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

# Problem Set 7

## Model Answers

### Problem 1

We prove that $X$ is not context-free. Suppose, toward a contradiction, that $X$ is context-free. Thus there is an $n \in \mathbb{N}$ with the property of the Pumping Lemma for Context-free Languages, where $X$ has been substituted for $L$. Because there are infinitely many primes, there is a $p \in \mathbb{N}$ such that $p \geq n$ and $p$ is prime. Let $z = \mathsf{0}^p$, so that $|z| = p$ is prime, and thus $z \in X$. Furthermore, $|z| = p \geq n$, and thus it follows that there are $u, v, w, x, y \in \mathbf{Str}$ such that $z = uvwxy$ and properties (1)–(3) of the lemma hold. Since $\mathsf{0}^p = z = uvwxy$, there are $i, j, k, l, m \in \mathbb{N}$ such that

$$u = \mathsf{0}^i, \quad v = \mathsf{0}^j, \quad w = \mathsf{0}^k, \quad x = \mathsf{0}^l \quad y = \mathsf{0}^m,$$

and $i + j + k + l + m = p$. By (2), we know that $j + l \geq 1$. There are two cases to consider.

- Suppose $i+k+m \leq 1$. By (3), we have that $\mathsf{0}^{i+k+m} = \mathsf{0}^i\mathsf{0}^k\mathsf{0}^m = uwy = u\%w\%y = uv^0wx^0y \in X$. Thus $i + k + m$ is prime. But $i + k + m \leq 1$—contradiction.

- Suppose $i + k + m \geq 2$. By (3), we have that

$$\begin{aligned} \mathsf{0}^{i+(i+k+m)j+k+(i+k+m)l+m} &= \mathsf{0}^i\mathsf{0}^{(i+k+m)j}\mathsf{0}^k\mathsf{0}^{(i+k+m)l}\mathsf{0}^m \\ &= \mathsf{0}^i(\mathsf{0}^j)^{i+k+m}\mathsf{0}^k(\mathsf{0}^l)^{i+k+m}\mathsf{0}^m \\ &= uv^{i+k+m}wx^{i+k+m}y \\ &\in X, \end{aligned}$$

  showing that $i + (i + k + m)j + k + (i + k + m)l + m$ is prime. But

$$\begin{aligned} i + (i + k + m)j + k + (i + k + m)l + m &= (i + k + m) + (i + k + m)j + (i + k + m)l \\ &= (i + k + m)(1 + j + l), \end{aligned}$$

  showing that $(i + k + m)(1 + j + l)$ is prime. Since $j + l \geq 1$, we have that $1 + j + l \geq 2$. And $i + k + m \geq 2$, showing that $(i + k + m)(1 + j + l)$ has a factor other than 1 and itself—contradiction.

Since we obtained a contraction in both cases, we have an overall contradiction. Thus $X$ is not context-free.

### Problem 2

We put a grammar generating all elements of $\{\mathsf{0}, \mathsf{1}\}^*$ with twice as many occurrences of $\mathsf{0}$ as $\mathsf{1}$ in the file `ps7-twice0as1-gram`:

```
{variables} A {start variable} A
{productions}
A -> % | 0A0A1 | 0A1A0 | 1A0A0 | AA
```

(See Problem Set 2 for the origin of this grammar.) And we put a DFA accepting all elements of $\{0,1\}^*$ with an even number of occurrences of 0 in the file `ps7-even0-dfa`:

```
{states} A, B {start state} A {accepting states} A
{transitions}
A, 0 -> B; A, 1 -> A;
B, 0 -> A; B, 1 -> B
```

Next, we load our grammar and DFA into Forlan:

```
- val twice0As1Gram = Gram.input "ps7-twice0as1-gram";
val twice0As1Gram = - : gram
- val even0DFA      = DFA.input "ps7-even0-dfa";
val even0DFA = - : dfa
```

Then we use alphabet renaming to create a grammar generating all elements of $\{0,1\}^*$ with twice as many occurrences of 1 as 0, and a DFA accepting all elements of $\{0,1\}^*$ with an even number of occurrences of 1:

```
- val swapRel       = SymRel.fromString "(0, 1), (1, 0)";
val swapRel = - : sym_rel
- val twice1As0Gram = Gram.renameAlphabet(twice0As1Gram, swapRel);
val twice1As0Gram = - : gram
- val even1DFA      = DFA.renameAlphabet(even0DFA, swapRel);
val even1DFA = - : dfa
```

Then we create the grammar generating $X$, and display it:

```
- val gram =
=       Gram.renameVariablesCanonically
=       (Gram.simplify
=        (Gram.union
=         (Gram.inter(twice0As1Gram, injDFAToEFA even1DFA),
=          Gram.inter(twice1As0Gram, injDFAToEFA even0DFA))));
val gram = - : gram
- Gram.output("", gram);
{variables} A, B, C, D, E, F, G, H, I, J, K {start variable} A
{productions}
A -> B | C; B -> D; C -> H;
D -> % | DD | EF | 0D0E1 | 0D1F0 | 0E0G1 | 0E1D0 | 1F0D0 | 1G0F0;
E -> DE | EG | 0D0D1 | 0D1G0 | 0E0F1 | 0E1E0 | 1F0E0 | 1G0G0;
F -> FD | GF | 0F0E1 | 0F1F0 | 0G0G1 | 0G1D0 | 1D0D0 | 1E0F0;
G -> % | FE | GG | 0F0D1 | 0F1G0 | 0G0F1 | 0G1E0 | 1D0E0 | 1E0G0;
H -> % | HH | IJ | 0J1H1 | 0K1J1 | 1H0J1 | 1H1I0 | 1I0H1 | 1I1K0;
I -> HI | IK | 0J1I1 | 0K1K1 | 1H0K1 | 1H1H0 | 1I0I1 | 1I1J0;
J -> JH | KJ | 0H1H1 | 0I1J1 | 1J0J1 | 1J1I0 | 1K0H1 | 1K1K0;
K -> % | JI | KK | 0H1I1 | 0I1K1 | 1J0K1 | 1J1H0 | 1K0I1 | 1K1J0
```

```
val it = () : unit
```

Finally, we test our grammar on all elements of $\{0,1\}^*$ with length no more than 15:

```
- val zero = Sym.fromString "0";
val zero = - : sym
- val one  = Sym.fromString "1";
val one = - : sym
- fun zeros(nil : str) : int = 0
=   | zeros(b :: bs)         =
=       if Sym.equal(b, zero) then 1 + zeros bs else zeros bs;
val zeros = fn : str -> int
- fun ones(nil : str) : int = 0
=   | ones(b :: bs)         =
=       if Sym.equal(b, one) then 1 + ones bs else ones bs;
val ones = fn : str -> int
- val zs = StrSet.power(StrSet.fromString "%, 0, 1", 15);
val zs = - : str set
- fun valid(w : str) =
=       zeros w = 2 * ones w andalso ones w mod 2 = 0
=       orelse
=       ones w = 2 * zeros w andalso zeros w mod 2 = 0;
val valid = fn : str -> bool
- val (goods, bads) = Set.partition valid zs;
val goods = - : str set
val bads = - : str set
- val generated     = Gram.generated gram;
val generated = fn : str -> bool
- Set.all generated goods andalso Set.all (not o generated) bads;
val it = true : bool
```