# Logical Grammar: Course Introduction

Carl Pollard

Department of Linguistics
Ohio State University

July 4, 2011

## What this Course is About

- Using basic tools of **mathematical logic** to construct theories about natural language (NL).
- More specifically: we use logic to write formally precise grammars of an especially simple kind, called **linear grammars**, focusing (for this course) on **syntax**, **semantics**, and the **interface** between them.

## Linear Grammar

**Linear Grammar (LG)** is a practical framework for linguistic analysis influenced by three traditions in linguistic theory:

- **categorial grammar (CG)**, a kind of syntactic analysis founded by Joachim Lambek (late 1950s) that treats lexical entries and grammar rules, respectively, as axioms and inference rules of a **proof theory**.
- **Montague semantics**, founded by Richard Montague (late 1960s), influenced by earlier philosophical logicians Frege (1892), Carnap (1947), and and Kripke (1963). Uses **type theory** (Church 1940, Henkin 1950) to analyze sentence meanings.
- **Dynamic semantics**, founded by Kamp (DRT, 1981), Heim (FCS, 1982), and others based on philosophical ideas of Stalnaker and Lewis (1960s and 1970s) about the role of context in the interpretation of multi-sentence discourses.

## Sources of Linear Grammar (1/2)

LG is a synthesis based on recent developments in all three of these traditions:

- **curryesque** CG, which analyzes syntax using **linear logic** (Girard 1987). Inspired by programmatic ideas of Curry (1961) and technical innovations of Oehrle (1994).

  Includes de Groote's (2001) **abstract categorial grammar (ACG)** and Muskens' (2003, 2007) **lambda grammar**.

- **hyperintensional semantics**, a kind of type-theoretic semantics which proposes a more **fine-grained** analysis of sentence meaning than Montague's.

  An early form was Thomason's (1980) **intentional semantics**. More recent avatars are Muskens (2005) and Pollard (2008).

## Sources of Linear Grammar (2/2)

- **Type-theoretic dynamic semantics**, which extends Montague's type-theoretic methods to analyze the kinds of discourse phenomena analyzed by DRT and FCS.

  Muskens (1994, 1996) pioneered this approach. More recent proposals are Beaver (2001), de Groote (2006), van Eijck and Unger (2010), and Martin and Pollard (2010, 2011).

## The Tools

The logical tools employed will all be introduced or reviewed in the course. They are:

- A certain style of proof theory, namely **sequent-style natural deduction**.
- Two simple kinds of propositional logic, namely:

  **implicative intuitionistic linear propositional logic**, here just called **linear logic (LL)**, and

  **positive intuitionistic propositional logic (PIPL)**
- **typed lambda calculus (TLC)**, an elaboration of PIPL
- **higher order logic (HOL), also called type theory** (here these two terms are more or less synonyous), an elaboration of TLC.

## Tentative Syllabus (1/3)

Suggested readings and these slides are at:

`http://www.coli.uni-saarland.de/courses/logical-grammar`

**Day 1** (Monday, July 4):

Course Overview; Sequent-Style Natural Deduction for LL and PIPL

Slides: introsl.pdf, ndsl.pdf
Reading: nd.pdf
Suggested background reading: crouch-genabith.pdf

**Day 2** (Wednesday, July 6):

Typed Lambda Calculus and the Curry-Howard Correspondence

Slides: tlcsl.pdf
Reading: tlc.pdf

## Tentative Syllabus (2/3)

**Day 3** (Friday, July 8):

Higher Order Logic and Hyperintensional Semantics

Slides: holsl.pdf, hypersl.pdf
Readings: hol.pdf, pollard2008.pdf

**Day 4** (Monday,July 11):

Linear Grammar Basics: Lexicon, Rules, Traces, Case, Modification and Predication

Slides: lgsl.pdf
Reading: mihalicek-pollard.pdf, pred.pdf
Suggested background reading: oehrle1994.pdf, degroote-acg.pdf, muskens-acl13.pdf, muskens2007.pdf

## Tentative Syllabus (3/3)

**Day 5** (Wednesday, July 13):

Linear Grammar Topics: Dummies, Raising, Verb Inflection, Agreement

Slides: lgsl.pdf (continued)
Reading: control.pdf

**Day 6** (Friday, July 15):

More Linear Grammar Topics: Nonlogical Rules, Quantifier Scope, Control, *Tough*-Movement
Slides: morelgsl.pdf
Reading: udc.pdf

## Intellectual Background of LG

The main ideas that LG is based on are drawn from the following people:

- Haskell Curry (a mathematical logician)
- Joachim Lambek (a mathematician)
- Richard Montague (a philosophical logician)
- Richard Oehrle (a theoretical and computational linguist)

## Curry

In a 1948 lecture, published in expanded form in 1961, Curry proposed that a linguistic expression should be analyzed as consisting of:

1. a **phenogrammatical** component: specifies the expression's superficial form
2. a **tectogrammatical** component: specifies the the expression's combinatory potential
3. a **semantic** component: specifies the expression's meaning

## The Phenogrammatical Component

- usually abbreviated to just **pheno**
- Corresponds roughly to what computer scientists sometimes call **concrete syntax**
- Also corresponds roughly to what linguists call **phonology**, broadly construed to include word order and nonsegmental (or prosodic) aspects
- relates to what the expression sounds like (or in the case of sign language, looks like)

## The Tectogrammatical Component

- usually abbreviated to just **tecto**
- Corresponds roughly to what computer scientists sometimes call **abstract syntax**
- Also corresponds roughly to what linguists call **syntactic category**.
- Relates to what other expressions the expression can combine with, and what results from the combination

## Lambek

- Invented his **syntactic calculus** in 1958, later called the **Lambek calculus**.
- A Lambek calculus is a grammar written in the form of a **logical proof system**.
- The role of linguist's trees is taken over by **proof trees**.
- Words correspond to **axioms**.
- Grammar rules are replaced by logical **inference rules**.
- Well-formed linguistic expressions correspond to **theorems** of the proof system.
- Unlike earlier forms of categorial grammar (CG) due to Ajdukiewicz and Bar-Hillel), the Lambek calculus makes (crucial!) use of the rule of **hypothetical proof**, which we will explain soon.

## Montague (1/2)

- In late 1960's, originated a style of CG influenced by ideas drawn from the philosophical logicians Gottlob Frege, Rudolph Carnap, Saul Kripke, and others.
- A Montague grammar recursively defines a set of triples, each of which consists of a **word string**, a **syntactic type**, and a typed lambda calculus term (TLC) denoting a meaning (often a function).
- In retrospect, we can relate Montague's string to Curry's pheno, and Montague's syntactic type to Curry's tecto.
- Some of the triples (**lexical entries**) are given, while the **rules** of the grammar produce new triples from old ones.
- Unlike Lambek calculus, Montague's CG was primitive (like Ajdukiewicz-Bar Hillel CG) in the sense of not having a rule like Hypothetical Proof.

## Montague (2/2)

- Each rule include ‘recipes’ specifying how to construct the string and meaning of a new expression, respectively, from the strings and meanings of the expression’s immediate constituents.
- The operation involved in constructing the new expression’s string is usually **concatenation**.
- The operation involved in constructing the new expression’s meaning is usually **function application**.
- On the semantic side, this last point is a version of Frege’s notion of semantic **compositionality**.

## Oehrle

- In the mid-to-late 1980's, categorial grammarians (such as van Benthem, Moortgat, Morrill) had the idea of combining Lambek calculus with Montague grammar.
- Within this setting, Oehrle introduced three technical innovations.
- The first was to replace the Lambek calculus with a simpler logic, namely **linear logic**.
- The second was to allow phenos to be not just strings, but also (possibly higher-order) functions over strings.
- The second innovation involved using TLC terms to denote phenos (not only for meanings as in Montague grammar).
- Oehrle's third innovation was a particular technique for analyzing quantified noun phrases ('quantifier lowering via $\beta$-reduction'), which we'll explain in due course.

## A First Glimpse of Linear Grammar (1/2)

- We write an LG to cover a **fragment** of an NL (as in Montague grammar).
- An LG recursively defines a set of triples called **signs**.
- Each sign specified by the LG is taken to represent one of the expressions of the NL in question.
- The three components of each sign correspond to the pheno, tecto, and meaning of the expression it represents.
- Some of these triples, called **lexical entries**, represent (syntactic) words.
- The set of lexical entries, called the **lexicon** is given 'in advance' and forms the base of the recursive definition.

## A First Glimpse of Linear Grammar (2/2)

- The recursive part of the definition is in the form of a few (at first, just two) **rules**, with whose help we can analyze more complex expressions.
- The lexical entries and rules can be thought of, respectively, as axioms and inference rules of a natural-deduction proof system.
- The signs specified by the grammar can be thought of as the theorems proved by the proof system.

## LG Signs

LG signs are notated in the form

$$\vdash \phi; \tau; \sigma$$

where:

- $\phi$ is a term in a typed lambda calculus (more precisely, a higher-order logical theory) called the **pheno calculus**. A pheno term is interpreted as a string of phonological words, or as a higher-order function over such strings.
- $\tau$ is a formula of linear logic, the **tecto type** of the sign. Intuitively, a tecto type can be thought of as the name of a syntactic category.
- $\sigma$ is a term in a typed lambda calculus (again, more precisely, a higher-order theory) called the **semantic calculus**. A semantic term is interpreted as an entity or a proposition, or as a higher-order function over such things.

## A Simple LG Lexical Entry

$$\vdash \text{pig}; \mathrm{N}; \mathsf{pig}$$

In this lexical entry:

- pig is a constant (of type s) of the pheno calculus, which denotes a string (of length 1) whose only element is the phonological word pIg/.
- N is a (basic) tectotype, namely the type of nouns.
- $\mathsf{pig}$ is a constant (of type $e \rightarrow p$) of the semantic calculus, which denotes a certain property, namely the function that maps any entity to the proposition that that entity is a pig.

## Traces (1/2)

- Besides lexical entries, which will play the role of **nonlogical axioms** when we think of a grammar as a proof system, there are also **logical axioms**, called **traces**.
- Traces play an important role in the analysis of scope of quantificational NPs, and also in the analysis of constructions usually analyzed in terms of movement in MGG (such as constituent questions, relative clauses, and clefts).
- Intuitively, a trace can be thought of as a 'hypothetical' syntactic consituent.

## Traces (2/2)

Here's a typical trace (more specifically, an NP-trace):

$$s; \mathrm{NP}; x \vdash s; \mathrm{NP}; x$$

Here:

- $s$ is a variable (of type s) of the pheno calculus, which denotes a hypothetical string
- NP is a (basic) tectotype, namely the type of noun phrases
- $x$ is a variable (of type e) of the semantic calculus, which denotes a hypothetical entity.

## LG Rules

- Besides axioms (lexical entries and traces), an LG also has **rules**.
- In its simplest form, LG has only two rules, corresponding to the two inference rules of linear logic, Modus Ponens and Hypothetical Proof.
- Modus Ponens corresponds roughly to Merge in MGG: it is used to combine two signs, a ‘function’ sign and its ‘argument’.
- Hypothetical Proof corresponds roughly to Move in MGG: it is used to ‘discharge’ a hypothetical sign, similar to ‘binding a trace with an empty operator’ in MGG.
- We postpone writing the rules down till after we introduce the relevant technical tools.