CS 516—Software Foundations via Formal Languages—Spring 2025

Problem Set 7

Model Answers

Problem 1

Let G be the grammar

$$\mathsf{A} \to \% \mid 1 \mid \mathsf{1A1A0} \mid \mathsf{AA}$$

Let X be the least subset of $\{0, 1\}^*$ such that:

(1) $\% \in X$;

(2) $1 \in X$;

(3) for all $x, y \in X$, $1x1y0 \in X$;

(4) for all $x, y \in X$, $xy \in X$.

In Problem Set 2, we proved $X = Y$. Hence it will suffice to show that $L(G) = X$.

Lemma PS7.1.1
For all $w \in \Pi_{\mathsf{A}}$, $w \in X$.

Proof. We proceed by induction on Π. There are four productions to consider.

($\mathsf{A} \to \%$) We must show that $\% \in X$, and this follows by Rule (1) of X's definition.

($\mathsf{A} \to 1$) We must show that $1 \in X$, and this follows by Rule (2) of X's definition.

($\mathsf{A} \to \mathsf{1A1A0}$) Suppose $x, y \in \Pi_{\mathsf{A}}$, and assume the inductive hypothesis: $x, y \in X$. Then $1x1y0 \in X$ by Rule 3 of X's definition.

($\mathsf{A} \to \mathsf{AA}$) Suppose $x, y \in \Pi_{\mathsf{A}}$, and assume the inductive hypothesis: $x, y \in X$. Then $xy \in X$ by Rule 4 of X's definition.

□

Lemma PS7.1.2
For all $w \in X$, $w \in \Pi_{\mathsf{A}}$.

Proof. We proceed by induction on X. There are four steps to show.

(1) We must show that $\% \in \Pi_{\mathsf{A}}$. And this follows because of the production $\mathsf{A} \to \%$ of G.

(2) We must show that $1 \in \Pi_{\mathsf{A}}$. And this follows because of the production $\mathsf{A} \to 1$ of G.

(3) Suppose $x, y \in X$, and assume the inductive hypothesis: $x, y \in \Pi_{\mathsf{A}}$. We must show that $1x1y0 \in \Pi_{\mathsf{A}}$, and this follows because of the production $\mathsf{A} \to \mathsf{1A1A0}$ and the inductive hypothesis.

(4) Suppose $x, y \in X$, and assume the inductive hypothesis: $x, y \in \Pi_{\mathsf{A}}$. We must show that $xy \in \Pi_{\mathsf{A}}$, and this follows because of the production $\mathsf{A} \to \mathsf{AA}$ and the inductive hypothesis.

□

Lemma PS7.1.1 tells us that $L(G) = \Pi_{\mathsf{A}} \subseteq X$, and Lemma PS7.1.2 tells us that $X \subseteq \Pi_{\mathsf{A}} = L(G)$. Thus $L(G) = X$.

Problem 2

First, we put the text

```
{variables} A, B, C, D {start variable} A
{productions}
A -> B | C | 0A3;
B -> D | 0B2;
C -> D | 1C3;
D -> % | 1D2
```

for a grammar generating $\{\, 0^i 1^j 2^k 3^l \mid i, j, k, l \in \mathbb{N} \text{ and } i + j = k + l \,\}$ in the file `ps7-p2-orig-gram.txt`. Next we put the text

```
{states} A, B {start state} A {accepting states} A
{transitions}
A, 0 -> B; A, 1 -> A; A, 2 -> A; A, 3 -> A;
B, 0 -> A; B, 1 -> B; B, 2 -> B; B, 3 -> B
```

for a DFA accepting all elements of $\{0, 1, 2, 3\}^*$ with an even number of 0's in the file `ps7-p2-even0s-dfa.txt`. Then we load the grammar and DFA into Forlan:

```
- val origGram = Gram.input "ps7-p2-orig-gram.txt";
val origGram = - : gram
- val even0sDFA = DFA.input "ps7-p2-even0s-dfa.txt";
val even0sDFA = - : dfa
```

Next, we put the Forlan program

```
(* standard definitions *)

val regToDFA  = nfaToDFA o efaToNFA o faToEFA o regToFA;
val minAndRen = DFA.renameStatesCanonically o DFA.minimize;

(* the alphabet {0, 1, 2, 3} *)

val syms0123 = SymSet.fromString "0, 1, 2, 3";

(* regular expression and DFA generating/accepting {0, 1, 2, 3}* *)

val allStrReg = Reg.closure(Reg.allSym syms0123);
val allStrDFA = minAndRen(regToDFA allStrReg);
```

```
(* symbolic relation on {0, 1, 2, 3} swapping 0 and 1 *)

val swap01 = SymRel.fromString "(0, 1), (1, 0), (2, 2), (3, 3)";

(* symbolic relation on {0, 1, 2, 3} swapping 0 and 2 *)

val swap02 = SymRel.fromString "(0, 2), (2, 0), (1, 1), (3, 3)";

(* symbolic relation on {0, 1, 2, 3} swapping 0 and 3 *)

val swap03 = SymRel.fromString "(0, 3), (3, 0), (1, 1), (2, 2)";

(* DFA accepting all elements of {0, 1, 2, 3}* with odd number of 1s *)

val odd1sDFA =
      minAndRen
      (DFA.minus
       (allStrDFA,
        DFA.renameAlphabet(even0sDFA, swap01)));

(* DFA accepting all elements of {0, 1, 2, 3}* with even number of 2s *)

val even2sDFA = DFA.renameAlphabet(even0sDFA, swap02);

(* DFA accepting all elements of {0, 1, 2, 3}* with odd number of 3s *)

val odd3sDFA =
      minAndRen
      (DFA.minus
       (allStrDFA,
        DFA.renameAlphabet(even0sDFA, swap03)));

(* DFA accepting all elements of {0, 1, 2, 3}* in which the
   number of 0s is even and the number of 1s is odd and the
   number of 2s is even and the number of 3s is odd *)

val paritiesDFA =
      minAndRen
      (DFA.genInter
       [even0sDFA, odd1sDFA, even2sDFA, odd3sDFA]);

(* grammar generating X *)

val gram0 =
      Gram.renameVariablesCanonically
      (Gram.inter(origGram, injDFAToEFA paritiesDFA));
```

in the file `ps7-p2-find.sml`, and proceed as follows:

```
- use "ps7-p2-find.sml";
[opening ps7-p2-find.sml]
val regToDFA = fn : reg -> dfa
val minAndRen = fn : dfa -> dfa
val syms0123 = - : sym set
val allStrReg = - : reg
val allStrDFA = - : dfa
val swap01 = - : sym_rel
val swap02 = - : sym_rel
val swap03 = - : sym_rel
val odd1sDFA = - : dfa
val even2sDFA = - : dfa
val odd3sDFA = - : dfa
val paritiesDFA = - : dfa
val gram0 = - : gram
val it = () : unit
- Gram.output("", gram0);
{variables} A, B, C, D, E, F, G, H, I {start variable} A
{productions}
A -> B; B -> F | 0C3; C -> E | 0B3; D -> H | 0E2; E -> 0D2; F -> 1G3;
G -> I | 1F3; H -> 1I2; I -> % | 1H2
val it = () : unit
```

In the grammar `gram0`, there are opportunities for hand-simplification using Forlan. We put the text

```
(* sumProdRHSLens : gram -> int

   sum the lengths of the right-hand sides of a grammar's
   productions *)

fun sumProdRHSLens gram =
      let fun sum nil             = 0
            | sum ((_, bs) :: ps) = length bs + sum ps
      in sum(Set.toList(Gram.productions gram)) end

(* better : gram * gram -> bool

   metric for gram1 being "better" than gram2 *)

fun better(gram1, gram2) =
      let val nv1 = Gram.numVariables gram1
          val nv2 = Gram.numVariables gram2
      in nv1 < nv2 orelse
         (nv1 = nv2 andalso
          let val np1 = Gram.numProductions gram1
              val np2 = Gram.numProductions gram2
```

```
          in np1 < np2 orelse
             (np1 = np2 andalso
              let val n1 = sumProdRHSLens gram1
                  val n2 = sumProdRHSLens gram2
              in n1 < n2 end)
          end)
      end;

(* best : gram * gram option list -> gram

   best(gram, gramOpts) returns gram if none of the optional
   grammars in gramOpts are better than gram; otherwise it returns
   one of the optional grammars that is better than gram and
   such that no other optional grammar is even better *)

fun best(gram, nil)                    = gram
  | best(gram, NONE :: gramOpts)       = best(gram, gramOpts)
  | best(gram, SOME gram' :: gramOpts) =
      if better(gram', gram)
      then best(gram', gramOpts)
      else best(gram, gramOpts);

(* elims : gram -> gram

   recursively eliminate variables of a grammar *)

fun elims gram =
      let val qs       =
            SymSet.minus
            (Gram.variables gram, Set.sing(Gram.startVariable gram))
          val gramOpts =
            map (fn q =>
                   case Gram.eliminateVariableOpt(gram, q) of
                        NONE       => NONE
                      | SOME gram' => SOME(elims gram'))
                (Set.toList qs)
      in best(gram, gramOpts) end;

(* handSimp : gram -> gram

   hand-simplify a grammar *)

fun handSimp gram =
      let val gram = elims gram
      in case Gram.restartOpt gram of
              NONE       => gram
            | SOME gram' => gram'
      end;
```

in the file `ps7-p2-hand-simp.sml`, and proceed as follows to obtain our answer, `gram`:

```
- use "ps7-p2-hand-simp.sml";
[opening ps7-p2-hand-simp.sml]
val sumProdRHSLens = fn : gram -> int
val better = fn : gram * gram -> bool
val best = fn : gram * gram option list -> gram
val elims = fn : gram -> gram
val handSimp = fn : gram -> gram
val it = () : unit
- val gram = Gram.renameVariablesCanonically(handSimp gram0);
val gram = - : gram
- Gram.output("", gram);
{variables} A, B, C, D {start variable} A
{productions}
A -> 0B3 | 1C3 | 00A33; B -> 00B22 | 01D22; C -> D | 11C33; D -> % | 11D22
val it = () : unit
- (Gram.numVariables gram, Gram.numProductions gram);
val it = (4,9) : int * int
```

Problem 3

Suppose, toward a contradiction, that X is context-free. Thus there is an $n \in \mathbb{N} - \{0\}$ with the property of the Pumping Lemma for Context-free Languages, where X has been substituted for L. Let $z = \mathsf{0}^n\mathsf{1}^{n+1}\mathsf{2}^{n+2}$. Since $n < n+1 < n+2$, we have that $z \in X$. Furthermore $|z| = 3n+3 \geq n$, and thus the property of the lemma tells us there are $u, v, w, x, y \in \mathbf{Str}$ such that $z = uvwxy$ and

(1) $|vwx| \leq n$; and

(2) $vx \neq \%$; and

(3) $uv^iwx^iy \in X$, for all $i \in \mathbb{N}$.

Because $\mathsf{0}^n\mathsf{1}^{n+1}\mathsf{2}^{n+2} = z = uvwxy$, we have that $u, v, w, x, y \in \{\mathsf{0}, \mathsf{1}, \mathsf{2}\}^*$, and (1) tells us that vwx does not have both 0's and 2's. There are four cases to consider:

(vx has one or more 0's, and has one or more 1's) Then vx has no 2's. Let k be the number of 1's in vx, so that $k \geq 1$. By (3), we have that $uvvwxxy = uv^2wx^2y \in X$. Since $uvwxy$ has $n+1$ 1's and $n+2$ 2's, $u(v)vwx(x)y$ has $n+1+k$ 1's, and has $n+2$ 2's. But $n+1+k \geq n+2$, so that $uvvwxxy \notin X$—contradiction.

(vx has one or more 0's, but has no 1's) Let k be the number of 0's in vx, so that $k \geq 1$. By (3), we have that $uvvwxxy = uv^2wx^2y \in X$. Since $uvwxy$ has n 0's and $n+1$ 1's, $u(v)vwx(x)y$ has $n+k$ 0's, and has $n+1$ 1's. But $n+k \geq n+1$, so that $uvvwxxy \notin X$—contradiction.

(vx has no 0's, but has one or more 1's) Let k be the number of 1's in vx, so that $k \geq 1$. By (3), we have that $uwy = uv^0wx^0y \in X$. Since $u(v)w(x)y$ has n 0's and $n+1$ 1's, uwy has n 0's, and has $n+1-k$ 1's. But $n \geq n+1-k$, so that $uwy \notin X$—contradiction.

(vx has no 0's, and has no 1's) By (2), it follows that vx has one or more 2's. Let k be the number of 2's in vx, so that $k \geq 1$. By (3), we have that $uwy = uv^0wx^0y \in X$. Since $u(v)w(x)y$ has $n+1$ 1's and $n+2$ 2's, uwy has $n+1$ 1's, and has $n+2-k$ 2's. But $n+1 \geq n+2-k$, so that $uwy \notin X$—contradiction.

Because we obtained a contradiction in each case, we have an overall contradiction. Thus X is not context-free.