Assignment 1:
Grammars and Induction

15-312: Foundations of Programming Languages
Joshua Dunfield (joshuad@cs.cmu.edu)

Out: Thursday, August 29, 2002
Due: Thursday, September 5, 2002 (1:30 pm)

50 points total

Welcome to 15-312! This assignment focuses on context-free grammars and inductive proofs. It is due September 5th at the start of lecture. You are encouraged, but not required, to typeset your answers; if you write by hand, write legibly. If I can’t read it, I can’t give credit for it.

Please make sure you understand the policy on collaboration; refer to http://www.cs.cmu.edu/~fp/courses/312/assignments.html

1 Grammars (35 points)

Consider the grammar $G$ (which should look familiar) over the alphabet $\Sigma = \{ \text{int, list, ->, (,)} \}$, with nonterminals tycon and type:

\[
\begin{align*}
tycon & ::= \text{int} | \text{tycon list} | (\text{type}) \\
type & ::= \text{tycon} | \text{type} \rightarrow \text{type}
\end{align*}
\]

Question 1.1 (5 points).

The first production for $\text{tycon}$ can be written in rule notation as

\[
\text{int tycon}
\]

Write grammar $G$ in rule notation.
Grammar $G$ is flawed: it does not pin down the associativity of $\rightarrow$. For example, there are two different derivations of the string $\text{int}\rightarrow\text{int}\rightarrow\text{int}$. We can fix the ambiguity by changing the second production of type from

\[
\text{type} ::= \text{type} \rightarrow \text{type}
\]

to

\[
\text{type} ::= \text{tycon} \rightarrow \text{type}
\]

Making this change results in the grammar $G'$ below. To avoid confusion, we rename tycon to $\text{tycon}'$ and type to $\text{type}'$.

\[
\begin{align*}
\text{tycon}' &::= \text{int} \mid \text{tycon}' \text{ list} \mid (\text{type}') \\
\text{type}' &::= \text{tycon}' \mid \text{tycon}' \rightarrow \text{type}'
\end{align*}
\]

**Question 1.2** (5 points).

Write grammar $G'$ in rule notation.

However, we haven’t proved that this new grammar $G'$ really is equivalent to $G$, that is, that the languages of type and $\text{type}'$ are the same: $L(\text{type}) = L(\text{type}')$. This can be proved in two steps: first prove $L(\text{type}') \subseteq L(\text{type})$, then prove $L(\text{type}) \subseteq L(\text{type}')$.

**Question 1.3** (10 points).

Prove $L(\text{type}') \subseteq L(\text{type})$ by proving

If $s \text{ type}'$ then $s \text{ type}$

by induction. If you need to generalize the induction hypothesis, be sure to clearly state your generalized induction hypothesis. If you need any lemmas, state them explicitly and prove them.

**Question 1.4** (15 points).

Prove $L(\text{type}) \subseteq L(\text{type}')$ by proving

If $s \text{ type}$ then $s \text{ type}'$

As in the previous question, clearly state any generalized induction hypothesis and prove any lemmas you need.
2 Propositional logic (15 points)

In this question we will look at a subset of Propositional Logic. Our universe of terms consists of an infinite number of arity 0 operators $P_0, P_1, \ldots, P_n$ ("propositional variables"), and the binary operator $\Rightarrow$ ("implication"). We define the sets prop and thm over this universe:

\[
\begin{align*}
& P_i \text{ prop} \quad \text{Var} \\
& A \text{ prop} \quad B \text{ prop} \quad \Rightarrow \text{ thm} \\
& A \Rightarrow B \text{ prop} \quad \Rightarrow \text{ Imp} \\
& A \Rightarrow B \text{ thm} \quad A \text{ thm} \quad \Rightarrow \text{ App}
\end{align*}
\]

**Truth Value.** If we have assignments (to true or false) for all of the propositional variables in a proposition, its truth value (either true or false) can be computed recursively using the following familiar truth table for $\Rightarrow$:

<table>
<thead>
<tr>
<th>Proposition</th>
<th>Truth Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>false $\Rightarrow$ false</td>
<td>true</td>
</tr>
<tr>
<td>false $\Rightarrow$ true</td>
<td>true</td>
</tr>
<tr>
<td>true $\Rightarrow$ false</td>
<td>false</td>
</tr>
<tr>
<td>true $\Rightarrow$ true</td>
<td>true</td>
</tr>
</tbody>
</table>

**Tautology.** A proposition is a tautology iff for every assignment of truth values (true, false) to the propositional variables $P_0, \ldots, P_n$, the truth value of the proposition is true.

**Question 2.1** (15 points).

Prove, using rule induction, that if $A \text{ thm}$ then $A$ is a tautology.

**Question 2.2** (EXTRA CREDIT).

Find a proposition $A$ that is a tautology, but not a theorem (that is, the judgment $A \text{ thm}$ cannot be derived). You do not need to prove that it is not a theorem!