Final Exam

15-816 Linear Logic
Frank Pfenning

May 8, 2012

Name: Andrew ID:

Instructions

- This exam is closed-book, closed-notes.
- You have 3 hours to complete the exam.
- There are 6 problems.

	Ordered Logic	Classical Lin. Logic	Resource Semantics	Forward Chaining	Possibility	Quotations	
	Prob 1	Prob 2	Prob 3	Prob 4	Prob 5	Prob 6	Total
Score							
Max	50	55	40	40	50	15	250

1 Ordered Logic (50 pts)

In this question we explore ordered logic programming. We have the following program load which takes a list of elements and loads them into the ordered context.

$$\mathsf{load}(\mathsf{cons}(x,l),k) \twoheadleftarrow (\mathsf{elem}(x) \twoheadrightarrow \mathsf{load}(l,k)).$$
$$\mathsf{load}(\mathsf{nil},k) \twoheadleftarrow \mathsf{gather}(k).$$

Both load and gather are *negative* atomic predicates. The intent is for gather to return a final value k to be computed from the state created by load. What exactly is to be computed will change from task to task.

Task 1 (5 pts). The query

$$\mathsf{load}(\mathsf{cons}(x_1,\mathsf{cons}(x_2,\ldots,\mathsf{cons}(x_n,\mathsf{nil}))),K)$$

for elements $x_1,\ldots,x_k$ and a free variable K will load the context. Show the contents of the ordered context at the point when $\mathsf{gather}(K)$ is called as a subgoal.

In the programming tasks below we assume that forward chaining takes precedence over backward chaining. In other words, we apply forward-chaining rules to quiescence before considering backward-chaining.

Task 2 (15 pts). Define $\mathsf{gather}(k)$ such that it succeeds with k the *reverse* of the original list. Your program should use the auxiliary *positive* predicate $\mathsf{collect}(l)$. For full credit, you should have just one clause for $\mathsf{gather}(k)$ and one clause for forward chaining.

$\mathsf{gather}(k) \twoheadleftarrow$

forward-chaining collecting clause below:

Task 3 (15 pts). Define alternative gather and collect predicates such that gather(k) succeeds with k being a *subsequence* of the original sequence. By subsequence we mean a list of elements in the same order as in the original, where some elements may have been deleted. For full credit, you should have just one clause for gather(k) and one clause for forward chaining.

gather(k) $\twoheadleftarrow$

forward-chaining collecting clause below:

Task 4 (15 pts). Define alternative gather and collect predicates such that gather(k) succeeds with k begin an arbitrary *permutation* of the original sequence. For full credit, you should have just one clause for gather(k) and one clause for forward chaining.

gather(k) $\twoheadleftarrow$

forward-chaining collecting clause below:

2 Classical Linear Logic (55 pts)

Recall that in classical linear logic, the cut and identity rules are as follows:

$$\frac{\vdash \Sigma, A \quad \vdash \Sigma', A^\perp}{\vdash \Sigma, \Sigma'}\ \mathsf{cut}_A \qquad \frac{}{\vdash A, A^\perp}\ \mathsf{id}_A$$

Task 1 (5 pts). The classical $A \otimes B$ behaves analogously to its intuitionistic version. Show the corresponding classical (right) rule.

Task 2 (5 pts). Rather than a left rule for $A \otimes B$, we define $(A \otimes B)^\perp = A^\perp \parr B^\perp$ and give a (right) rule for $\parr$. Show this rule.

Task 3 (10 pts). Show the identity expansion for $\parr$ in the classical sequent calculus.

Task 4 (15 pts). Show the cut reduction between $\otimes$ and $\invamp$ in the classical sequent calculus.

Task 5 (10 pts). In classical linear logic, the exponential is defined by $(!A)^{\perp} = ?(A^{\perp})$, the following rules

$$\frac{\vdash ?\Sigma, A}{\vdash ?\Sigma, !A}\ ! \qquad \frac{\vdash \Sigma, A}{\vdash \Sigma, ?A}\ ?$$

plus two additional rules. Please name them and show them.

Task 6 (10 pts). While cut elimination holds in classical linear logic, a structural induction proof of its admissibility in the cut-free classical sequent calculus does not go through. Identify the critical case and explain why the induction fails.

3 Resource Semantics (40 pts)

In the resource semantics we track linearity through algebraic reasoning on resource expressions

$$\text{Resource exprs.} \quad p \quad ::= \quad \epsilon \mid p_1 * p_2 \mid \alpha$$

Task 1 (5 pts). Write out the resource equations that characterize linear logic.

Task 2 (5 pts). Recall the $\multimap R$ rule.

Task 3 (5 pts). Recall the $\multimap L$ rule. You may use the tethered or untethered form.

Strict logic has just two forms of resources: *persistent* ones, which can be used arbitrarily often, and *strict* ones, which must be used at least once. We claim that the resource semantics with *exactly the same rules as linear logic* represents strict logic if we add the law of idempotence for resource expressions:

$$p * p = p$$

Task 4 (15 pts). Prove that $\vdash (A \multimap A \multimap B) \multimap (A \multimap B) @ \epsilon$ using your resource rules, where $A \multimap B$ now represents a *strict implication*.

Task 5 (10 pts). Prove that $\vdash B \multimap (A \multimap B)@\epsilon$ does *not* hold in general in strict logic. You may assume cut elimination and that identity can be reduced to atomic propositions.

4 Forward Chaining (40 pts)

Consider a representation of binary numbers in ordered linear logic, where the number $b_{n-1} \cdots b_0$ (with b_0 representing the least significant bit) is represented by the *ordered* context

$$\mathsf{end}, \mathsf{bit}(b_{n-1}), \ldots, \mathsf{bit}(b_0)$$

where each bit b_i is either 0 or 1.

Task 1 (10 pts). The following ordered program *increments* the represented number if started with inc added at the right end of the context. Complete the program, assuming bit, end, and inc are all positive.

$$\mathsf{bit}(0) \bullet \mathsf{inc} \twoheadrightarrow \mathsf{bit}(1)$$

$$\mathsf{bit}(1) \bullet \mathsf{inc} \twoheadrightarrow \underline{\hspace{20em}}$$

$$\mathsf{end} \bullet \mathsf{inc} \twoheadrightarrow \underline{\hspace{20em}}$$

Task 2 (15 pts). Rewrite the above program in *linear* logic. We represent the number now as

$$\mathsf{end}(d_n), \mathsf{bit}(d_n, b_{n-1}, d_{n-1}), \ldots, \mathsf{bit}(d_1, b_0, d_0)$$

where each b_i is either 0 or 1, and the d_i are mutually distinct destinations. The command to increment starting at bit i is represented as the proposition $\mathsf{inc}(d_i)$.

Write a forward chaining program to increment a number in this representation. When $\mathsf{inc}(d_0)$ is added to the context representing the number n, it should reach quiescence with the context containing the represention of the number $n+1$. You may assume bit, end, and inc are positive, or you may use a monad.

Task 3 (15 pts). Your program for incrementing a number is likely sequential. Write a forward-chaining program that computes the *parity* of the binary number. When given a number in the representation above, it should reach quiescence in a state with only $\mathsf{bit}(d', 1, d)$ if there are an odd number of bits 1 and $\mathsf{bit}(d', 0, d)$ if there are an even number of bits 1. The destinations d and d' are irrelevant and may be arbitrary. Your program should admit some parallelism.

5 Possibility (50 pts)

We can introduce $?A$ into intuitionistic linear logic with the new judgment $A\ poss$ and the following rules:

$$\frac{\Gamma\ ;\ \Delta \vdash A}{\Gamma\ ;\ \Delta \vdash A\ poss}\ \mathsf{poss}$$

$$\frac{\Gamma\ ;\ \Delta \vdash A\ poss}{\Gamma\ ;\ \Delta \vdash ?A}\ ?R \qquad \frac{\Gamma\ ;\ A \vdash C\ poss}{\Gamma\ ;\ ?A \vdash C\ poss}\ ?L$$

All the left rules as well as copy, cut and cutbang are generalized to allow a succedent of the form $C\ poss$

Task 1 (10 pts). State the new cut rule needed, cut?.

Task 2 (10 pts). Prove $\vdash\ !(A \multimap B) \multimap ?A \multimap ?B$

Task 3 (10 pts). Show the identity expansion for $?A$.

Task 4 (5 pts). What is the polarity of $?A$?

Task 5 (15 pts). Give focusing versions of the new rules poss, $?R$ and $?L$.

6 Quotations (15 pts)

Task 1 (15 pts).

> *Give a man a fish and you feed him for a day. Teach a man how to fish and you feed him for a lifetime.* – Chinese proverb

Express this quotation in linear logic, using the following vocabulary:

Types	$\mathsf{person}, \mathsf{food}$	
Predicates	$\mathsf{own}(x, y)$	person x owns food y
	$\mathsf{eat}(x, y)$	person x can eat food y
	$\mathsf{fish}(x)$	food x is fish

Since we do not model time, think of "*for a day*" as "*once*", and "*for a lifetime*" as "*arbitrarily often*".