Lecture Notes on
From Rules to Propositions

15-816: Substructural Logics
Frank Pfenning

Lecture 2
September 1, 2016

We review the ideas of ephemeral truth and linear inference with another example from graph theory: constructing spanning trees for graphs. Next we consider how the process of grammatical inference in the Lambek calculus [Lam58], where an order of facts is prescribed, can be used to define other forms of computation. Finally, we generalize our formalization of logical inference to encompass *hypothetical reasoning* which will give rise to Gentzen's sequent calculus [Gen35]. The sequent calculus will be the gateway that allows us to move from pure logical inference to a definition of logical connectives.

1 Example: Spanning Trees

A *spanning tree* for a connected graph is a graph that has the same nodes but only a subset of the edges such that there is no cycle. In order to define rules for constructing a spanning tree for a graph we will simultaneously manipulate two graphs: the original graph and its spanning tree. We therefore add a third argument to our representation of graphs (from Lecture 1) which identifies *which* graph a node or edge belongs to.

$\mathsf{node}(x, g)$	x is a node in graph g
$\mathsf{edge}(x, y, g)$	there is an edge from x to y in graph g

The rule of symmetry stays within one graph g:

$$\frac{\mathsf{edge}(x, y, g)}{\mathsf{edge}(y, x, g)}\ \mathsf{sym}$$

Now assume we have a graph g and want to build a spanning tree t. Here is a simple algorithm for building t. We begin by picking an arbitrary node x from g and create t with x as its only node. Now we repeatedly pick an edge that connects a node x already in the tree with a node y not yet in the tree and add that edge and the node y into the tree. When no such edges exist any more, we either must have a spanning tree already or the original graph was not connected. We can determine this, for example, by checking if there are any nodes left in the graph that haven't been added to the tree.

This algorithm has two kinds of steps, so its representation in linear logic has two rules. The first step moves an arbitrary node from the graph to the tree.

$$\frac{\mathsf{node}(x, \mathsf{g})}{\mathsf{node}(x, \mathsf{t})}\ \mathsf{start}?$$

This rule can be used only once, at the very beginning of the algorithm and must be prohibited afterwards, or we could just use it to move all nodes from the graph to the tree without moving any edges. So we can either say the rule must be ephemeral itself, or we create a new ephemeral proposition init which only exists in the initial state and is consumed by the first step.

$$\frac{\mathsf{init} \quad \mathsf{node}(x, \mathsf{g})}{\mathsf{node}(x, \mathsf{t})}\ \mathsf{start}$$

The next rule implements the idea we described in the text above. All propositions are ephemeral, so we can implement "*a node y not yet in the tree*" by checking whether it is still in the graph, thereby consuming it.

$$\frac{\mathsf{node}(x, \mathsf{t}) \quad \mathsf{edge}(x, y, \mathsf{g}) \quad \mathsf{node}(y, \mathsf{g})}{\mathsf{node}(x, \mathsf{t}) \quad \mathsf{edge}(x, y, \mathsf{t}) \quad \mathsf{node}(y, \mathsf{t})}\ \mathsf{move}$$

A proof using these two rules describes a particular sequence of moves, taking edges from the graph and adding them to the spanning tree.

In order to convince ourselves that this is correct, it is important to understand the state invariants. Initially, we have

init	
$\mathsf{node}(x, \mathsf{g})$	for every node x in g
$\mathsf{edge}(x, y, \mathsf{g})$	for every edge from x to y in g

Rule move does not apply, because we do not yet have a node in t, so any inference must begin with rule start, consuming init and producing one

node x_0 in t.

$\mathsf{node}(x_0, \mathsf{t})$	for some node x_0
$\mathsf{node}(x, \mathsf{g})$	for every node $x \neq x_0$ in g
$\mathsf{edge}(x, y, \mathsf{g})$	for every edge from x to y in g

Now rule start can no longer be applied, and we apply move as long as we can. The rule preserves the invariant that each node x from the initial graph is either in t ($\mathsf{node}(x, \mathsf{t})$) or in g ($\mathsf{node}(x, \mathsf{g})$). It further preserves the invariant that each edge in the original graph is either in t ($\mathsf{edge}(x, y, \mathsf{t})$) or still in g ($\mathsf{edge}(x, y, \mathsf{g})$).

If the algorithm stops and no nodes are left in g, we must have moved all n nodes originally in g. One is moved in the start rule, and $n - 1$ are moved in applications of the move rule. In every application of the move rule we also move exactly one edge from g to t, so t now has n nodes and $n - 1$ edges. Further, it is connected since anytime we move an edge it connects to something already in the partial spanning tree. A connected graph with n nodes and $n - 1$ edges must be a tree, and it spans g because it has all the nodes of g.

If the algorithms stops and there are some nodes left in g, then the original graph must have been disconnected. Assume that g is connected, y is left in g, and we started with x_0 in the first step. Because g is connected, there must be a path from x_0 to y. We prove that this is impossible by induction on the structure of this path. The last edge connects some node y' to y. If y' is in the tree, then the rule move would apply, but we stipulated that the algorithm only stops if move does not apply. If y' is in the graph but not in the tree, then we apply the induction hypothesis to the subpath from x_0 to y'.

2 Example: Counting in Binary

In this section we see how to encode binary counting via ordered inference as in the Lambek calculus. We represent a binary number 1011_2 (which is eleven) by the following ordered propositions:

eps b1 b0 b1 b1

where b1 represents bit 1, b0 represents the bit 0, and eps represents the empty string, thereby marking the end of the binary string. We think of increment as another proposition we add at the *right end* of the string. For

example, if we want to increment the number above twice, we would write

```
eps b1 b0 b1 b1 inc inc
```

If we define the correct rules we would like to infer

```
eps b1 b0 b1 b1 inc inc
⋮
eps b1 b1 b0 b1
```

Before you turn the page you might consider if you can define ordered inference rules to define the increment operation.

We need the following three rules:

$$\frac{\mathsf{b0\ inc}}{\mathsf{b1}}\ \mathsf{inc0} \qquad \frac{\mathsf{b1\ inc}}{\mathsf{inc\ b0}}\ \mathsf{inc1} \qquad \frac{\mathsf{eps\ inc}}{\mathsf{eps\ b1}}\ \mathsf{inceps}$$

The inc1 rule implements the carry bit by incrementing the remainder of the bit string, while inceps deposits the carry as the highest bit in case we have reached the end of the bit string.

These rules encode some parallelism. For example, after a single step of inference we have

eps b1 b0 b1 b1 inc inc
eps b1 b0 b1 inc b0 inc
⋮
eps b1 b1 b0 b1

Here we only show the state after each inference and not the rule used (which is inc1) for the sake of conciseness. In the second line, we can apply inc1 or inc0 or (because they are independent) both of them simultaneously, which gives us

eps b1 b0 b1 b1 inc inc
eps b1 b0 b1 inc b0 inc
eps b1 b0 inc b0 b1
⋮
eps b1 b1 b0 b1

Now we can obtain the desired conclusion with one more step of inference.

eps b1 b0 b1 b1 inc inc
eps b1 b0 b1 inc b0 inc
eps b1 b0 inc b0 b1
eps b1 b1 b0 b1

3 Ordered Hypothetical Judgments

The notion of grammatical inference represents parsing as the process of constructing a proof. For example, if we have a phrase (= sequence of words) $w_1 \dots w_n$ we find their syntactical types $x_1 \dots x_n$ (guessing if necessary if they are ambiguous) and then set up the problem

$$(w_1 : x_1) \cdots (w_n : x_n)$$
$$\vdots$$
$$? : s$$

where "?" will represent the parse tree as a properly parenthesized expression (assuming, of course, we can find a proof).

So far, we can only represent the inference itself, but not the *goal* of parsing a whole sentence. In order to express that we introduce *hypothetical judgments* as a new primitive concept. The situation above is represented as

$$(w_1 : x_1) \cdots (w_n : x_n) \vdash ? : s$$

or, more generally, as

$$(p_1 : x_1) \cdots (p_n : x_n) \vdash r : z$$

The turnstile symboxl "$\vdash$" here separates the *succedent* $r : z$ from the *antecedents* $p_i : x_i$. We sometimes call the left-hand side the *context* or the *hypotheses* and the right-hand side the *conclusion*. Calling the succedent a conclusion is accurate in the sense that it is the conclusion of a hypothetical deduction, but it can also be confusing since we also used "conclusions" to describe what is below the line in a rule of inference. We hope it will always be clear from the situation which of these we mean.

Since we are studying ordered inference right now, the antecedents that form the context are intrinsically ordered. When we want to refer to a sequence of such antecedents we write Ω where "Omega" is intended to suggest "Order". When we capture other forms of inference like linear inference we will revisit this assumption.

4 Inference with Sequents: Looking Left

Now that we have identified hypothetical judgments, written as sequents $\Omega \vdash r : z$, we should examine what this means for our logical rules of inference. Fortunately, we have had only two connectives, *over* and *under*, first shown here without the proof terms (that is, without the parse trees):

$$\frac{x \,/\, y \quad y}{x}\ \text{over} \qquad \frac{y \quad y \setminus x}{x}\ \text{under}$$

Now that the propositions we know appear as antecedents, the direction of the rules appears to be reversed when considered on sequents.

$$\frac{\Omega_L\ x\ \Omega_R \vdash z}{\Omega_L\ (x \,/\, y)\ y\ \Omega_R \vdash z}\ /L^* \qquad \frac{\Omega_L\ x\ \Omega_R \vdash z}{\Omega_L\ y\ (y \setminus x)\ \Omega_R \vdash z}\ \setminus L^*$$

We have written Ω_L and Ω_R to indicate the rest of the context, which remains unaffected by the inference. These rules operate on the left of the turnstile, that is, on antecedents, and we have therefore labeled them $/L^*$ and $\backslash L^*$, pronounced *over left* and *under left*. While helpful for today's lecture, we will have to revise these rules at the beginning of the next lecture, so we have marked them with an asterisk to remind us that they are only preliminary.

Redecorating the rules with proof terms (that is, parse trees in grammatical inference):

$$\frac{p : x \,/\, y \quad q : y}{(p\,q) : x}\ \mathsf{over} \qquad \frac{q : y \quad p : y \setminus x}{(q\,p) : x}\ \mathsf{under}$$

Now that the propositions we know appear as antecedents, the direction of the rules appears to be reversed when considered on sequents.

$$\frac{\Omega_L\ ((p\,q) : x)\ \Omega_R \vdash r : z}{\Omega_L\ (p : x \,/\, y)\ (q : y)\ \Omega_R \vdash r : z}\ /L^* \qquad \frac{\Omega_L\ ((q\,p) : x)\ \Omega_R \vdash r : z}{\Omega_L\ (q : y)\ (p : y \setminus x)\ \Omega_R \vdash r : z}\ \backslash L^*$$

Our inferences, now taking place on the antecedent, take us upward in the tree, so when we have a situation such as

$$\begin{array}{c}(w_1 : x_1) \cdots (w_n : x_n) \\ \vdots \\ p : s\end{array}$$

where we *have* deduced $p : s$, we are now in the situation

$$\begin{array}{c}p : s \vdash \mathsf{?} : s \\ \vdots \\ (w_1 : x_1) \cdots (w_n : x_n) \vdash \mathsf{?} : s\end{array}$$

This means we need one more rule to complete the proof and signal the success of a hypothetical proof. Both forms with and without the proof terms should be self-explanatory. We use id (for *identity*) to label this inference.

$$\frac{}{x \vdash x}\ \mathsf{id}_x \qquad \frac{}{p : x \vdash p : x}\ \mathsf{id}_x$$

Because we wanted to represent the goal of parsing a sequence of words as complete sentence, no additional antecedents besides x are permitted in this rule. Otherwise, a phrase such as *Bob likes Alice likes* could be incorrectly seen to parse as the sentence *((Bob likes) Alice)* ignoring the second *likes*.

5 Inference with Sequents: Looking Right

We already noted in Lecture 1 that $x \setminus (y \,/\, z)$ should be somehow equivalent to $(x \setminus y) \,/\, z$ since both yield a y when given and x to the left and z to the right. Setting this equivalence up as two hypothetical judgments

$$x \setminus (y \,/\, z) \vdash (x \setminus y) \,/\, z$$

and

$$(x \setminus y) \,/\, z \vdash x \setminus (y \,/\, z)$$

that we are trying to prove however fails. No inference is possible. We are lacking the ability to express when we can deduce a *succedent* with a logical connective. Lambek states that we should be able to deduce

$$\frac{z}{x \,/\, y} \qquad \text{if} \qquad \frac{z \quad y}{x}$$

So $x \,/\, y$ should follow from z if we get x if we put y to the right of z. With pure inference, as practiced in the last lecture, we had no way to turn this "*if*" into form of inference rule. However, armed with hypothetical judgments it is not difficult to express precisely this:

$$\frac{z \; y \vdash x}{z \vdash x \,/\, y}$$

Instead of a single proposition z we allow a context, so we write this

$$\frac{\Omega \; y \vdash x}{\Omega \vdash x \,/\, y} \; /R$$

This is an example of a *right rule*, because it analyzes the structure of a proposition in the succedent and we pronounce it as *over right*. The $\setminus R$ (*under right*) rule can be derived analogously.

$$\frac{y \; \Omega \vdash x}{\Omega \vdash y \setminus x} \; \setminus R$$

In the next lecture we will look at the question how we know that these rules are correct. For example, we might have accidentally swapped these two rules, in which case our logic would somehow be flawed. And, in fact, our rules are already flawed but we do not have the tools yet to see this.

Let's come back to the motivating example and try to construct a proof of

$$x \setminus (y \,/\, z) \vdash (x \setminus y) \,/\, z$$

Remember, all the rules work bottom-up, either on some antecedent (a left rule) or on the succedent (a right rule). No left rule applies here (there is no x to the left of $x \setminus (\ldots)$) but fortunately the $/R$ rule does.

$$\frac{x \setminus (y \,/\, z) \quad z \vdash x \setminus y}{x \setminus (y \,/\, z) \vdash (x \setminus y) \,/\, z}\, /R$$

Again, no left rule applies (the parentheses are in the wrong place) but a right rule does.

$$\dfrac{\dfrac{x \quad x \setminus (y \,/\, z) \quad z \vdash x}{x \setminus (y \,/\, z) \quad z \vdash x \setminus y}\, \setminus R}{x \setminus (y \,/\, z) \vdash (x \setminus y) \,/\, z}\, /R$$

Finally, now a left rule applies.

$$\dfrac{\dfrac{\dfrac{y \,/\, z \quad z \vdash y}{x \quad x \setminus (y \,/\, z) \quad z \vdash y}\, \setminus L^*}{x \setminus (y \,/\, z) \quad z \vdash x \setminus y}\, \setminus R}{x \setminus (y \,/\, z) \vdash (x \setminus y) \,/\, z}\, /R$$

One more left rule, and then we can apply identity.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{y \vdash y}\, \mathsf{id}_y}{y \,/\, z \quad z \vdash y}\, /L^*}{x \quad x \setminus (y \,/\, z) \quad z \vdash y}\, \setminus L^*}{x \setminus (y \,/\, z) \quad z \vdash x \setminus y}\, \setminus R}{x \setminus (y \,/\, z) \vdash (x \setminus y) \,/\, z}\, /R$$

The proof in the other direction is similar and left as Exercise 5.

We have left out the proof terms here, concentrated entirely on the logical connectives. We will return to proof terms for ordered hypothetical judgment in a future lecture and proceed to conjecture some logical connectives and how to define them via their left and right rules.

6 Alternative Conjunction

As already mentioned in the last lecture, some words have more than one syntactic type. For example, *and* has type $s \setminus s / s$ (omitting parentheses now since the two forms are equivalent by the reasoning the previous section) and also type $n \setminus n^* / n$, constructing a plural noun from two singular ones. We can combine this into a single type $x \,\&\, y$, pronounced x *with* y:

$$\textit{and} : (s \setminus s / s) \,\&\, (n \setminus n^* / n)$$

Then, in a deduction, we are confronted with a choice between the two for every occurrence of *and*. For example, in typing *Alice and Bob work and Eve likes Alice*, we choose $n \setminus n^* / n$ for the first occurence of *and*, and $s \setminus s / s$ for the second.

Lambek did not explicitly define this connective, but it would be defined by the rules

$$\frac{x \,\&\, y}{x}\ \mathsf{with}_1 \qquad \frac{x \,\&\, y}{y}\ \mathsf{with}_2$$

In the proof term we might write .1 for the first meaning and .2 for the second meaning of the word.

$$\frac{p : x \,\&\, y}{p.1 : x}\ \mathsf{with}_1 \qquad \frac{p : x \,\&\, y}{p.2 : y}\ \mathsf{with}_2$$

so that the parse tree for the sentence above might become

((Alice and.1 Bob) work) and.2 (Eve likes Bob)

where we have omitted parentheses that are redundant due to the associativity of $\setminus - /$.

As before, these rules turn into left rules in the sequent calculus, shown here only without the proof terms.

$$\frac{\Omega_L\ x\ \Omega_R \vdash z}{\Omega_L\ x \,\&\, y\ \Omega_R \vdash z}\ \&L_1 \qquad \frac{\Omega_L\ y\ \Omega_R \vdash z}{\Omega_L\ x \,\&\, y\ \Omega_R \vdash z}\ \&L_2$$

To derive the right rule we must ask ourselves under which circumstances we could use a proposition both as an x and as a y. That's true, if we can show both.

$$\frac{\Omega \vdash x \quad \Omega \vdash y}{\Omega \vdash x \,\&\, y}\ \&R$$

7 Concatenation

In a sequent, there are multiple antecedents (in order!) but only one succedent. So how could we encode the goal we had in the binary counting example:

$$\begin{array}{l}\mathsf{eps\ b1\ b0\ b1\ b1\ inc\ inc}\\ \vdots\\ \mathsf{eps\ b1\ b1\ b0\ b1}\end{array}$$

Clearly, this is a hypothetical judgment but the succedent is not a single proposition. In order to define over and under, it is important to maintain a single succedent, so we need to define a new connective that expresses adjacency as a new proposition. We write $x \bullet y$ (read: x *fuse* y). In the Lambek calculus, we would simply write

$$\frac{x \bullet y}{x\ y}\ \mathsf{fuse}$$

As a left rule, this is simple turned upside down and becomes

$$\frac{\Omega_L\ x\ y\ \Omega_R \vdash z}{\Omega_L\ x \bullet y\ \Omega_R \vdash z}\ \bullet L$$

As a right rule for $x \bullet y$, we have to divide the context into to segments, one proving x and the other proving y.

$$\frac{\Omega_1 \vdash x \quad \Omega_2 \vdash y}{\Omega_1\ \Omega_2 \vdash x \bullet y}\ \bullet R$$

Note that there is some nondeterminism in this rule if we decide to use it to prove a sequent, because we have to decide *where* to split the context $\Omega = (\Omega_1\ \Omega_2)$. For a context with n propositions there are $n + 1$ possibilities. For example, if we want to express that a phrase represented by Ω is parsed into *two sentences* we can prove the hypothetical judgment

$$\Omega \vdash s \bullet s$$

We can then prove

$$\begin{array}{cccccccc} \textit{Alice} & \textit{works} & \textit{Bob} & \textit{sleeps} & & & ? \\ : & : & : & : & & & : \\ n & n \setminus s & n & n \setminus s & \vdash & & s \bullet s \end{array}$$

but we have to split the phrase exactly between *works* and *Bob* so that both premises can be proved. Assuming a notation of $p \cdot q : x \bullet y$ if $p : x$ and $q : y$, the proof term for $s \bullet s$ in this example would be $(\textit{Alice works}) \cdot (\textit{Bob sleeps})$.

8 Emptiness

In this section we consider $\mathbf{1}$, the unit of concatenation, which corresponds to the empty context. The left and right rules are nullary versions of the binary concatenation. In particular, there must be no antecedents in the right rule for $\mathbf{1}$.

$$\frac{}{\vdash \mathbf{1}}\ \mathbf{1}R \qquad \frac{\Omega_L\ \Omega_R \vdash z}{\Omega_L\ \mathbf{1}\ \Omega_R \vdash z}\ \mathbf{1}L$$

9 An Unexpected Incompleteness

In functional programming there is a pattern called *currying* which says that instead of a functions of type $(\tau \times \sigma) \to \rho$, passing a pair with values of type τ and σ, we can pass the arguments sequentially as indicated by the type $\tau \to (\sigma \to \rho)$. Logically, this is manifested by the isomorphism between these two types when considered as propositions, where $\times$ is conjunction and $\to$ is implication.

Is there something similar in ordered logic? First we note that *over* and *under* are a form of implication, distinguished merely by whether they expect their argument on the left or on the right. Concatenation is a form of conjunction since it puts together two proofs. Let's consider $(x \bullet y) \setminus z$. This expect x followed by y to its left and concludes z. Similarly, $y \setminus (x \setminus z)$ expects a y to its left and then an x next to that which, if you had concatenated them together, would be $x \bullet y$. So these two seem like they should be intuitively equivalent. Let's try to use the tools of logic to prove that.

First, $y \setminus (x \setminus z) \vdash (x \bullet y) \setminus z$. We show here the completed proof, but you should view it step by step *going upward* from the conclusion.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{z \vdash z}\ \mathsf{id}_z}{x \quad x \setminus z \vdash z}\ \setminus L^*}{x \quad y \quad y \setminus (x \setminus z) \vdash z}\ \setminus L^*}{x \bullet y \quad y \setminus (x \setminus z) \vdash z}\ \bullet L}{y \setminus (x \setminus z) \vdash (x \bullet y) \setminus z}\ \setminus R$$

Now for the other direction. Unfortunately, this does not go as well.

$$\dfrac{\dfrac{\overset{?}{x \quad y \quad (x \bullet y) \setminus z \vdash z}}{y \quad (x \bullet y) \setminus z \vdash x \setminus z}\setminus R}{(x \bullet y) \setminus z \vdash y \setminus (x \setminus z)}\setminus R$$

The sequent at the top should be intuitively provable, since we should be able to combine x and y to $x \bullet y$ and then use the $\setminus L^*$ rule, but there is no such rule. All rules in the sequent calculus so far *decompose* connectives in the antecedent (left rules) or succedent (right rules), but here we would like to *construct* a proof of a compound proposition. We could add an ad hoc rule to handle this situation, but how do we know that the resulting system does not have other unexpected sources of incompleteness?

In the next lecture we will first fix this problem and then systematically study how to ensure that our inference rules do not exhibit similar problems.

Exercises

Exercise 1 Consider variations of the representation and rules in the spanning tree example from Section 1. Consider all four possibilities of nodes and edges in g and t being ephemeral or persistent. In each case show the form of the three rules in question: sym (possibly with two variants), start, and move, indicate if the modification would be correct, and spell out how to check if a proper spanning tree has been built in the final state.

Exercise 2 Consider the encoding of binary numbers in ordered logic as in Section 2. Assume a new proposition par for *parity* and write rules so that the binary representation of a number followed by par computes eps b0 or eps b1 if we have an even or odd number of ones, respectively.

Exercise 3 Represent the computation of a Turing machine using ordered inference as in Section 2. You will need to decide on a finite, but potentially unbounded representation of the tape, the finite number of states, and the state transitions, such that each step of the Turing machine corresponds to one or more steps of ordered inference. Make sure to describe all parts of the encoding carefully.

Exercise 4 Represent instances of Post correspondence problem in ordered logic so that ordered inference as in Section 2 from an initial state proves a distinguished proposition s (for success) if and only if the problem has a solution. One should be able to extract the actual solution from the proof. Make sure to describe all parts of the encoding carefully.

Exercise 5 Prove $(x \setminus y) \,/\, z \vdash x \setminus (y \,/\, z)$

Exercise 6 Find equivalences along the lines of associativity, currying, or distributivity laws as in the first two examples and prove both directions. Note (but do not prove) where they don't seem to exist if we restrict ourselves to the over, under, fuse, and with connectives. You may need to refer to Lecture 3 to use the stronger versions of $\setminus L$ and $/L$ that resolve the incompleteness in Section 9.

$(x \setminus y) \,/\, z \dashv\vdash x \setminus (y \,/\, z)$ (see Section 5 and Exercise 5)

$(x \bullet y) \setminus z \dashv\vdash y \setminus (x \setminus z)$ (see Section 9 and Lecture 3)

1. $x \,/\, (y \bullet z) \dashv\vdash A(x, y, z)$

2. $(x \bullet y) \,/\, z \dashv\vdash B(x, y, z)$
3. $(x \,\&\, y) \,/\, z \dashv\vdash C(x, y, z)$
4. $x \,/\, (y \,\&\, z) \dashv\vdash D(x, y, z)$

References

[Gen35] Gerhard Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935. English translation in M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, pages 68–131, North-Holland, 1969.

[Lam58] Joachim Lambek. The mathematics of sentence structure. *The American Mathematical Monthly*, 65(3):154–170, 1958.