15-399 Constructive Logic

Midterm II

Model Solution

November 9, 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.
- **Unless otherwise indicated, proofs should be informal, but rigorous and detailed. Clearly state the induction principle you use, the cases you distinguish, and the reasoning in each case.**

Problem 1	**Problem 2**	**Problem 3**	**Problem 4**	**Total**	**EC**
60	**30**	**60**	**50**	**200**	**40**

1. First-Order Logic (60 pts)

For each of the following, give a formal 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) Formal proof of $((\exists x.\ A(x)) \supset B) \supset \forall x.\ (A(x) \supset B)$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(\exists x.\ A(x)) \supset B}\,u \qquad \dfrac{\dfrac{}{c \in \tau}\,c \quad \dfrac{}{A(c)}\,w}{\exists x.\ A(x)}\,\exists I}{B}\,\supset E}{A(c) \supset B}\,\supset I^{w}}{\forall x.\ (A(x) \supset B)}\,\forall I^{c}}{((\exists x.\ A(x)) \supset B) \supset \forall x.\ (A(x) \supset B)}\,\supset I^{u}$$

2. (15 pts) Proof term for $((\exists x.\ A(x)) \supset B) \supset \forall x.\ (A(x) \supset B)$

$$\lambda u.\ \lambda c.\ \lambda w.\ u\ \langle c, w \rangle$$

3. (15 pts) Formal proof of $(\forall x.\ (A(x) \supset B)) \supset ((\exists x.\ A(x)) \supset B)$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\forall x.\ (A(x) \supset B)}\,u \quad \dfrac{}{c \in \tau}\,c}{A(c) \supset B}\,\forall E \qquad \dfrac{}{A(c)}\,w}{B}\,\supset E \qquad \dfrac{}{\exists x.\ A(x)}\,v}{B}\,\exists E^{c,w}}{(\exists x.\ A(x)) \supset B}\,\supset I^{v}}{(\forall x.\ (A(x) \supset B)) \supset ((\exists x.\ A(x)) \supset B)}\,\supset I^{u}$$

4. (15 pts) Proof term for $(\forall x.\ (A(x) \supset B)) \supset ((\exists x.\ A(x)) \supset B)$

$$\lambda u.\ \lambda v.\ \mathbf{let}\ \langle c, w\rangle = v\ \mathbf{in}\ (u\ c)\ w$$

2. Arithmetic (30 pts)

1. (30 pts) Prove that

$$\forall x{\in}\mathbf{nat}.\ \forall y{\in}\mathbf{nat}.\ x < y \vee x =_N y \vee y < x.$$

The proof is by induction on x, considering subcases on y. We explicitly show which disjunct is true in each case, but elide the disjunction introductions.

Case: $x = \mathbf{0}$. Then we distinguish cases on y.

Case: $y = \mathbf{0}$. Then $x =_N y$ by rule $=_N I_0$.

Case: $y = \mathbf{s}(y')$. Then $x < y$ by rule $<I_0$.

Case: $x = \mathbf{s}(x')$. Then we distinguish cases on y.

Case: $y = \mathbf{0}$. Then $y < x$ by rule $<I_0$.

Case: $y = \mathbf{s}(y')$. Then, by induction hypothesis,

$$x' < y' \vee x' =_N y' \vee y' < x'.$$

We distinguish the three cases.

Subcase: $x' < y'$. Then $x < y$ by rule $<I_s$.

Subcase: $x' =_N y'$. Then $x =_N y$ by rule $=_N I_s$.

Subcase: $y' < x'$. Then $y < x$ by rule $<I_s$.

3. Computational Contents (60 pts)

Below is a proof term verifying the following decomposition lemma.

$$\forall l{\in}\tau\,\mathbf{list}.\ \exists k{\in}\tau\,\mathbf{list}.\ l =_L \mathbf{nil} \vee \exists y{\in}\tau.\ l =_L y :: k$$

In this proof term we use *refl* for the reflexivity lemma on lists.

$$\begin{array}{rcl} \mathit{dec} & = & \lambda l{\in}\tau\,\mathbf{list}.\,\mathbf{rec}\ l \\ & & \quad \mathbf{of}\ f(\mathbf{nil}) \Rightarrow \langle \mathbf{nil}, \mathbf{inl}\,(\mathit{refl}\ \mathbf{nil})\rangle \\ & & \quad \mid\ f(x :: l') \Rightarrow \langle l', \mathbf{inr}\,\langle x, \mathit{refl}\ (x :: l')\rangle\rangle \end{array}$$

1. (5 pts) Clearly annotate the proposition

   $$\forall\, l \in \tau\,\mathbf{list}.\ \exists\, k \in \tau\,\mathbf{list}.\ l =_L \mathbf{nil}\ \vee\ \exists\, y \in \tau.\ l =_L y :: k$$

   so that the extracted function returns the tail of a non-empty list.

   $$\forall\, l \in \tau\,\mathbf{list}.\ \exists\, k \in \tau\,\mathbf{list}.\ [l =_L \mathbf{nil}\ \vee\ \exists\, y \in \tau.\ l =_L y :: k]$$

2. (10 pts) Annotate the proof term correspondingly

   $$\begin{array}{rcl} \mathit{dec} & = & \lambda\, l \in \tau\,\mathbf{list}.\ \mathbf{rec}\ l \\ & & \quad \mathbf{of}\ f\ (\mathbf{nil})\ \Rightarrow\ \langle\ \mathbf{nil}\ ,\ \mathbf{inl}\ (\mathit{refl}\ \mathbf{nil})\ \rangle \\ & & \quad \mid\ f\ (x :: l')\ \Rightarrow\ \langle\ l'\ ,\ \mathbf{inr}\ \langle\ x\ ,\ \mathit{refl}\ (x :: l')\ \rangle\ \rangle \end{array}$$

   $$\begin{array}{rcl} \mathit{dec} & = & \lambda\, l \in \tau\,\mathbf{list}.\ \mathbf{rec}\ l \\ & & \quad \mathbf{of}\ f\ (\mathbf{nil})\ \Rightarrow\ \langle\ \mathbf{nil}\ ,\ [\mathbf{inl}\ (\mathit{refl}\ \mathbf{nil})]\ \rangle \\ & & \quad \mid\ f\ (x :: l')\ \Rightarrow\ \langle\ l'\ ,\ [\mathbf{inr}\ \langle\ x\ ,\ \mathit{refl}\ (x :: l')\ \rangle]\ \rangle \end{array}$$

3. (5 pts) Give the type of the simplified extracted function.

   $$\mathit{dec}' : \tau\,\mathbf{list} \to \tau\,\mathbf{list}$$

4. (10 pts) Give the definition of the simplified extracted function.

   $$\begin{array}{rcl} \mathit{dec}' & = & \lambda\, l \in \tau\,\mathbf{list}.\ \mathbf{rec}\ l \\ & & \quad \mathbf{of}\ f\ (\mathbf{nil})\ \Rightarrow\ \mathbf{nil} \\ & & \quad \mid\ f\ (x :: l')\ \Rightarrow\ l' \end{array}$$

5. (5 pts) Clearly annotate the proposition

$$\forall\, l \in \tau\,\mathbf{list}.\ \exists\, k \in \tau\,\mathbf{list}.\ l =_L \mathbf{nil} \ \vee\ \exists\, y \in \tau.\ l =_L y :: k$$

so that the extracted function returns either the head of l or an indication that the input list l is empty.

$$\forall\, l \in \tau\,\mathbf{list}.\ \exists\, [k \in \tau\,\mathbf{list}].\ [l =_L \mathbf{nil}] \ \vee\ \exists\, y \in \tau.\ [l =_L y :: k]$$

6. (10 pts) Annotate the proof term correspondingly

$$\begin{array}{rcl} \mathit{dec} & = & \lambda\, l \in \tau\,\mathbf{list}.\ \mathbf{rec}\ l \\ & & \qquad \mathbf{of}\ f\ (\mathbf{nil}) \ \Rightarrow\ \langle\, \mathbf{nil}\ ,\ \mathbf{inl}\ (\mathit{refl}\ \mathbf{nil})\, \rangle \\ & & \qquad \mid\ f\ (x :: l') \ \Rightarrow\ \langle\, l'\ ,\ \mathbf{inr}\ \langle\, x\ ,\ \mathit{refl}\ (x :: l')\, \rangle\, \rangle \end{array}$$

$$\begin{array}{rcl} \mathit{dec} & = & \lambda\, l \in \tau\,\mathbf{list}.\ \mathbf{rec}\ l \\ & & \qquad \mathbf{of}\ f\ (\mathbf{nil}) \ \Rightarrow\ \langle\, [\mathbf{nil}]\ ,\ \mathbf{inl}\ [(\mathit{refl}\ \mathbf{nil})]\, \rangle \\ & & \qquad \mid\ f\ (x :: l') \ \Rightarrow\ \langle\, [l']\ ,\ \mathbf{inr}\ \langle\, x\ ,\ [\mathit{refl}\ (x :: l')\, \rangle]\, \rangle \end{array}$$

7. (5 pts) Give the type of the simplified extracted function.

$$\mathit{dec}' : \tau\,\mathbf{list} \to 1 + \tau$$

8. (10 pts) Give the definition of the simplified extracted function.

$$\begin{array}{rcl} \mathit{dec} & = & \lambda\, l \in \tau\,\mathbf{list}.\ \mathbf{rec}\ l \\ & & \qquad \mathbf{of}\ f\ (\mathbf{nil}) \ \Rightarrow\ \mathbf{inl}\ \langle\, \rangle \\ & & \qquad \mid\ f\ (x :: l') \ \Rightarrow\ \mathbf{inr}\ x \end{array}$$

4. Induction (50 points)

We specify

$$\begin{array}{rcl} sq & \in & \mathbf{nat} \to \mathbf{nat} \to \mathbf{nat} \\ sq\ \mathbf{0}\ a & = & a \\ sq\ (\mathbf{s}(n'))\ a & = & sq\ n'\ (\mathbf{s}(a + n' + n')) \end{array}$$

$$\begin{array}{rcl} square & \in & \mathbf{nat} \to \mathbf{nat} \\ square\ n & = & sq\ n\ \mathbf{0} \end{array}$$

We do not write out the corresponding straightforward implementations using primitive recursion.

We would like to prove that

$$\forall n{\in}\mathbf{nat}.\ square\ n =_N n \times n$$

where $x \times y$ is ordinary multiplication. For this problem, you may assume any properties of addition $x+y$ and multiplication $x \times y$ such as commutativity, associativity, and distributivity as needed.

1. (20 pts) Generalize the induction hypothesis to a property of sq.

$$\forall n{\in}\mathbf{nat}.\ \forall a{\in}\mathbf{nat}.\ sq\ n\ a =_N n \times n + a$$

2. (10 pts) Prove that your generalized property of sq implies the desired property of *square*

   The proof is direct. Let $n{\in}\mathbf{nat}$. Then

   $square\ n$
   $= sq\ n\ \mathbf{0}$
   $=_N n \times n + \mathbf{0}$ — By generalized property
   $=_N n \times n$ — By property of addition

3. (20 pts) Prove your generalized property of sq by an induction argument. As mentioned in the problem statement, you may freely use properties of addition and multiplication, but you must make your steps explicit.

The proof is by induction on n.

Case: $n = \mathbf{0}$. Then

$$sq\ \mathbf{0}\ a \Longrightarrow a =_N \mathbf{0} \times \mathbf{0} + a$$

Case: $n = \mathbf{s}(n')$. Then

$sq\ (\mathbf{s}(n'))\ a$
$\Longrightarrow sq\ n'\ (\mathbf{s}(a + n' + n'))$
$=_N (n' \times n' + (\mathbf{s}(a + n' + n')))$ By ind. hyp.
$=_N \mathbf{s}(n') \times \mathbf{s}(n') + a$ By properties of $+$, $\times$
$= n \times n + a.$

5. More Induction (40 points extra credit)

We specify the new primitive recursive helper function sq_2

$$\begin{array}{rcl} sq_2 & \in & \mathbf{nat} \to \mathbf{nat} \to \mathbf{nat} \to \mathbf{nat} \\ sq_2\ \mathbf{0}\ d\ a & = & a \\ sq_2\ (\mathbf{s}(n'))\ d\ a & = & sq_2\ n'\ (\mathbf{s}(\mathbf{s}(d)))\ (\mathbf{s}(a+d)) \end{array}$$

1. (10 pts) Define a (non-recursive) function *square* that calls sq_2 with appropriate arguments.

2. (15 pts) Generalize the induction hypothesis by giving a property of sq_2 that implies the correctness of your implementation of *square*.

3. (15 pts) Prove inductively that sq_2 satisfies your property.