# PS2, Solutions

February 7, 2010

## Problem 1

The first rule simply takes a production which gives a terminal symbol, and an edge which is labeled with the corresponding symbol. It then "labels" the edge with the producing nonterminal:

$$\frac{X \to \alpha \quad n \xrightarrow{\alpha} m}{n \xrightarrow{X} m}$$

The next rule "builds up" paths from components of nonterminals:

$$\frac{X \to YZ \quad n \xrightarrow{Y} p \quad p \xrightarrow{Z} m}{n \xrightarrow{X} m}$$

Let $|V|$ and $|E|$ be the number of vertices and edges in the graph, respectively. Also let $n$ be the number of nonterminals in the grammar, and $|\mathcal{G}|$ the number of productions. In the first inference rule, the first intecedent may be instantiated in at most $n$ ways, while the second may in instantiated in at most $|E|$. In the second inference rule, the first antecedant may be instantiated in a most $|\mathcal{G}|$ ways, and the second and third (together) in at most $|V|^3$. Hence, there will be a total of at most $n|E| + |\mathcal{G}||V|^3$ rule firings over the course of a run of the algorithm, which gives the running time.

## Problem 2

We will first define a cut:

$$\begin{aligned}
\mathbb{R} \quad = \quad & (S : \text{setof}(\mathbb{Q}) \text{ s.t.} \\
& \exists (x : \mathbb{Q}) . (x \in_{\mathbb{Q}} S) \\
& \exists (x : \mathbb{Q}) . \forall (y : \mathbb{Q}) . ((y \in_{\mathbb{Q}} S) \Rightarrow (x <_{\mathbb{Q}} y)) \\
& \forall (x : \mathbb{Q}) . \forall (y : \mathbb{Q}) . ((x \in_{\mathbb{Q}} S) \wedge (x <_{\mathbb{Q}} y) \Rightarrow (y \in S)) \\
& \neg \exists (x : \mathbb{Q}) . \forall (y : \mathbb{Q}) . ((x \in_{\mathbb{Q}} S) \wedge (y \in S) \Rightarrow (x \leq_{\mathbb{Q}} y)))
\end{aligned}$$

The four claims above assert that a cut is nonempty, bounded below, contains all elements of $\mathbb{Q}$ past a certain point, and has no minimal element, respectively. To define 0 and 1, we'll define a MakeReal function:

$$\text{MakeReal} \quad = \quad \lambda (q : \mathbb{Q}) . \text{the} (x : \mathbb{R} \ (\forall y : \mathbb{Q} . (((y >_{\mathbb{Q}} q) \Rightarrow (y \in_{\mathbb{Q}} x)) \wedge ((y \leq_{\mathbb{Q}} q) \Rightarrow \neg (y \in_{\mathbb{Q}} x)))))$$

the definitions of 0 and 1 are then just applications of MakeReal on the corresponding definitions on the rationals:

$$\begin{aligned}
0_{\mathbb{R}} \quad &= \quad \text{MakeReal}\, 0_{\mathbb{Q}} \\
1_{\mathbb{R}} \quad &= \quad \text{MakeReal}\, 1_{\mathbb{Q}}
\end{aligned}$$

It will also be useful to define a comparison operator, which we note by the definition of a Dedekind cut may be defined such that $\leq$ is exactly the superset relation $\supseteq$:

$$\leq_{\mathbb{R}} \quad = \quad \lambda\left(X:\mathbb{R}\right).\lambda\left(Y:\mathbb{R}\right).\left(\forall\left(z:\mathbb{Q}\right).\left(\left(z\in_{\mathbb{Q}}X\right)\Rightarrow\left(z\in_{\mathbb{Q}}Y\right)\right)\right)$$

From this definition, we may easily define $<_{\mathbb{R}}$, $\geq_{\mathbb{R}}$ and $>_{\mathbb{R}}$, which I won't write out. We will define $+$ as the cut which contains all sums of all elements of the parameters, but no sums of rationals less than the parameters:

$$\begin{aligned}+_{\mathbb{R}} \quad = \quad & \lambda\left(X:\mathbb{R}\right).\lambda\left(Y:\mathbb{R}\right).\mathrm{the}\left(Z:\mathbb{R}\right.\\ & \left(\forall\left(x:\mathbb{Q}\right).\forall\left(y:\mathbb{Q}\right).\left(\left(x\in_{\mathbb{Q}}X\right)\wedge\left(y\in_{\mathbb{Q}}Y\right)\Rightarrow\left(x+_{\mathbb{Q}}y\in_{\mathbb{Q}}Z\right)\right)\right)\\ & \wedge\left(\forall\left(x:\mathbb{Q}\right).\forall\left(y:\mathbb{Q}\right).\left(\neg\left(x\in_{\mathbb{Q}}X\right)\wedge\neg\left(y\in_{\mathbb{Q}}Y\right)\Rightarrow\neg\left(x+_{\mathbb{Q}}y\in_{\mathbb{Q}}Z\right)\right)\right)\end{aligned}$$

If we define subtypes for the nonnegative rationals and reals:

$$\begin{aligned}\mathbb{Q}^{+} \quad &= \quad \left(x:\mathbb{Q}\,\mathrm{s.t.}\,0_{\mathbb{Q}}\leq_{\mathbb{Q}}x\right)\\ \mathbb{R}^{+} \quad &= \quad \left(x:\mathbb{R}\,\mathrm{s.t.}\,0_{\mathbb{R}}\leq_{\mathbb{R}}x\right)\end{aligned}$$

then we may define multiplication on the positive reals in the same was as addition was defined:

$$\begin{aligned}\times_{\mathbb{R}^{+}} \quad = \quad & \lambda\left(X:\mathbb{R}^{+}\right).\lambda\left(Y:\mathbb{R}^{+}\right).\mathrm{the}\left(Z:\mathbb{R}^{+}\right.\\ & \left(\forall\left(x:\mathbb{Q}^{+}\right).\forall\left(y:\mathbb{Q}^{+}\right).\left(\left(x\in_{\mathbb{Q}}X\right)\wedge\left(y\in_{\mathbb{Q}}Y\right)\Rightarrow\left(x\times_{\mathbb{Q}}y\in_{\mathbb{Q}}Z\right)\right)\right)\\ & \wedge\left(\forall\left(x:\mathbb{Q}^{+}\right).\forall\left(y:\mathbb{Q}^{+}\right).\left(\neg\left(x\in_{\mathbb{Q}}X\right)\wedge\neg\left(y\in_{\mathbb{Q}}Y\right)\Rightarrow\neg\left(x\times_{\mathbb{Q}}y\in_{\mathbb{Q}}Z\right)\right)\right)\right)\end{aligned}$$

The final ingredient which we need is negation. Note that this definition is different than the one which was given in class, since we must ensure that the resulting cut is left-open. This is the reason for using MakReal and the comparison operators, rather than $\in$:

$$\begin{aligned}-_{\mathbb{R}} \quad = \quad & \lambda\left(X:\mathbb{R}\right).\mathrm{the}\left(Y:\mathbb{R}\right.\\ & \forall\left(x:\mathbb{Q}\right).\left(\left(\mathrm{MakeReal}\,x>_{\mathbb{R}}X\right)\Rightarrow\left(-_{\mathbb{Q}}x\notin_{\mathbb{Q}}Y\right)\right)\\ & \forall\left(x:\mathbb{Q}\right).\left(\left(\mathrm{MakeReal}\,x<_{\mathbb{R}}X\right)\Rightarrow\left(-_{\mathbb{Q}}x\in_{\mathbb{Q}}Y\right)\right)\end{aligned}$$

We now define multiplication on the reals as:

$$\begin{aligned}\times_{\mathbb{R}} \quad = \quad & \lambda\left(X:\mathbb{R}\right).\lambda\left(Y:\mathbb{R}\right).\mathrm{the}\left(Z:\mathbb{R}\right.\\ & \left(X\geq_{\mathbb{R}}0\right)\wedge\left(Y\geq_{\mathbb{R}}0\right)\Rightarrow\left(Z=X\times_{\mathbb{R}^{+}}Y\right)\\ & \left(X\geq_{\mathbb{R}}0\right)\wedge\left(Y<_{\mathbb{R}}0\right)\Rightarrow\left(Z=-_{\mathbb{R}}\left(X\times_{\mathbb{R}^{+}}\left(-_{\mathbb{R}}Y\right)\right)\right)\\ & \left(X<_{\mathbb{R}}0\right)\wedge\left(Y<_{\mathbb{R}}0\right)\Rightarrow\left(Z=\left(\left(-_{\mathbb{R}}X\right)\times_{\mathbb{R}^{+}}\left(-_{\mathbb{R}}Y\right)\right)\right)\\ & \left(X<_{\mathbb{R}}0\right)\wedge\left(Y\geq_{\mathbb{R}}0\right)\Rightarrow\left(Z=-_{\mathbb{R}}\left(\left(-_{\mathbb{R}}X\right)\times_{\mathbb{R}^{+}}Y\right)\right)\end{aligned}$$

# Problem 3

We will define a predicate on structure types of the form MaybeNat $=\{N:\mathrm{type},\mathit{zero}:N,\mathit{successor}:N\rightarrow N\}$, which asserts that a well order exists on this type, which honors the *zero* and *successor* definitions:

$$\begin{aligned}\mathrm{WellOrdered} \quad = \quad & \lambda\left(T:\mathrm{MaybeNat}\right).\lambda\leq:\left(T.N\times T.N\rightarrow\mathrm{boole}\right).\\ & \forall\left(x:T.N\right).\left(T.\mathit{zero}\leq x\right)\\ & \wedge\forall\left(x:T.N\right).\left(\neg\left(x=T.\mathit{successor}\left(x\right)\right)\wedge\left(x\leq T.\mathit{successor}\left(x\right)\right)\right)\\ & \wedge\forall\left(x:T.N\right).\forall\left(y:T.N\right).\left(\left(x\leq y\right)\wedge\left(y\leq x\right)\Rightarrow\left(x=y\right)\right)\\ & \wedge\forall\left(x:T.N\right).\forall\left(y:T.N\right).\forall\left(z:T.N\right).\left(\left(x\leq y\right)\wedge\left(y\leq z\right)\Rightarrow\left(x\leq z\right)\right)\\ & \wedge\forall\left(x:T.N\right).\forall\left(y:T.N\right).\forall\left(z:T.N\right).\left(\left(x\leq y\right)\vee\left(y\leq x\right)\right)\end{aligned}$$

The first two portions of the above claim that *zero* is a minimal element, and that every element less than than its successor. The remainder claim that the order is a total order (`http://en.wikipedia.org/wiki/Total_order`): antisymmetric, transitive, and total.

A MaybeNat which satisfies the WellOrdered predicate will not necessarily be isomorphic to the natural numbers: we also need to claim that there are an infinite number of natural numbers, and that all of them can be reached by performing a finite number of *successor* operations from *zero*. The first claim, that there are an infinite number, is in fact implied by the WellOrdered predicate, but there is no harm in stating it anyway–the following asserts that there is no largest element:

$$\text{Infinite} \quad = \quad \lambda\left(T : \text{MaybeNat}\right).\lambda \leq : \left(T.N \times T.N \to \text{boole}\right).\left(\neg\exists\left(x : T.N\right).\left(\forall\left(y : T.N\right).\left(y \leq x\right)\right)\right)$$

The last claim will be asserted somewhat more subtly. We will claim that there exists a *unique* predicate $P$ such that $P\left(zero\right) = \text{true}$, and $P\left(successor\, x\right) = P\left(x\right)$. Essentially, this predicate will inductively mark all of the natural numbers (*zero*, and those numbers which can be reached via a finite number of successor opertions) true, but can take on different values on elements which are "left over". Claiming that there is a *unique* such predicate ensures that there are no left-over elements, so that all we have are the naturals:

$$\begin{aligned}\text{Countable} \quad = \quad & \lambda\left(T : \text{MaybeNat}\right).\left(\exists\left(P : T.N \to \text{boole}\right).\right. \\ & P\left(T.zero\right) \wedge \forall\left(x : T.N\right).\left(P\left(T.successor\, x\right) = P\left(x\right)\right) \wedge \\ & \forall\left(Q : T.N \to \text{boole}\right). \\ & \left.Q\left(T.zero\right) \wedge \forall\left(x : T.N\right).\left(Q\left(T.successor\, x\right) = Q\left(x\right)\right) \Rightarrow P = Q\right)\end{aligned}$$

We may now define a predicate which asserts that an element of the MaybeNat type follows the axioms of the natural numbers:

$$\begin{aligned}\text{IsNatural} \quad = \quad & \lambda\left(T : \text{MaybeNat}\right).\exists\left(\leq : T.N \times T.N \to \text{boole}\right). \\ & \text{WellOrdered}\left(T, \leq\right) \\ & \wedge\text{Infinite}\left(T, \leq\right) \\ & \wedge\text{Countable}\left(T\right)\end{aligned}$$

Proving that there is an isomorphism between any two types which satisfy the IsNatural predicate is now straightforward. Since we know that, by starting from *zero*, and performing the *successor* operation a finite number of times, we will always visit distinct elements, and will eventually visit all elements, we may identify $successor\, zero$ with 1, $successor\, successor\, zero$ with 2, and so on. The isomorphism is to, for each $n \in \mathbb{N}$, identify the $n$ in one type with the corresponding $n$ in the other.