*)* transformed by reduction rule 16 at position []
to % + 0(0 + (11*)0)*11* transformed by reduction rule 22 at position
[2, 2, 1, 1] to % + 0((% + 11*)0)*11* transformed by reduction rule 25 at
position [2, 2, 1, 1, 1] to % + 0(1*0)*11* transformed by reduction rule 15 at
position [2, 2, 1] to % + 0(% + (1 + 0)*0)11* weakly simplifies to
% + 0(% + (0 + 1)*0)11* transformed by structural rule 7 at position [2, 2, 2]
to % + 0(% + (0 + 1)*0)1*1 transformed by structural rule 4 at position [2, 2]
to % + 0((% + (0 + 1)*0)1*)1 transformed by reduction rule 24 at position
[2, 2, 1] to % + 0(0 + 1)*1
considering candidates with explanations of length 9
considering candidates with explanations of length 10
considering candidates with explanations of length 11
considering candidates with explanations of length 12
considering candidates with explanations of length 13
considering candidates with explanations of length 14
considering candidates with explanations of length 15
considering candidates with explanations of length 16
considering candidates with explanations of length 17
considering candidates with explanations of length 18
considering candidates with explanations of length 19
considering candidates with explanations of length 20
search completed after considering 3796 candidates with maximum size 21
(00*11*)* transformed by reduction rule 16 at position [] to
% + 0(0 + (11*)0)*11* transformed by reduction rule 22 at position [2, 2, 1, 1]
to % + 0((% + 11*)0)*11* transformed by reduction rule 25 at position
[2, 2, 1, 1, 1] to % + 0(1*0)*11* transformed by reduction rule 15 at position
[2, 2, 1] to % + 0(% + (1 + 0)*0)11* weakly simplifies to
```

```
% + 0(% + (0 + 1)*0)11* transformed by structural rule 7 at position [2, 2, 2]
to % + 0(% + (0 + 1)*0)1*1 transformed by structural rule 4 at position [2, 2]
to % + 0((% + (0 + 1)*0)1*)1 transformed by reduction rule 24 at position
[2, 2, 1] to % + 0(0 + 1)*1 is globally simplified
val it = (true,-) : bool * reg
```

So the first step of global simplification uses reduction rule (16) to produce $\% + 0(0 + (11^*)0)^*11^*$. When local simplification considers this rule at stage 1, it first weakly simplifies it, obtaining

```
- weakSimpPr "% + 0(0 + (11*)0)*11*";
% + 0(0 + 11*0)*11*
val it = () : unit
```

This has the same complexity but is greater in the total ordering of regular expressions (used to break ties) as $\% + 00^*1(1 + 00^*1)^*$, which is what local simplification finds to be the locally optimal result:

```
- cmpCompl("% + 0(0 + 11*0)*11*", "% + 00*1(1 + 00*1)*");
val it = EQUAL : order
- cmpTot("% + 0(0 + 11*0)*11*", "% + 00*1(1 + 00*1)*");
val it = GREATER : order
```

But if local simplification had selected $\% + 0(0 + 11^*0)^*11^*$, the rest of the local simplification process would have produced the optimal result, α (note that the trace uses somewhat different steps than global simplification!):

```
- locTr "% + 0(0 + 11*0)*11*";
considered all 160 structural reorganizations of % + 0(0 + 11*0)*11*
% + 0(0 + 11*0)*11* transformed by structural rule 4 at position [2, 2, 1, 1, 2]
to % + 0(0 + (11*)0)*11* transformed by reduction rule 22 at position
[2, 2, 1, 1] to % + 0((% + 11*)0)*11*
considered all 160 structural reorganizations of % + 0((% + 11*)0)*11*
% + 0((% + 11*)0)*11* transformed by structural rule 7 at position [2, 2, 2] to
% + 0((% + 11*)0)*1*1 transformed by structural rule 4 at position [2, 2] to
% + 0(((% + 11*)0)*1*)1 transformed by reduction rule 25 at position
[2, 2, 1, 1, 1, 1] to % + 0((1*0)*1*)1 weakly simplifies to % + 01*(01*)*1
considered all 50 structural reorganizations of % + 01*(01*)*1
% + 01*(01*)*1 transformed by reduction rule 14 at position [2, 2, 2, 1] to
% + 01*(% + 0(0 + 1)*)1
considered all 40 structural reorganizations of % + 01*(% + 0(0 + 1)*)1
% + 01*(% + 0(0 + 1)*)1 transformed by structural rule 4 at position [2, 2] to
% + 0(1*(% + 0(0 + 1)*))1 transformed by structural rule 5 at position
[2, 2, 1, 2, 2, 2, 1] to % + 0(1*(% + 0(1 + 0)*))1 transformed by reduction rule
23 at position [2, 2, 1] to % + 0(1 + 0)*1 weakly simplifies to % + 0(0 + 1)*1
considered all 8 structural reorganizations of % + 0(0 + 1)*1
% + 0(0 + 1)*1 is locally simplified
val it = (true,-) : bool * reg
```

In summary, we can say that the non-optimality of local simplification for this problem stems from either its selection in the first step or the last step (and maybe some of the intermediate steps).

Problem 2

(a) If $\mathbf{hasEmp}\,\alpha$, i.e., $\% \in L(\alpha)$, then reduction rule (8) would also apply to $(\alpha\beta^*)^*$ ($\mathbf{hasEmp}(\beta^*)$ holds, because β^* is a closure), yielding $(\alpha+\beta^*)^*$. This can be restructured using structural rule (5) into $(\beta^*+\alpha)^*$, which can then be turned into $(\beta+\alpha)^*$ using reduction rule (7). So the rationale for restricting reduction rule (14) to only apply when $\mathbf{hasEmp}\,\alpha$ is false, is to prefer the above sequence of simplifications, over one starting with reduction rule (14).

(If the conservative approximation of *sub* is powerful enough, then reduction rule (14) can be followed by reduction rules (1) and (4), resulting in $(\alpha+\beta)^*$. But it's nicer not to have to depend upon properties of *sub*, or to incur the cost of running *sub*.)

(b) First we prove a lemma:

Lemma PS3.2.1
For all $ns, ms \in \mathbf{CC}$, *if* $ns <_{cc} ms$ *and* $|ns| \geq |ms|$, *then* $[0] \cup ns <_{cc} ms$.

Proof. Suppose $ns, ms \in \mathbf{CC}$, $ns <_{cc} ms$ and $|ns| \geq |ms|$. Because $ns <_{cc} ms$ and ns is at least as long as ms, there is an $i \in \mathbb{N} - \{0\}$ such that

- $i \leq |ns|$ and $i \leq |ms|$,
- for all $j \in [1 : i-1]$, $ns\,j = ms\,j$; and
- $ns\,i < ms\,i$.

Thus it follows that $i \in \mathbb{N} - \{0\}$,

- $i \leq |ns| < |ns| + 1 = |ns \mathbin{@} [0]|$ and $i \leq |ms|$;
- for all $j \in [1 : i-1]$, $(ns \mathbin{@} [0])\,j = ns\,j = ms\,j$; and
- $(ns \mathbin{@} [0])\,i = ns\,i < ms\,i$.

Because 0 is $\leq$ every element of ns, we have that $[0] \cup ns = ns \mathbin{@} [0]$, and so we can conclude from the above that $[0] \cup ns <_{cc} ms$. $\square$

Suppose $\alpha, \beta \in \mathbf{Reg}$ and $\mathbf{cc}\,\alpha \cup \overline{\mathbf{cc}\,\beta} <_{cc} \overline{\overline{\mathbf{cc}\,\beta}}$. By Proposition 3.3.1, we have that

$$\begin{aligned}
\mathbf{cc}(\% + \alpha(\alpha+\beta)^*) &= \mathbf{cc}\,\% \cup \mathbf{cc}(\alpha(\alpha+\beta)^*) = [0] \cup \mathbf{cc}\,\alpha \cup \mathbf{cc}((\alpha+\beta)^*) \\
&= [0] \cup \mathbf{cc}\,\alpha \cup \overline{\mathbf{cc}(\alpha+\beta)} = [0] \cup \mathbf{cc}\,\alpha \cup \overline{\mathbf{cc}\,\alpha \cup \mathbf{cc}\,\beta} \\
&= [0] \cup \mathbf{cc}\,\alpha \cup \overline{\mathbf{cc}\,\alpha} \cup \overline{\mathbf{cc}\,\beta} = [0] \cup \mathbf{cc}\,\alpha \cup \overline{\mathbf{cc}\,\beta} \cup \overline{\mathbf{cc}\,\alpha}, \\
\mathbf{cc}((\alpha\beta^*)^*) = \overline{\mathbf{cc}(\alpha\beta^*)} &= \overline{\mathbf{cc}\,\alpha \cup \mathbf{cc}(\beta^*)} = \overline{\mathbf{cc}\,\alpha \cup \overline{\mathbf{cc}\,\beta}} = \overline{\mathbf{cc}\,\alpha} \cup \overline{\overline{\mathbf{cc}\,\beta}} = \overline{\overline{\mathbf{cc}\,\beta}} \cup \overline{\mathbf{cc}\,\alpha}.
\end{aligned}$$

Thus to show $\mathbf{cc}(\% + \alpha(\alpha+\beta)^*) <_{cc} \mathbf{cc}((\alpha\beta^*)^*)$, it will suffice to show $[0] \cup \mathbf{cc}\,\alpha \cup \overline{\mathbf{cc}\,\beta} \cup \overline{\mathbf{cc}\,\alpha} <_{cc} \overline{\overline{\mathbf{cc}\,\beta}} \cup \overline{\mathbf{cc}\,\alpha}$. By Proposition 3.3.3(2), it will suffice to show $[0] \cup \mathbf{cc}\,\alpha \cup \overline{\mathbf{cc}\,\beta} <_{cc} \overline{\overline{\mathbf{cc}\,\beta}}$. And this follows by Lemma PS3.2.1, since $\mathbf{cc}\,\alpha \cup \overline{\mathbf{cc}\,\beta} <_{cc} \overline{\overline{\mathbf{cc}\,\beta}}$ and $|\mathbf{cc}\,\alpha \cup \overline{\mathbf{cc}\,\beta}| \geq |\overline{\mathbf{cc}\,\beta}| = |\overline{\overline{\mathbf{cc}\,\beta}}|$.

Problem 3

(a)

$$(\% + 1)(0 + 11(\% + 1))^*(\% + 1)$$

(b) Let

$$U = \{\%, 1\} \qquad \text{and} \qquad V = \{0, 11, 111\}.$$

Since $L(\alpha) = UV^*U$, it will suffice to show that $UV^*U = X$.

Every substring of an element of X is itself an element of X, because if the substring had an occurrence of 010, then so would the whole string.

Lemma PS3.3.1
(1) $UX \subseteq X$.

(2) $XU \subseteq X$.

Proof.

(1) Suppose $w \in UX$, so that $w = ux$ for some $u \in U$ and $x \in X$. We must show that $w \in X$. Clearly $w \in \{0, 1\}^*$. Suppose, toward a contradiction, that 010 is a substring of w. Because $u \in U$ and $w = ux$, this means that $w = x$ or $w = 1x$, and thus that 010 is a substring of x. But this contradicts the fact that $x \in X$. Thus 010 is not a substring of w, completing the proof that $w \in X$.

(2) Suppose $w \in XU$, so that $w = xu$ for some $x \in X$ and $u \in U$. We must show that $w \in X$. Clearly $w \in \{0, 1\}^*$. Suppose, toward a contradiction, that 010 is a substring of w. Because $u \in U$ and $w = xu$, this means that $w = x$ or $w = x1$, and thus that 010 is a substring of x. But this contradicts the fact that $x \in X$. Thus 010 is not a substring of w, completing the proof that $w \in X$.

□

Let

$$Y = \{\, w \in X \mid w \neq 1 \text{ and } 10 \text{ is not a prefix of } w \text{ and } 01 \text{ is not a suffix of } w \,\}.$$

It is easy to see that $\% \in Y$ and $V \subseteq Y$.

Lemma PS3.3.2
$YY \subseteq Y$.

Proof. Suppose $w \in YY$, so that $w = uv$ for some $u, v \in Y$. We must show that $w \in Y$.

Clearly $w \in \{0, 1\}^*$. To finish the proof that $w \in X$, suppose, toward a contradiction, that 010 is a substring of w. Because $w = uv$ and $u, v \in X$, there are two cases to consider.

- Suppose 01 is a suffix of u and 0 is a prefix of v. But $u \in Y$, and so 01 is not a suffix of u—contradiction.

- Suppose 0 is a suffix of u and 10 is a prefix of v. But $v \in Y$, and so 10 is not a prefix of v—contradiction.

Because we obtained a contradiction in both cases, we have an overall contradiction. Thus $w \in X$.

Because $u, v \in Y$, neither u nor v is 1, and thus $w = uv \neq 1$.

Suppose, toward a contradiction, that 10 is a prefix of w. Because $w = uv$, there are three cases to consider.

- Suppose $u = \%$ and 10 is a prefix of v. But $v \in Y$, and thus 10 is not a prefix of v—contradiction.
- Suppose $u = 1$ and 0 is a prefix of v. But $u \in Y$, and thus $u \neq 1$—contradiction.
- Suppose 10 is a prefix of u. But $u \in Y$, and thus 10 is not a prefix of u—contradiction.

Because we obtained an overall contradiction, it follows that 10 is not a prefix of w.

Suppose, toward a contradiction, that 01 is a suffix of w. Because $w = uv$, there are three cases to consider.

- Suppose $v = \%$ and 01 is a suffix of u. But $u \in Y$, and thus 01 is not a suffix of u—contradiction.
- Suppose $v = 1$ and 0 is a suffix of u. But $v \in Y$, and thus $v \neq 1$—contradiction.
- Suppose 01 is a suffix of v. But $v \in Y$, and thus 01 is not a suffix of v—contradiction.

Because we obtained an overall contradiction, it follows that 01 is not a suffix of w.

Because $w \in X$, $w \neq 1$, 10 is not a prefix of w, and 01 is not a suffix of w, it follows that $w \in Y$, as required. □

Lemma PS3.3.3
$Y^* \subseteq Y$.

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

(Basis) Since $\% \in Y$, we have that $Y^0 = \{\%\} \subseteq Y$.

(Inductive Step) Suppose $n \in \mathbb{N}$, and assume the inductive hypothesis: $Y^n \subseteq Y$. We must prove that $Y^{n+1} \subseteq Y$. We have that

$$\begin{aligned} Y^{n+1} &= YY^n \\ &\subseteq YY && \text{(inductive hypothesis)} \\ &\subseteq Y && \text{(Lemma PS3.3.2).} \end{aligned}$$

□

Lemma PS3.3.4
$UV^*U \subseteq X$.

Proof. Because $V \subseteq Y$, we have that $V^* \subseteq Y^* \subseteq Y \subseteq X$, by Lemma PS3.3.3. Thus $UV^* \subseteq UX \subseteq X$, by Lemma PS3.3.1(1). Finally, $UV^*U = (UV^*)U \subseteq XU \subseteq X$, by Lemma PS3.3.1(2). □

Lemma PS3.3.5
$X \subseteq UV^*U$.

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

$$\text{if } w \in X, \text{ then } w \in UV^*U.$$

We prove this using strong string induction. Suppose $w \in \{\mathtt{0},\mathtt{1}\}^*$, and assume the inductive hypothesis: for all $x \in \{\mathtt{0},\mathtt{1}\}^*$, if x is a proper substring of w, then

$$\text{if } x \in X, \text{ then } x \in UV^*U.$$

We must show that

$$\text{if } w \in X, \text{ then } w \in UV^*U.$$

Suppose $w \in X$. We must show that $w \in UV^*U$. Since $w \in \{\mathtt{0},\mathtt{1}\}^*$, there are three cases to consider.

- Suppose $w = \%$. Then $w = \% = \%\%\% \in UV^*U$.
- Suppose $w = \mathtt{1}t$, for some $t \in \{\mathtt{0},\mathtt{1}\}^*$. Because t is a substring of w, we have that $t \in X$. Furthermore, since t is a proper substring of w, the inductive hypothesis tells us that $t \in UV^*U$, so that $t = xyz$ for some $x \in U$, $y \in V^*$ and $z \in U$. Thus $w = \mathtt{1}t = \mathtt{1}xyz$. Because $x \in U$, there are two sub-cases to consider.
  - Suppose $x = \%$. Thus $w = \mathtt{1}xyz = \mathtt{1}\%yz = \mathtt{1}yz \in UV^*U$.
  - Suppose $x = \mathtt{1}$. Thus $w = \mathtt{1}xyz = \mathtt{11}yz = \%((\mathtt{11})y)z \in U(VV^*)U \subseteq UV^*U$.
- Suppose $w = \mathtt{0}t$, for some $t \in \{\mathtt{0},\mathtt{1}\}^*$. There are three sub-cases to consider.
  - Suppose $t = \%$. Thus $w = \mathtt{0}t = \mathtt{0}\% = \mathtt{0} = \%\mathtt{0}\% \in UVU \subseteq UV^*U$.
  - Suppose $t = \mathtt{0}s$, for some $s \in \{\mathtt{0},\mathtt{1}\}^*$. Because t is a substring of w, we have that $t \in X$. Furthermore, since t is a proper substring of w, the inductive hypothesis tells us that $t \in UV^*U$, so that $t = xyz$ for some $x \in U$, $y \in V^*$ and $z \in U$. Because $\mathtt{0}s = t = xyz$, x cannot be $\mathtt{1}$. But $x \in U$, and thus $x = \%$. Thus $w = \mathtt{0}t = \mathtt{0}xyz = \mathtt{0}\%yz = \mathtt{0}yz = \%(\mathtt{0}y)z \in U(VV^*)U \subseteq UV^*U$.
  - Suppose $t = \mathtt{1}s$, for some $s \in \{\mathtt{0},\mathtt{1}\}^*$. Thus $w = \mathtt{0}t = \mathtt{01}s$. Because $w \in X$, s cannot begin with $\mathtt{0}$. Thus there are two sub-sub-cases to consider.
    - Suppose $s = \%$. Thus $w = \mathtt{01}s = \mathtt{01}\% = \mathtt{01} = \%\mathtt{01} \in UVU \subseteq UV^*U$.
    - Suppose $s = \mathtt{1}r$, for some $r \in \{\mathtt{0},\mathtt{1}\}^*$. Thus $w = \mathtt{01}s = \mathtt{011}r$. Because r is a substring of w, we have that $r \in X$. Furthermore, since r is a proper substring of w, the inductive hypothesis tells us that $r \in UV^*U$, so that $r = xyz$ for some $x \in U$, $y \in V^*$ and $z \in U$. Thus $w = \mathtt{011}r = \mathtt{011}xyz$. Since $x \in U$, there are two sub-sub-sub-cases to consider.
      - Suppose $x = \%$. Thus $w = \mathtt{011}xyz = \mathtt{011}\%yz = \mathtt{011}yz = \%(\mathtt{0}(\mathtt{11})y)z \in U(VVV^*)U \subseteq UV^*U$.

- Suppose $x = \mathtt{1}$. Thus $w = \mathtt{011}xyz = \mathtt{0111}yz = \%(\mathtt{0}(\mathtt{111})y)z \in U(VVV^*)U \subseteq UV^*U$.

□

By Lemmas PS3.3.4 and PS3.3.5, we have that $UV^*U \subseteq X \subseteq UV^*U$, i.e., $UV^*U = X$.