Computational Linguistics II

— Grammars, Algorithms, Statistics —

Dan Flickinger

Oslo and Stanford Universities

`danf@csli.stanford.edu`

Tore Langholm

Universitetet i Oslo

`torel@ifi.uio.no`

Stephan Oepen

Oslo and Stanford Universities

`oe@csli.stanford.edu`

So, Why (Computational) Grammar?

Wellformedness

- *Kim was happy because* ____ *passed the exam.*
- *Kim was happy because* ____ *final grade was an A.*
- *Kim was happy when she saw* ____ *on television.*

Meaning

- *Kim gave Sandy a book.*
- *Kim gave a book to Sandy.*
- *Sandy was given a book by Kim.*

Ambiguity

- *I saw the astronomer with the telescope.*
- *Have her report on my desk immediately!*

What We Are About to Do (and Why)

Course Outline

- Extend understanding of (natural) language as a system of rules;
- learn how to *formalize* grammars through typed feature structures;
- design and implement common algorithms and probabilistic models;
- solve weekly exercises: immediate gratification (risk of late hours).

Three Interacting Components

- **grammar engineering** formalize linguistic theories with complex interactions of multiple phenomena; implementation and debugging;
- **processing** understand common parsing algorithms; unification of feature structures; implement an efficient unification-based parser;
- **probabilistic models** capture relative frequency of (competing) phenomena; approximate graded grammaticality or soft constraints.

Student Experimentation — Immediate Gratification

Some Applications of Computational Grammars

Machine Translation

- Traditional: analyse source to some degree, transfer, generate target.

Text ‘Understanding’

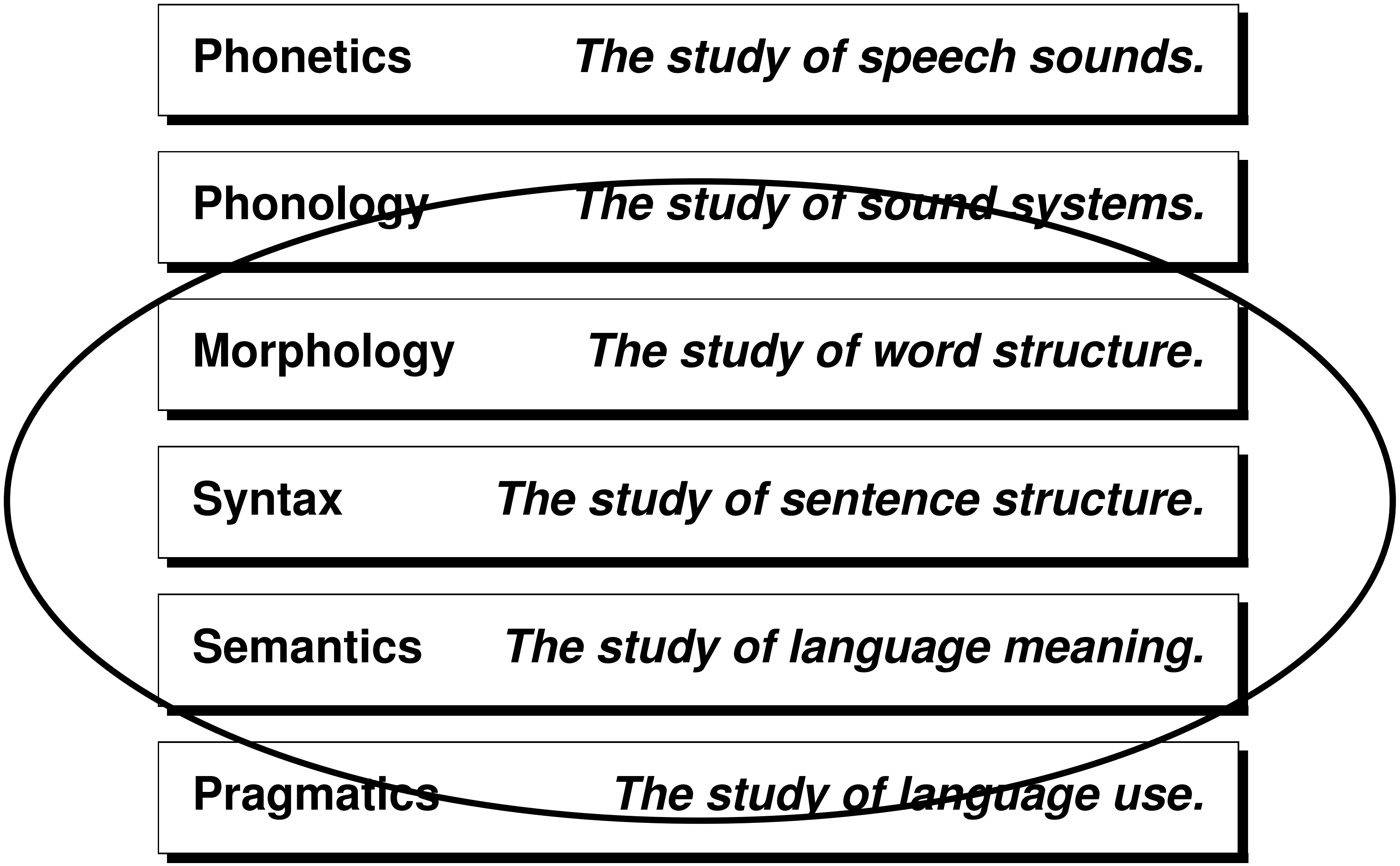
- Email auto- (or assisted) response: interpret customer requests;
- Semantic Web: annotate WWW with structured, conceptual data.

(Spoken) Dialogue Systems

Grammar & Controlled Language Checking

Summarization & Text Simplification

Some Areas of Descriptive Grammar



Phonetics	***The study of speech sounds.***
Phonology	***The study of sound systems.***
Morphology	***The study of word structure.***
Syntax	***The study of sentence structure.***
Semantics	***The study of language meaning.***
Pragmatics	***The study of language use.***

Grammar Engineering from a CS Perspective

Implementation Goals

- Translate linguistic constraints into specific formalism → formal model;
- computational grammar provides mapping between form and meaning;
- assign correct analyses to grammatical, reject ungrammatical inputs;
- parsing and generation algorithms: apply mapping in either direction.

Analogy to (Object-Oriented) Programming

- Computational system with observable behavior: immediately testable;
- typed feature structures as a specialized (OO) programming language;
- make sure that all the pieces fit together; revise – test – revise – test ...

The Linguistic Knowledge Builder (LKB)

General & History

- Specialized grammar engineering environment for TFS grammars;
- main developers: Copestake (original), Carroll, Malouf, and Oepen;
- open-source and binary distributions (Linux, Windows, and Solaris).

Grammar Engineering Functionality

- Compiler for typed feature structure grammars → wellformedness;
- parser and generator: map from strings to meaning and vice versa;
- visualization: inspect trees, feature structures, intermediate results;
- debugging and tracing: interactive unification, ‘stepping’, et al.

Why Common-Lisp for Implementation Exercises?

- Arguably most widely used language for ‘symbolic’ computation;
- easy to learn: extremely simple syntax; straightforward semantics;
- a rich language: multitude of built-in data types and operations;
- full standardization; Common-Lisp has been stable for a decade;
- LKB (experimentation environment) implemented in Common-Lisp;

→ for our purposes, (at least) as good a choice as any other language.

$$n! \quad \equiv \quad \begin{cases} 1 & \text{for } n = 0 \\ n \times (n-1)! & \text{for } n > 0 \end{cases}$$

```
(defun ! (n)
  (if (= n 0)
    1
    (* n (! (- n 1)))))
```

Course Organization

Comments on Background Literature

Formal Grammar and General NLP

- Sag, Ivan A. Tom Wasow, and Emily M. Bender: *Syntactic Theory. A Formal Introduction (2nd Edition).* Stanford, CA: CSLI Publications (2003);
- Jurafsky, Daniel and Martin, James H.: *Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Upper Saddle River, NJ: Prentice Hall (2000).

The Linguistic Knowledge Builder

- Copestake, Ann: *Implementing Typed Feature Structure Grammars.* Stanford, CA: CSLI Publications (2001).