Final Exam

15-317: Constructive Logic

December 17, 2012

Name: Andrew ID:

Instructions

- This exam is open notes, open book, and closed Internet. The last page of the exam recaps some rules you may find useful.
- The exam consists of fifteen pages in total.
- There are five problems. Not all problems are the same size or difficulty, so it may help to read through the whole exam first. You have three hours to complete the exam.
- When writing derivations, remember to label each inference with the rule used and any variables or parameters discharged (e.g., $\supset I^u$).
- You may find it helpful to construct your proofs on scratch paper (such as the back of a page) before writing it clearly in the space provided.
- Good luck!

	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5	**Total**
Score						
Max	50	50	50	50	50	250

1 Pawns' Passage

The game In this problem you will use linear logic to model a simple two-player game called Pawns' Passage. The players (white and black) sit on opposite sides of a board that is eight squares long and one square wide. The square nearest the white player is square 0, and the square nearest the black player is square 7:

white | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | black

Thus, "forward" means "toward higher numbers" for white, and "toward lower numbers" for black. "Backward" means the opposite.

Each square may hold a single pawn (either white or black), or it may be empty. The players alternate turns. During a player's turn, he may make one of two actions:

- Push one his pawns forward, provided the next square does not hold one of his own pawns. If the next square contains one of the opposing player's pawns, the opposing pawn is removed from the game.
- Place a new pawn in a square immediately behind one of his pawns, provided that square is empty.

For example, suppose the board looks like:

white | 0 | ∘ | 2 | 3 | ∘ | ∘ | • | 7 | black

If it is white's turn, he may move from 1 to 2, or move from 5 to 6 (capturing the black pawn in 6), or place a white pawn in 0 or 3. If it is black's turn, he may move from 6 to 5 (capturing the white pawn in 5), or place a black pawn in 7.

The model In our model, we will ignore the length of the board and number the squares with arbitrary integers. We describe the state of the world with the following propositions:

`whiteturn`	it is white's turn
`blackturn`	it is black's turn
`whitepawn`(i)	square i contains a white pawn
`blackpawn`(i)	square i contains a black pawn
`empty`(i)	square i is empty

Problem 1: Write linear logic rules expressing each of the possible state transitions. (Hint: you should have six rules altogether.)

Problem 2: In the board's initial state, white has a pawn at 0, black has a pawn at 7, and the remaining squares are empty:

white	∘	1	2	3	4	5	6	•	black

Give a linear logic formula that expresses the state of the world when it is white's turn and the board is in its initial state.

Problem 3: In our desired final state, white has a pawn in square 7, and we don't care about anything else. Express the final state as a linear logic formula.

Problem 4: Give a linear logic formula that says that we can reach the desired final state from an initial state in which: (1) the board is in its initial state, and (2) we get to pick who goes first.

Problem 5: Give a linear logic formula that expresses the same thing, **except** we do not get to pick who goes first. Instead, the choice is made for us.

2 Cut Elimination

Recall the unusual logical connective $A \mathbin{\odot} B$ with the sequent-calculus rules:

$$\frac{\Gamma \longrightarrow A \quad \Gamma, B \longrightarrow \bot}{\Gamma \longrightarrow A \mathbin{\odot} B} \mathbin{\odot} R \qquad \frac{\Gamma, A \mathbin{\odot} B, A \longrightarrow C}{\Gamma, A \mathbin{\odot} B \longrightarrow C} \mathbin{\odot} L1 \qquad \frac{\Gamma, A \mathbin{\odot} B, A \longrightarrow B}{\Gamma, A \mathbin{\odot} B \longrightarrow C} \mathbin{\odot} L2$$

We wish to extend the cut elimination theorem to include the happy-face connective. Suppose $\mathcal{D}$ is a derivation of $\Gamma \longrightarrow D$, and $\mathcal{E}$ is a derivation of $\Gamma, D \longrightarrow E$. Prove the cut elimination theorem for the cases in which:

- the last rule of $\mathcal{D}$ is $\odot R$ and the last rule of $\mathcal{E}$ is $\odot L1$,
- the last rule of $\mathcal{D}$ is $\odot R$ and the last rule of $\mathcal{E}$ is $\odot L2$,
- the last rule of $\mathcal{D}$ is $\odot L1$

Whenever you invoke the induction hypothesis, be sure to explain why you are permitted to invoke the induction hypothesis.

[Additional space.]

3 Natural Deduction

The natural deduction rules for the happy-face connective are as follows:

$$\frac{\begin{array}{c}[B\ \mathsf{true}]^x \\ \vdots \\ A\ \mathsf{true} \qquad \bot\ \mathsf{true}\end{array}}{A \mathbin{\☺} B\ \mathsf{true}}\ \☺I^x \qquad \frac{A \mathbin{\☺} B\ \mathsf{true}}{A\ \mathsf{true}}\ \☺E1 \qquad \frac{A \mathbin{\☺} B\ \mathsf{true} \qquad B\ \mathsf{true}}{C\ \mathsf{true}}\ \☺E2$$

Prove local soundness and completeness for ☺. That is, prove the following:

1. If you have a derivation in which the last two rules are $\☺I$ and then $\☺E1$, you can produce a smaller derivation of the same judgement without using the happy-face connective.
2. If you have a derivation in which the last two rules are $\☺I$ and then $\☺E2$, you can produce a smaller derivation of the same judgement without using the happy-face connective.
3. If you have a derivation of $A \mathbin{\☺} B\ \mathsf{true}$, you can produce a larger derivation that eliminates $A \mathbin{\☺} B$ and then reconstructs it.

If necessary, you may use weakening and/or substitution.

[Additional space.]

4 Adequacy

The *polymorphic lambda calculus*, or System F, is an extension of the simply typed lambda calculus. Its grammar is:

$$\begin{array}{rcl} \text{types } \tau & ::= & \alpha \mid o \mid \tau \to \tau \mid \forall \alpha.\tau \\ \text{expressions } e & ::= & x \mid b \mid e\ e \mid \lambda x{:}\tau.e \mid \Lambda\alpha.e \mid e[\tau] \end{array}$$

The form α is a type variable that ranges over types (just as x ranges over terms). $\forall\alpha.\tau$ is the type of terms polymorphic over the type variable α. There are two new expressions which introduce and eliminate $\forall$. Given a term e containing a free type variable α, $\Lambda\alpha.e$ binds α to create a polymorphic term. Conversely, $e[\tau]$ instantiates a polymorphic term e at the type τ. For example, the polymorphic identity function is written $\Lambda\alpha.\lambda x{:}\alpha.x$ and has type $\forall\alpha.\alpha \to \alpha$.

Here is an LF encoding of System F's type syntax:

```
tp : type.
o : tp.
arrow : tp -> tp -> tp.
forall : (tp -> tp) -> tp.
```

and a corresponding definition of the embedding ($\ulcorner - \urcorner$) and de-embedding ($\llcorner - \lrcorner$) functions:

$$\begin{array}{rcl} \ulcorner o \urcorner & = & \texttt{o} \\ \ulcorner \tau_1 \to \tau_2 \urcorner & = & \texttt{arrow}\ \ulcorner \tau_1 \urcorner\ \ulcorner \tau_2 \urcorner \\ \ulcorner \forall\alpha.\tau \urcorner & = & \texttt{forall}\ (\lambda\alpha{:}\texttt{tp}.\ulcorner \tau \urcorner) \\ \llcorner \texttt{o} \lrcorner & = & o \\ \llcorner \texttt{arrow T1 T2} \lrcorner & = & \llcorner \texttt{T1} \lrcorner \to \llcorner \texttt{T2} \lrcorner \\ \llcorner \texttt{forall}\ (\lambda\alpha{:}\texttt{tp.T}) \lrcorner & = & \forall\alpha.\llcorner \texttt{T} \lrcorner \end{array}$$

Problem 1: Encode System F's term syntax in LF.

Problem 2: Define embedding and de-embedding functions for your encoding from Problem 1.

Problem 3: We may define **adequacy for type syntax** (for closed types) as follows:

⌜−⌝ *and* ⌞−⌟ *form a bijection between closed LF terms of type* `tp` *and closed types in System F.*

Unfortunately, this statement is false for the encoding we have provided. Give a counterexample, and justify your answer. (The restriction to closed types is not the error.)

5 Twelf

Recall the following Twelf definitions of `nat` and `list`.

```
nat : type.
z : nat.
s : nat -> nat.

list : nat -> type.
nil : list z.
cons : nat -> list N -> list (s N).
```

In this problem, we define four predicates in Twelf which operate on `nats` and `lists`. All four predicates typecheck, but some or all of them fail `%mode`, `%worlds`, or `%total` checking.

For each code snippet which follows, either:

1. state that it loads without error in Twelf, or
2. describe the error that Twelf will display, and circle the code which causes the error (if applicable). Justify your answer. You do not need to explain how to fix the error.

Problem 1:

```
take : {M:nat} list N -> list M -> type.
%mode take +M +L1 -L2.

take/z : take z nil nil.
take/s : take (s N) (cons X Xs) (cons X L)
         <- take N Xs L.

%worlds () (take _ _ _).
%total M (take M _ _).
```

Problem 2:

```
count-z : list N -> nat -> type.
%mode count-z +L -N.

count-z/nil    : count-z nil z.
count-z/cons/z : count-z (cons z L) (s N)
                 <- count-z L N.
count-z/cons/s : count-z (cons (s _) L) N
                 <- count-z L N.

%worlds () (count-z _ _).
%total L (count-z L _).
```

Problem 3:

```
blend : list M -> list N -> list P -> type.
%mode blend +L1 +L2 -L3.

blend/nil  : blend nil L L.
blend/cons : blend (cons X L1) L2 (cons X L)
             <- blend L2 L1 L.

%worlds () (blend _ _ _).
%total L (blend L _ _).
```

Problem 4:

```
is-prefix : list M -> list N -> type.
%mode is-prefix +L1 -L2.

is-prefix/nil  : is-prefix nil L.
is-prefix/cons : is-prefix (cons X L1) (cons X L2)
                 <- is-prefix L1 L2.

%worlds () (is-prefix _ _).
%total L (is-prefix L _).
```

A Linear Logic

$$\frac{}{A\ \mathsf{true} \Vdash A\ \mathsf{true}} \qquad \frac{\Gamma, A\ \mathsf{true} \Vdash B\ \mathsf{true}}{\Gamma \Vdash A \multimap B\ \mathsf{true}} \qquad \frac{\Gamma \Vdash A \multimap B\ \mathsf{true} \quad \Gamma' \Vdash A\ \mathsf{true}}{\Gamma, \Gamma' \Vdash B\ \mathsf{true}}$$

$$\frac{\Gamma \Vdash A\ \mathsf{true} \quad \Gamma' \Vdash B\ \mathsf{true}}{\Gamma, \Gamma' \Vdash A \otimes B\ \mathsf{true}} \qquad \frac{\Gamma \Vdash A \otimes B\ \mathsf{true} \quad \Gamma', A\ \mathsf{true}, B\ \mathsf{true} \Vdash C\ \mathsf{true}}{\Gamma, \Gamma' \Vdash C\ \mathsf{true}}$$

$$\frac{\Gamma \Vdash A\ \mathsf{true} \quad \Gamma \Vdash B\ \mathsf{true}}{\Gamma \Vdash A \,\&\, B\ \mathsf{true}} \qquad \frac{\Gamma \Vdash A \,\&\, B\ \mathsf{true}}{\Gamma \Vdash A\ \mathsf{true}} \qquad \frac{\Gamma \Vdash A \,\&\, B\ \mathsf{true}}{\Gamma \Vdash B\ \mathsf{true}}$$

$$\frac{\Gamma \Vdash A\ \mathsf{true}}{\Gamma \Vdash A \oplus B\ \mathsf{true}} \qquad \frac{\Gamma \Vdash B\ \mathsf{true}}{\Gamma \Vdash A \oplus B\ \mathsf{true}} \qquad \frac{\Gamma \Vdash A \oplus B\ \mathsf{true} \quad \Gamma', A\ \mathsf{true} \Vdash C\ \mathsf{true} \quad \Gamma', B\ \mathsf{true} \Vdash C\ \mathsf{true}}{\Gamma, \Gamma' \Vdash C\ \mathsf{true}}$$

$$\frac{}{\Vdash 1\ \mathsf{true}} \qquad \frac{\Gamma \Vdash 1\ \mathsf{true} \quad \Gamma' \Vdash C\ \mathsf{true}}{\Gamma, \Gamma' \Vdash C\ \mathsf{true}} \qquad \frac{}{\Gamma \Vdash \top\ \mathsf{true}} \qquad \frac{\Gamma \Vdash 0\ \mathsf{true}}{\Gamma, \Gamma' \Vdash C\ \mathsf{true}}$$