Introduction to Natural Deduction

Carl Pollard

November 8, 2011

Logics for Linguistics

Many different kinds of logic are directly applicable to formalizing theories in syntax, semantics, formal pragmatics, and computational linguistics. We'll consider:

- linear logic (LL)
- positive intuitionistic propositional logic (PIPL)
- typed lambda calculus (TLC)
- higher order logic (HOL)

To explain these, we first introduce a kind of **proof theory** called **(sequent-style) natural deduction**, **ND** for short.

What is Proof Theory?

- **Proof theory** is the part of logic concerned with purely **syntactic** methods for determining whether a **formula** is deducible from a **collection** of formulas.
- Here 'syntactic' means that we are only concerned with the **form** of the formulas, not their semantic interpretation. (The part of logic concerned with that is **model theory**).
- What counts as a 'formula' varies from one proof theory to the next. Usually they are certain strings of symbols.
- What counts as a 'collection' also varies from one proof theory to the next.
- In some proof theories, collections are taken to be sets; in others, strings. In the proof theory we will be concerned with, they will be taken to be **finite multisets**.

Finite Multisets

- Roughly speaking, finite multisets are a sort of compromise between strings and finite sets:
  - They are stringlike because **repetitions matter**.
  - But they are setlike because **order does not matter.**
- Technically, for any set S, a finite S-multiset is an equivalence class of S-strings, where two strings count as equivalent if they are permutations of each other.
- Alternatively, we can think of a finite S-multiset as a function from a finite subset of S to $\omega \setminus \{0\}$.
- So if we indicate multisets between square brackets, then $[A]$ is a different multiset from $[A, A]$, but $[A, B]$ and $[B, A]$ are the same multiset.

Formulas

- To define a proof theory, we first need a recursively defined set of **formulas**.
- The base of the recursion specifies some **basic** formulas.
- Then the recursion clauses tell how to get additional formulas using **connectives**.

Example: Formulas in Linear Logic (LL)

- The set of LL formulas is defined as follows:
  1. Any basic formula is a formula.
  2. If A and B are formulas, then so is $A \multimap B$.
  3. Nothing else is a formula.
- The connective $\multimap$ is called **linear implication** (informally called 'lollipop').
- We adopt the convention that $\multimap$ 'associates to the right', e.g. $A \multimap B \multimap C$ abbbreviates $A \multimap (B \multimap C)$, not $(A \multimap B) \multimap C$.
- As we'll see, $\multimap$ works much like the implication $\rightarrow$ of ordinary propositional logic, but with fewer options.

Note: Actually, there are many linear logics. The one we describe here, whose only connective is $\multimap$, is implicative intuitionistic linear propositional logic.

Application: Tectogrammar (1/4)

- LL is used in *linear grammar (LG)* frameworks, such as λ-grammar and abstract categorial grammar (ACG), to analyze natural-language **tectogrammatical structure**, also called **abstract syntax** or **syntactic combinatorics**.
- In LG, tectogrammatical structure ('tectogrammar' or simply 'tecto' for short) 'drives' semantic composition.
- Tecto is distinguished from **phenogrammatical structure**, also called **concrete syntax**.
- Phenogrammatical structure ('phenostructure' or simply 'pheno') is concerned with superficial matters of realization, such as word order and intonation.

Application: Tectogrammar (2/4)

- In the context of LG, the LL formulas are called **tectotypes**, or **abstract syntactic types**.
- The role played by the tectotypes in LG is analogous to the role of nonterminals in CFG: they can be thought of as names of syntactic categories of linguistic expressions.
- As we'll see, a LG requires many fewer rules than a CFG, because the 'combinatory potential' of a linguistic expression is encoded in its tectotype.

Application: Tectogrammar (3/4)

In a simple LG of English (ignoring details such as case, agreement, and verb inflectional form), we might take the basic tectotypes to be:

S: sentences

$\bar{\text{S}}$: '*that*-sentences'

NP: noun phrases, such as names

It: 'dummy pronoun' *it*

N: common nouns

Application: Tectogrammar (4/4)

Some nonbasic tectotypes:

N $\multimap$ NP: determiners

N $\multimap$ N: attributive adjectives

S $\multimap$ $\bar{\text{S}}$: ‘complementizer’ *that*

NP $\multimap$ S: intransitive verbs

NP $\multimap$ NP $\multimap$ S: transitive verbs

NP $\multimap$ NP $\multimap$ NP $\multimap$ S: ditransitive verbs

NP $\multimap$ $\bar{\text{S}}$ $\multimap$ S: sentential-complement verbs

Contexts

- A finite multiset of formulas is called a **context**.
- Careful: this is a distinct usage from the notion of context as linguistically relevant features of the situation in which an expression is uttered.
- We use capital Greek letters (usually Γ or Δ) as metavariables ranging over contexts.

Sequents

- An ordered pair $\langle \Gamma, A \rangle$ of a context and a formula is called a **sequent**.
- Γ is called the **context** of the sequent and A is called the **statement** of the sequent.
- The formula occurences in the context of a sequent are called its **hypotheses** or **assumptions**.

What the Proof Theory Does

- The proof theory recursively defines a set of sequents.
- That is, it recursively defines a relation between contexts and formulas.
- The relation defined by the proof theory is called **deducibility**, **derivability**, or **provability**, and is denoted by $\vdash$ (read ‘deduces’, ‘derives’, or ‘proves’).

Sequent Terminology

- The metalanguage assertion that $\langle \Gamma, A \rangle \in \vdash$ is usually written $\Gamma \vdash A$.
- Such an assertion is called a **judgment**.
- The symbol '$\vdash$' that occurs between the context and the statment of a judgment is called 'turnstile'.
- If Γ is empty, we usually just write $\vdash A$.
- If Γ is the singleton multiset with one occurrence of B, we write $B \vdash A$.
- Commas in contexts represent multiset union, e.g. if $\Gamma = A, B$ and $\Delta = B$, then $\Gamma, \Delta = A, B, B$.

Proof Theory Terminology

- The proof theory itself is a recursive definition.
- The base clauses of the proof theory are called **axioms**, and the recursion clauses are called **(inference) rules**.
- Axioms are just certain judgments.
- Rules are metalanguage conditional statements, whose antecedents are conjunctions of judgments and whose consequent is a judgment.
- The judgments in the antecedent of a rule are called the **premisses** of the rule, and the consequent is called the **conclusion** of the rule.
- Rules are notated by a horizontal line with the premisses above and the conclusion below.

Axioms of (Pure) Linear Logic

- The proof theory for (pure) LL has one schema of axioms, and two schemas of rules.
- The axiom schema, variously called Refl (Reflexive Axioms), Hyp (Hypotheses), or simply Ax (Axioms) looks like this:

$$A \vdash A$$

- To call this an axiom **schema** is just to say that upon replacing the metavariable A by any (not necessarily basic) formula, we get an axiom, such as

$$\mathrm{NP} \vdash \mathrm{NP}$$

Note: In LG, hypotheses play a role analogous to that of 'traces' in frameworks such as GB and HPSG.

Rules of Linear Logic

- Modus Ponens, also called $\multimap$-Elimination:

$$\frac{\Gamma \vdash A \multimap B \qquad \Delta \vdash A}{\Gamma, \Delta \vdash B} \multimap\mathrm{E}$$

- Hypothetical Proof, also called $\multimap$-Introduction:

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B} \multimap\mathrm{I}$$

- Notice that Modus Ponens **eliminates** the connective $\multimap$, in the sense that there is an occurrence of $\multimap$ in one of the premisses (called the **major** premiss; the other premiss is called the **minor** premiss).
- Whereas, Hypothetical Proof **introduces** $\multimap$, in the sense that there is an occurrence of $\multimap$ in the conclusion but not in the (single) premiss.
- Pairs of rules that eliminate and introduce connectives are characteristic of the natural-deduction style of proof theory.

Theorems of a Proof Theory

- If in fact $\Gamma \vdash A$, then we call the sequent $\langle \Gamma, A \rangle$ a **theorem** (in the present case, of linear logic).
- It is not hard to see that $\Gamma \vdash A$ if and only if there is a **proof tree** whose root is labelled with the sequent $\langle \Gamma, A \rangle$.
- Here, by a proof tree we mean an ordered tree whose nodes are labelled by sequents, such that
  - the label of each leaf node is the sequent of an axiom; and
  - the label of each nonleaf node is the sequent of the conclusion of a rule such that the sequents of the premisses of the rule are the labels of the node's daughters.

Proof Tree Notation

- In displaying a proof tree, the root appears at the bottom and the leaves at the top (so from a logician's point of view, linguist's trees are upside down).
- Even though technically the labels are sequents, we conventionally write the corresponding judgments (metalanguage assertions that the sequents are deducible).

- Instead of edges connecting mothers to daughters as in linguist's trees, we write horizontal lines with the label of the mother below and the labels of the daughters above (just as in inference rules).
- Sometimes as a mnemonic we label the horizontal line with the name of the rule schema that was instantiated there.

The Simplest Proof Tree

- The simplest possible proof tree in linear logic has just one leaf, which is also the root.
- In this case the only option is for the node to be labelled by a Hypothesis, e.g.:

$$\mathrm{NP} \vdash \mathrm{NP}$$

- Intuitively, this can be thought of as: suppose you had an NP; then, sure enough, you'd have an NP.
- Don't worry if this doesn't seem to make any sense yet; it will become clear what this means when we use an elaborated form of Hypothesis (namely, trace) in a linguistic analysis.

A (Very) Slightly Less Trivial Proof Tree

$$\frac{\mathrm{NP} \vdash \mathrm{NP}}{\vdash \mathrm{NP} \multimap \mathrm{NP}}\,{\multimap}\mathrm{I}$$

Another Proof Tree (Type Raising)

$$\cfrac{\cfrac{\cfrac{\mathrm{NP} \vdash \mathrm{NP} \qquad \mathrm{NP} \multimap \mathrm{S} \vdash \mathrm{NP} \multimap \mathrm{S}}{\mathrm{NP}, \mathrm{NP} \multimap \mathrm{S} \vdash \mathrm{S}}\,{\multimap}\mathrm{E}}{\mathrm{NP} \vdash (\mathrm{NP} \multimap \mathrm{S}) \multimap \mathrm{S}}\,{\multimap}\mathrm{I}}{\vdash \mathrm{NP} \multimap (\mathrm{NP} \multimap \mathrm{S}) \multimap \mathrm{S}}\,{\multimap}\mathrm{I}$$

Note: Type Raising plays an important role in the analysis of quantificational noun phrases, topicalization, focus constructions, etc.