15-399 Constructive Logic

Midterm I

Model Solution

October 5, 2000

Name: ______________________________

Andrew ID: ______________________________

- This is a closed-book exam; only 1 two-sided sheet of notes is permitted.
- Write your answer legibly in the space provided.
- There are 12 pages in this exam, including 3 worksheets.
- It consists of 4 questions worth a total of 200 points, plus one question for 40 points extra credit.
- Extra credit is recorded separately, so make sure your answers to question 1–4 are correct before attempting to solve the extra credit question.
- You have 80 minutes for this exam.

Problem 1	**Problem 2**	**Problem 3**	**Problem 4**	**Total**	**EC**
60	**50**	**40**	**50**	**200**	**40**

1. Proofs and Proof Terms (60 pts)

For each of the following, give a constructive proof in natural deduction and a proof term. You may use a proof tree representation, or a linear form. If you choose the latter, you need to justify each line by the name of an inference rule so we can easily verify your reasoning.

1. (15 pts) Proof of $(A \supset B) \supset (A \supset (B \wedge A))$

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{A \supset B}^{\,u} \qquad \overline{A}^{\,w}}{B}\supset E \qquad \overline{A}^{\,w}}{B \wedge A}\wedge I}{A \supset (B \wedge A)}\supset I^{w}}{(A \supset B) \supset (A \supset (B \wedge A))}\supset I^{u}$$

2. (15 pts) Proof term of $(A \supset B) \supset (A \supset (B \wedge A))$

$$\lambda u{:}A \supset B.\ \lambda w{:}A.\ \langle u\, w, w \rangle$$

3. (15 pts) Proof of $((A \lor B) \land \neg B) \supset A$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(A \lor B) \land \neg B}\,u}{A \lor B}\,\land E_L \qquad \dfrac{}{A}\,v \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{(A \lor B) \land \neg B}\,u}{\neg B}\,\land E_L \qquad \dfrac{}{B}\,w}{\bot}\,\supset E}{A}\,\bot E}{A}\,\lor E^{v,w}}{((A \lor B) \land \neg B) \supset A}\,\supset I^u$$

4. (15 pts) Proof term of $((A \lor B) \land \neg B) \supset A$

$$\begin{aligned} \lambda u{:}(A \lor B) \land \neg B.\ \mathbf{case}\ & \mathbf{fst}\, u \\ \mathbf{of}\ & \mathbf{inl}(v) \Rightarrow v \\ \mid\ & \mathbf{inr}(w) \Rightarrow \mathbf{abort}((\mathbf{snd}\, u)\, w) \end{aligned}$$

2. Derived Rules (50 pts)

1. (10 pts) What is a *derived rule of inference*? Explain the concept in 1–3 sentences.

   A derived rule of inference is an evident hypothetical judgment. The evidence is given by a hypothetical derivation of the conclusion of the derived rule from its premises.

2. (20 pts) Show that

$$\frac{A \supset (B \supset C)\ \mathit{true} \qquad A \supset B\ \mathit{true}}{A \supset C\ \mathit{true}}$$

   is a valid derived rule of inference.

$$\dfrac{\dfrac{\dfrac{A \supset (B \supset C) \qquad \dfrac{}{A}\,u}{B \supset C}\supset E \qquad \dfrac{A \supset B \qquad \dfrac{}{A}\,u}{B}\supset E}{C}\supset E}{A \supset C}\supset I^u$$

3. (20 pts) Prove that

$$\frac{A \vee B\ \mathit{true}}{A\ \mathit{true}}$$

is not a valid derived rule of inference.

Assume the rule above were a valid derived rule of inference. Then

$$\dfrac{\dfrac{\dfrac{}{\top\ \mathit{true}}\ \top I}{\bot \vee \top\ \mathit{true}}\ \vee I_R}{\bot\ \mathit{true}}$$

would be a correct derivation of $\bot$ *true*. But this is impossible, since we know that $\bot$ *true* does not have a normal proof and therefore cannot have a proof.

3. Primitive Recursion over Natural Numbers (40 pts)

In this problem we consider a function

$$\mathit{upto} \in \mathbf{nat} \to \mathbf{nat\,list}$$

such that $\mathit{upto}(\mathbf{0}) = \mathbf{nil}$ and $\mathit{upto}(\mathbf{s}(n)) = \mathbf{0} :: \mathbf{s}(\mathbf{0}) :: \cdots :: n :: \mathbf{nil}$.

1. (20 pts) Give a specification of *upto* suitable for implementation by primitive recursion. [**Hint:** use an auxiliary function with an accumulator argument.]

$$\begin{array}{rcl} \mathit{upto}' & : & \mathbf{nat} \to \mathbf{nat\,list} \to \mathbf{nat\,list} \\ \mathit{upto}'\; 0\; l & = & l \\ \mathit{upto}'\; (\mathbf{s}(n))\; l & = & \mathit{upto}'\; n\; (n :: l) \\ \mathit{upto}\; n & = & \mathit{upto}'\; n\; \mathbf{nil} \end{array}$$

2. (20 pts) Give the implementation of *upto* as a primitive recursion.

$$\begin{array}{rcl} \mathit{upto}' & = & \lambda n{\in}\mathbf{nat}.\; \mathbf{rec}\; n \\ & & \quad \mathbf{of}\; f(\mathbf{0}) \Rightarrow \lambda l.\, l \\ & & \quad \mid\; f(\mathbf{s}(n)) \Rightarrow \lambda l.\, f(n)\; (n :: l) \\ \mathit{upto} & = & \lambda n{\in}\mathbf{nat}.\; \mathit{upto}'\; n\; \mathbf{nil} \end{array}$$

4. Data Types (50 pts)

In this problem we explore the representation of integers according to the following scheme:

- Any positive integer i is represented by $\mathbf{pos}(i)$.
- Zero is represented by $\mathbf{pos}(\mathbf{0})$.
- Any negative integer i is represented by $\mathbf{neg}(-i-1)$.

So the type **int** has two constructors **pos** and **neg**, both taking a natural number as an argument. For example, the integer 3 is represented by $\mathbf{pos}(\mathbf{s}(\mathbf{s}(\mathbf{s}(\mathbf{0}))))$, the integer -1 by $\mathbf{neg}(\mathbf{0})$. The reason for the subtracting 1 is to ensure that every integer has a *unique* representation (without it, $\mathbf{pos}(\mathbf{0})$ and $\mathbf{neg}(\mathbf{0})$ would both represent zero).

1. (10 pts) Show the formation rule for type **int**.

$$\frac{}{\mathbf{int}\ \mathit{type}}\ \mathbf{int}F$$

2. (10 pts) Give the two introduction rules for elements of type **int**.

$$\frac{\Gamma \vdash n \in \mathbf{nat}}{\Gamma \vdash \mathbf{pos}(n) \in \mathbf{int}}\ \mathbf{int}I_p \qquad \frac{\Gamma \vdash m \in \mathbf{nat}}{\Gamma \vdash \mathbf{neg}(m) \in \mathbf{int}}\ \mathbf{int}I_n$$

3. (10 pts) Give the elimination rule for elements of type **int**. This should take the form of a primitive recursion or case operator on integers.

$$\frac{\Gamma \vdash i \in \mathbf{int} \qquad \Gamma, n{\in}\mathbf{nat} \vdash t \in \tau \qquad \Gamma, m{\in}\mathbf{nat} \vdash s \in \tau}{\Gamma \vdash (\mathbf{case}\ i\ \mathbf{of}\ \mathbf{pos}(n) \Rightarrow t \mid \mathbf{neg}(m) \Rightarrow s) \in \tau}\ \mathbf{int}E$$

4. (10 pts) We define the increment function on integers as $inc(i) = i + 1$. Give a specification of inc using the representation above.

$$\begin{array}{rcl} inc(\mathbf{pos}(n)) & = & \mathbf{pos}(\mathbf{s}(n)) \\ inc(\mathbf{neg}(\mathbf{0})) & = & \mathbf{pos}(\mathbf{0}) \\ inc(\mathbf{neg}(\mathbf{s}(n))) & = & \mathbf{neg}(n) \end{array}$$

5. (10 pts) Define the inc function using your primitive recursion or case operator over integers. You may freely use primitive recursion over natural numbers (type **nat**) and standard functions on natural numbers such as predecessor or addition.

$$\begin{array}{l} inc = \lambda i{\in}\mathbf{nat}.\ \mathbf{case}\ i \\ \qquad\qquad \mathbf{of}\ \mathbf{pos}(n) \Rightarrow \mathbf{pos}(\mathbf{s}(n)) \\ \qquad\qquad\ \mid \mathbf{neg}(m) \Rightarrow\ \mathbf{rec}\ m \\ \qquad\qquad\qquad\qquad \mathbf{of}\ f(\mathbf{0}) \Rightarrow \mathbf{pos}(\mathbf{0}) \\ \qquad\qquad\qquad\qquad\ \mid f(\mathbf{s}(m')) \Rightarrow \mathbf{neg}(m') \end{array}$$

5. Primitive Recursion over Lists (40 pts extra credit)

Consider the following specification:

$$\begin{array}{rcl} \textit{mapc}\ f\ \mathbf{nil} & = & \mathbf{nil} \\ \textit{mapc}\ f\ (x :: l) & = & (f\ x)\ :: (\textit{mapc}\ (\lambda x \in \mathbf{nat}.\ f\ (f\ x))\ l) \end{array}$$

1. (10 pts) What is the normal form of $\textit{mapc}\ (\lambda x{\in}\mathbf{nat}.\ \mathbf{s}(x))\ (\mathbf{0} :: \mathbf{0} :: \mathbf{nil})$?

$$\mathbf{s}(\mathbf{0}) :: \mathbf{s}(\mathbf{s}(\mathbf{0})) :: \mathbf{nil}$$

2. (10 pts) Give the type of *mapc*.

$$\textit{mapc} : (\mathbf{nat} \to \mathbf{nat}) \to \mathbf{nat\ list} \to \mathbf{nat\ list}$$

3. (20 pts) Give the implementation of *mapc* as a primitive recursion function.

$$\begin{array}{rcl} \textit{mapc}' & = & \lambda l.\ \mathbf{rec}\ l \\ & & \quad \mathbf{of}\ r(\mathbf{nil}) \Rightarrow \lambda f.\ \mathbf{nil} \\ & & \quad \ \mid\ r(x :: l') \Rightarrow \lambda f.\ f(x) :: r(l')\,(\lambda x{\in}\mathbf{nat}.\ f(f(x))) \\ \textit{mapc} & = & \lambda f.\ \lambda l.\ \textit{mapc}'\ l\ f \end{array}$$