CS 381 Fall 2000
Solutions to Homework 13

32.1

Let P1 stand for \{ (NPDA_1, NPDA_2) \mid L(NPDA_1) \cup L(NPDA_2) \neq \emptyset \}
Let P2 stand for \{ M \mid L(M) \neq \emptyset \}

Call the variant of the VALCOMPS given in the hint has REVCOMPS. Given a string
x of the form \alpha_0 \# rev(\alpha_1) \# \alpha_2 \# rev(\alpha_3) \# \ldots
in REVCOMPS, a NPDA can check if
\alpha_{2i+1} can legally follow \alpha_{2i}.
Using this fact we will reduce P2 to P1.

It should be clear that there is a one to one correspondence between the sets VAL-
COMPS & REVCOMPS. i.e the cardinality of both the sets is same. If \alpha_0 \# \alpha_1 \# \alpha_2 \# \ldots
is in VALCOMPS then \alpha_0 \# rev(\alpha_1) \# \alpha_2 \# \ldots
is in REVCOMPS.

If a TM M is in P2 then the set VALCOMPS corresponding to it is not empty.
Now we construct NPDA_1 & NPDA_2 which accept strings of the form in REVCOMPS as follows:

NPDA_1 : It checks whether \alpha_{2i+1} can legally follow \alpha_{2i}. If it is not a legal move then it
rejects the string. If the string has an odd number of \alpha_i’s, then the last \alpha_i must represent
an accepting configuration, otherwise it rejects the string.\n
NPDA_2 : It ignores \alpha_0. It checks whether \alpha_{2i} can legally follow \alpha_{2i-1}. If it is not a
legal move then it rejects the string. If the string has an even number of \alpha_i’s, then the last
\alpha_i must represent an accepting configuration, otherwise it rejects the string, else accepts the
string.

If M accepts x, let \beta = \alpha_0 \# \alpha_1 \# \alpha_2 \# \ldots \# \alpha_n is in VALCOMPS which represent the set
of legal moves made by M while accepting x. It follows that \beta_1 = \alpha_0 \# rev(\alpha_1) \# \alpha_2 \# \ldots
is accepted by both NPDA_1 & NPDA_2. Hence their intersection is not empty.
If M is not in P1 then VALCOMPS corresponding to M is empty which implies REVCOMPS is empty
which implies the language accepted by NPDA_1 and NPDA_2 are empty. Hence we have
reduce P1 to P2. Therefore P2 is undecidable.

32.2

Let P1 stand for \{ G \mid L(G) = \sum^* \}
Let P2 stand for \{ G \mid L(G) = \sum^* - \epsilon \}

We will reduce P1 to P2. Given a G, first determine if \epsilon is in L(G). If not then G
cannot be in P1 so reject it. else construct \textit{G'} such that L(G') = L(G) - \epsilon. Then G is in
P1 iff \textit{G'} is in P2. Since P1 is undecidable, P2 is also undecidable.

33.1

Unrestricted grammar for \textit{ww} over a,b.

The approach : The first \textit{w} is generated between markers \textit{M}_1 and \textit{M}_2. Then the first
symbol to the right of \textit{M}_1 is recorded and \textit{M}_1 is advanced. The recorded symbol is copied
just to the left of \textit{M}_4. Thus we generate the second \textit{w} between \textit{M}_3 and \textit{M}_4. If you dont
like this approach, an alternate shorter grammar by Martin is given after this one.
\[ S \rightarrow M_1XM_2M_3M_4 \]
\[ S \rightarrow \epsilon \]
\[ X \rightarrow aX \]
\[ X \rightarrow bX \]
\[ aXM_2 \rightarrow M_2a \]
\[ bXM_2 \rightarrow M_2b \]
\[ aM_2 \rightarrow M_2a \]
\[ bM_2 \rightarrow M_2b \]
\[ M_1aM_2 \rightarrow aM_1A \]
\[ M_1bM_2 \rightarrow bM_1B \]
\[ Aa \rightarrow aA \]
\[ Ba \rightarrow aB \]
\[ Ab \rightarrow bA \]
\[ Bb \rightarrow bB \]
\[ AM_3 \rightarrow M_3A \]
\[ BM_3 \rightarrow M_3B \]
\[ AM_4 \rightarrow M_2aM_4 \]
\[ BM_4 \rightarrow M_2bM_4 \]
\[ aM_2 \rightarrow M_2a \]
\[ bM_2 \rightarrow M_2b \]
\[ M_1M_3M_2 \rightarrow E \]
\[ aM_3M_2 \rightarrow M_2aM_3 \]
\[ bM_3M_2 \rightarrow M_2bM_3 \]
\[ Ea \rightarrow aE \]
\[ Eb \rightarrow bE \]
\[ EM_4 \rightarrow \epsilon \]

Martin's solution:
\[ S \rightarrow AS \mid BS \mid T \]
\[ A \rightarrow A_1A_2 \]
\[ B \rightarrow B_1B_2 \]
\[ A_2A_1 \rightarrow A_1A_2 \]
\[ A_2B_1 \rightarrow B_1A_2 \]
\[ B_2B_1 \rightarrow B_1B_2 \]
\[ B_2A_1 \rightarrow A_1B_2 \]
\[ A_2T \rightarrow Ta \]
\[ B_2T \rightarrow Tb \]
\[ T \rightarrow U \]
\[ A_1U \rightarrow Ua \]
\[ B_1U \rightarrow Ub \]
\[ U \rightarrow \epsilon \]

33.2

Post system for computing \( f(n) = 3^n \).
\[ \Sigma = \{1, \cdot\}, \quad N = \{S\}, \quad \text{variables} \quad X \text{and} \quad Y. \quad \text{Productions} \]

\[ S \rightarrow 1 \]
\[ X \cdot Y \rightarrow X1 \cdot YYY \]

**33.3**

Prove that function \( f \) such that \( f(n) = n \) form \( n \geq 3 \) and undefined for \( n = 0, 1, 2 \) is \( \mu \) recursive.

We need to show that \( f \) can be expressed using primitive recursive functions and minimization. Since primitive recursive functions cannot express infinite loops, we need to use minimization. Thus

\[ f(n) = \mu x. \ (g(n, x) = 0) \]

for some primitive recursive function \( g \). Note that this is nothing else as the following piece of C code:

```c
int f(int n)
{
    int x;
    x = 0;
    while (g(n, x) > 0)
    {
        x = x+1;
        return x;
    }
}
```

We want that the function loops forever if \( n < 3 \). Thus we would like \( g(n, x) > 0 \) for all \( n < 3 \) and all \( x \). For \( n \geq 3 \), we need the loop to stop exactly when \( x = n \), therefore we require \( g(n, x) > 0 \) for \( x < n \) and \( g(n, n) = 0 \). The function

\[ g(n, x) = (n - x) + (3 - x) \]

does exactly what we need, since the first term is positive for \( n > x \) and the second is positive exactly for \( n < 3 \).

**34.1**

\[ \lambda fgh. \ \lambda x. \ f(gx)(hx) \space \lambda yz. \ (y + z^2) \space \lambda x. \ \sin x \space \lambda x. \ x + 1 \space \rightarrow \]
\[ \lambda gh. \ \lambda x. \ (\lambda yz. \ (y + z^2) \space (gx) \space (hx)) \space \lambda x. \ \sin x \space \lambda x. \ x + 1 \space \rightarrow \]
\[ \lambda h. \ \lambda x. \ (\lambda yz. \ (y + z^2)) \space (\lambda x. \ \sin x \space x \space (hx)) \space (\lambda x. \ x + 1) \space \rightarrow \]
\[ \lambda x. \ ((\lambda yz. \ (y + z^2)) \space ((\lambda x. \ \sin x \space x) \space ((\lambda x. \ x + 1) \space x)) \space \rightarrow \]
\[ \lambda x. \ \sin x + ((\lambda x. \ x + 1) \space x) \space \rightarrow \]
\[ \lambda x. \ \sin x + (x + 1)^2 \]

We are doing detailed step-by-step substitution; however, in practice it is possible to do multiple steps at once. You can perform the substitutions (\( \beta \)-reductions) in any order you wish.
35.1

\[ \forall n. \exists p. \ p > n \land \text{Prime}(p) \land \text{Prime}(p + 2) \]  \hspace{1cm} (1)

35.2

There were many solutions to this question. We

\[ \text{PRIMEPOWER}(x) = \exists p. \ \text{PRIME}(p). \ \forall q. \ \text{DIV}(q, x) \Rightarrow \text{DIV}(p, q) \]  \hspace{1cm} (2)