Assignment 1: Grammars and Induction

15-312: Foundations of Programming Languages
Matt Moore (`mlmlm@cmu.edu`)

Out: Thursday, September 2, 2004
Due: Thursday, September 9, 2004 (1:30 pm)

50 points total

Welcome to 15-312! This assignment focuses on context-free grammars and inductive proofs. It is due September 9^{th} at the start of lecture. You are encouraged, but not required, to typeset your answers; if you write them out 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 over the alphabet $\Sigma = \{\texttt{int}, \texttt{list}, \texttt{->}, \texttt{(}, \texttt{)}\}$ with nonterminals tycon and type:

$$\begin{aligned} \mathsf{tycon} &::= \texttt{int} \mid \mathsf{tycon}\ \texttt{list} \mid \texttt{(}\ \mathsf{type}\ \texttt{)} \\ \mathsf{type} &::= \mathsf{tycon} \mid \mathsf{type}\ \texttt{->}\ \mathsf{type} \end{aligned}$$

Question 1.1 (5 points).

The first production for tycon can be written in rule notation as

$$\frac{}{\texttt{int}\ \mathsf{tycon}}$$

Write grammar G in rule notation.

Grammar G is flawed: it does not pin down the associativity of `->`. For example, there are two different derivations of the string `int->int->int`. We can fix the ambiguity by changing the second production of type from

$$\mathsf{type} ::= \mathsf{type}\ \texttt{->}\ \mathsf{type}$$

to

$$\mathsf{type} ::= \mathsf{tycon}\ \texttt{->}\ \mathsf{type}$$

Making this change results in the grammar G' below. To avoid confusion, we rename tycon to tycon′ and type to type′.

$$\mathsf{tycon}' ::= \texttt{int} \mid \mathsf{tycon}'\ \texttt{list} \mid (\ \mathsf{type}'\ )$$
$$\mathsf{type}' ::= \mathsf{tycon}' \mid \mathsf{tycon}'\ \texttt{->}\ \mathsf{type}'$$

Question 1.2 (5 points).

Write grammar G' in rule notation.

Note, however, that we have not shown that this new grammar G' is equivalent to G, that is, that the languages of type and type′ are the same: $L(\mathsf{type}) = L(\mathsf{type}')$. This can be proved in two steps: first prove $L(\mathsf{type}') \subseteq L(\mathsf{type})$, then prove $L(\mathsf{type}) \subseteq L(\mathsf{type}')$.

Question 1.3 (10 points).

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

If s type′ then s 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 $L(\mathsf{type}) \subseteq L(\mathsf{type}')$ by proving

If s type then s 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 nullary 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:

$$\frac{}{P_i \text{ prop}}\ \textit{Var} \qquad \frac{A \text{ prop} \quad B \text{ prop}}{(A \Rightarrow (B \Rightarrow A)) \text{ thm}}\ K$$

$$\frac{A \text{ prop} \quad B \text{ prop}}{(A \Rightarrow B) \text{ prop}}\ \textit{Imp} \qquad \frac{A \text{ prop} \quad B \text{ prop} \quad C \text{ prop}}{((A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))) \text{ thm}}\ S$$

$$\frac{(A \Rightarrow B) \text{ thm} \quad A \text{ thm}}{B \text{ thm}}\ \textit{App}$$

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 $\Rightarrow$:

Proposition	**Truth Value**
(`false` $\Rightarrow$ `false`)	`true`
(`false` $\Rightarrow$ `true`)	`true`
(`true` $\Rightarrow$ `false`)	`false`
(`true` $\Rightarrow$ `true`)	`true`

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 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 thm cannot be derived). You do not need to prove that it is not a theorem!