# 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.

$$\begin{aligned} \mathsf{load}(\mathsf{cons}(x,l),k) &\leftarrow (\mathsf{elem}(x) \twoheadrightarrow \mathsf{load}(l,k)). \\ \mathsf{load}(\mathsf{nil},k) &\leftarrow \mathsf{gather}(k). \end{aligned}$$

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

$$load(cons(x_1, cons(x_2, ..., cons(x_n, nil))), K)$$

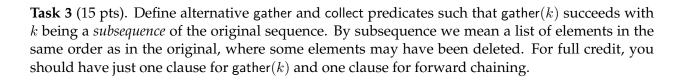
for elements  $x_1, \ldots, x_k$  and a free variable K will load the context. Show the contents of the ordered context at the point when gather(K) is called as a subgoal.

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 gather(k) such that it succeeds with k the *reverse* of the original list. Your program should use the auxiliary *positive* predicate collect(l). For full credit, you should have just one clause for gather(k) and one clause for forward chaining.

$$gather(k) \leftarrow$$

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.

$$gather(k) \leftarrow$$

forward-chaining collecting clause below:

### 2 Classical Linear Logic (55 pts)

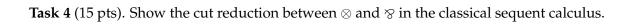
Recall that in classical linear logic, the cut and identity rules are as follows:

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

**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  $(A \otimes B)^{\perp} = A^{\perp} \otimes B^{\perp}$  and give a (right) rule for  $\otimes$ . Show this rule.

Task 3 (10 pts). Show the identity expansion for  $\mathop{\otimes}$  in the classical sequent calculus.



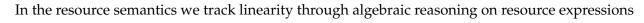
**Task 5** (10 pts). In classical linear logic, the exponential is defined by  $(!A)^{\perp} = ?(A^{\perp})$ , the following rules

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

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)



Resource exprs. 
$$p ::= \epsilon \mid p_1 * p_2 \mid \alpha$$

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

**Task 2** (5 pts). Recall the -∞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

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.

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

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

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

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  $inc(d_i)$ .

Write a forward chaining program to increment a number in this representation. When  $inc(d_0)$  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  $\operatorname{bit}(d',1,d)$  if there are an odd number of bits 1 and  $\operatorname{bit}(d',0,d)$  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:

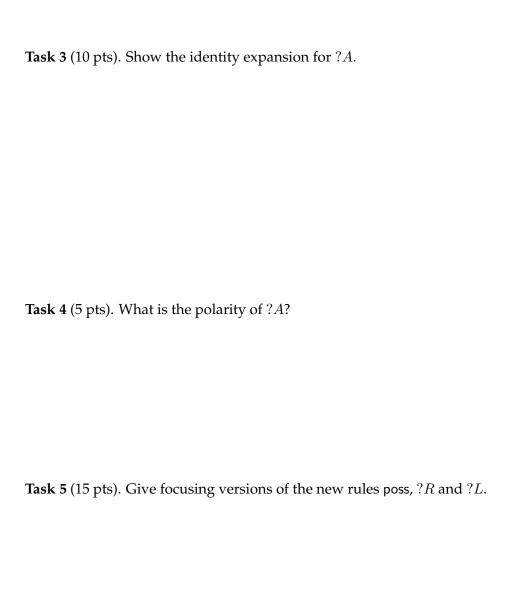
$$\frac{\Gamma \; ; \; \Delta \vdash A}{\Gamma \; ; \; \Delta \vdash A \; poss} \; \mathsf{poss}$$

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

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  $\vdash !(A \multimap B) \multimap ?A \multimap ?B$ 



## 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:

 $\begin{array}{lll} \text{Types} & \text{person, food} \\ \text{Predicates} & \text{own}(x,y) & \text{person } x \text{ owns food } y \\ & \text{eat}(x,y) & \text{person } x \text{ can eat food } y \\ & \text{fish}(x) & \text{food } x \text{ is fish} \end{array}$ 

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