Lecture Notes on Linear Logic

15-816: Substructural Logics
Frank Pfenning

Lecture 11
October 4, 2016

In this lecture we introduce the sequent calculus for the multiplicative-additive fragment of linear logic [**?**], often abbreviated as MALL. From intuitionistic linear logic it is only missing the exponential, $!A$, which, as an antecedent, allows A to be used arbitrarily many times. The exponential will be introduced next week, in Lecture 13.

I have to admit this is not actually what happened in lecture when in addition to introducing linear logic I also tried to talk about combining logics. I slightly revised my—ahem—somewhat cryptic approach and tried again in Lecture 12.

1 Exchange

The way we obtain linear logic from ordered logic is just to allow exchange among the antecedents of a sequent. This is importing into the sequent calculus the difference between ordered and linear inference from the beginning of the course. Allowing exchange can be formalized in two ways: with an explicit inference rule

$$\frac{\Omega_L\ B\ A\ \Omega_R \vdash C}{\Omega_L\ A\ B\ \Omega_R \vdash C}\ \mathsf{exchange}$$

or we can treat the context as a multiset rather than a sequence of propositions. We write

$$\Delta = (A_1, \ldots, A_n)$$

for antecedents that form multisets and for now reserve Ω for ordered antecedents. We will usually use the latter approach to reduce the bureaucracy of explicit inference rules. The comma separating antecedents and the use of the letter Δ will remind us that "*order doesn't matter*".

2 Collapsing Connectives

When we identify antecedents up to exchange, several previously distinct connectives become indistinguishable. For example, $A \setminus B = B \,/\, A$ become the single connective of *linear implication*, written $A \multimap B$. Similarly, $A \bullet B = A \circ B$ becomes *multiplicative conjunction* (or *simultaneous conjunction*), written $A \otimes B$. All the other connectives remain the same. This information can be gleaned from the inference rules. We only show one example, the emergent rules for linear implication.

$$\frac{\Delta, A \vdash B}{\Delta \vdash A \multimap B} \multimap R \qquad \frac{\Delta' \vdash A \quad \Delta, B \vdash C}{\Delta, \Delta', A \multimap B \vdash C} \multimap L$$

In the latter rule we write $A \multimap B$ on the right end of the antecedents, but due to exchange we could have written it anywhere. Similarly, writing Δ, Δ' means (reading the rule bottom-up as we are used to) that we take a multiset of antecedents and split it into two. Thus there are 2^n ways to split a context of n antecedents, while in the ordered case there are only $n + 1$ ways.

3 Cut Reduction and Identity Expansion

The properties of cut reduction and identity expansion carry over. Identity expansion is automatic, since we only relax the applicability of rules. Cut reduction also works exactly as before, because the only difference is a relaxed order among antecedents. Similarly, the admissibility of cut in the cut-free sequent calculus works just as before, with the exact same induction measure and argument. In fact, we will unify the different proofs into a single one in a future lecture.

4 Operational Interpretation

The assignment of proof terms and processes remains *exactly* the same as for the ordered case. In other words, linear processes (as compared to or-

dered ones) only relax the requirements on typing of processes proofs while retaining their operational meaning from ordered logic. Therefore we do not need to introduce any new rules.

In principle, we would have to reprove progress and preservation, since the more relaxed form of typing a priori might be too weak to guarantee them. However, the proof follows the exact same patterns as before and contains no new insights.

As an example, we reconsider lists in their linear rather than ordered form and show that they can be reversed.

$\mathsf{list}_A = \oplus\{\mathsf{cons} : A \otimes \mathsf{list}_A, \mathsf{nil} : \mathbf{1}\}$

Linearity guarantees that elements of a list are preserved, although their order may not. To define *reverse* we use an auxiliary process *rev* which has an accumulator argument in which we construct the reversed list.

$\mathsf{list}_A = \oplus\{\mathsf{cons} : A \otimes \mathsf{list}_A, \mathsf{nil} : \mathbf{1}\}$

$(k{:}\mathsf{list}_A)\ (a{:}\mathsf{list}_A) \vdash \mathit{rev} :: (l{:}\mathsf{list}_A)$

$$
\begin{array}{lll}
l \leftarrow \mathit{rev} \leftarrow k\ a = & & \\
\quad \mathsf{case}\ k\ \ (\mathsf{cons} \Rightarrow x \leftarrow \mathsf{recv}\ k\ ; & \% & (x{:}A)\ (k{:}\mathsf{list}_A)\ (a{:}\mathsf{list}_A) \vdash (l{:}\mathsf{list}_A) \\
\qquad\qquad\qquad\quad a' \leftarrow \mathit{cons} \leftarrow x\ a\ ; & \% & (k{:}\mathsf{list}_A)\ (a'{:}\mathsf{list}_A) \vdash (l{:}\mathsf{list}_A) \\
\qquad\qquad\qquad\quad l \leftarrow \mathit{rev} \leftarrow k\ a' & & \\
\qquad\quad \mid \mathsf{nil} \Rightarrow \mathsf{wait}\ k\ ;\ l \leftarrow a' & \% & \text{accumulator becomes result}
\end{array}
$$

$k{:}\mathsf{list}_A \vdash \mathit{reverse} :: (l{:}\mathsf{list}_A)$

$$
\begin{array}{lll}
l \leftarrow \mathit{reverse} \leftarrow k & & \\
\quad n \leftarrow \mathit{nil}\ ; & & \\
\quad l \leftarrow \mathit{rev} \leftarrow k\ n & \% & \text{initialize accumulator with } \mathit{nil}
\end{array}
$$

Observe that the call to *cons* could not be typed in the case of ordered lists.

Exercises

Exercise 1 Show one principal and one commutative case among the $\multimap$ and $\otimes$ connectives in the proof of the admissibility of cut for linear logic.

Exercise 2 A linear proposition can be turned into an ordered proposition by deciding for each muliplicative conjunction ($\otimes$) if it should become fuse ($\bullet$) or twist ($\circ$) and for each linear implication ($\multimap$) if it should become under ($\backslash$) or over ($/$). For example, we can translate the provable $(A \otimes (A \multimap B)) \multimap B$ into the provable $(A \bullet (A \backslash B)) \,/\, B$.

1. If possible, construct a provable linear formula containing only $\otimes$, $\multimap$ and propositional variables such that all of its ordered translations are provable in ordered logic. If one exists, try to minimize the number of its connectives.

2. If possible, construct a provable linear formula such that none of its translations to ordered logic are provable. If one exists, try to minimize the number of its connectives.

Exercise 3 Analyze the parallel complexity of the *rev* and *reverse* processes in **??** in terms of latency and throughput. Given a list k of length n that produces its elements with a constant delay c between consecutive elements:

1. (*Latency*) How many steps (counting only communications, but not spawns or forwards) until the first element can be retrieved from l?

2. (*Throughput*) How many steps until the whole reversed lists can be retrieved from l, assuming a client that has no delay between successive interactions along l.

3. If we reverse twice

$$l \leftarrow \mathsf{reverse} \leftarrow k \; ; \; k' \leftarrow \mathsf{reverse} \leftarrow l$$

   the result k' should be observationally equivalent to the given k. Analyze latency and throughput for the pipeline from k to k'.

References

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.