

# Topics in Computational Linguistics — Grammar Engineering —

**Dan Flickinger**

CSLI Stanford & Saarland University

`danf@csli.stanford.edu`

**Stephan Oepen**

Universitetet i Oslo & CSLI Stanford

`oe@csli.stanford.edu`

<http://lingo.stanford.edu/courses/05/ge/>

# So, What is Computational Linguistics?

*... teaching computers our language.* (Alien Researcher)

*... the scientific study of human language—specifically of the system of rules and the ways in which they are used in communication—using mathematical models and formal procedures that can be realized and validated using computers; a cross-over of many disciplines.* (Stanford Professor)

*... a cornerstone of our pioneering .NET initiative and the operating systems of the future; innovative technology that will change our world.* (President of US-Based Software Company)

*... a sub-discipline of our Artificial Intelligence programmes.*

(CMU Professor)



# What About (Computational) Grammar Then?

## 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

- Develop understanding of (natural) language as a system of rules;
- learn how to *formalize* grammars through typed feature structures;
- adapt and develop sequence of trivial HPSG grammars in LKB;
- solve weekly excercises: immediate gratification (risk of late hours).

## Why Computational Grammars

- **research** formalize linguistic theories with complex interactions of language phenomena; identify cross-language generalizations;
- **education** teach frameworks or analyses in formal morphology, syntax, and semantics; support student experimentation;
- **applications** embed grammar-based natural language analysis or generation in research prototypes and commercial applications.



# Student Experimentation — Immediate Gratification



STANFORD — 4-JAN-05 (oe@csli.stanford.edu)

Computational Linguistics: Grammar Engineering (5)

# 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.*

**Prgamatics**      *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.



# Course Organization



---

STANFORD — 4-JAN-05 (oe@csli.stanford.edu)

Computational Linguistics: Grammar Engineering (10)

# Comments on Background Literature

## Formal Syntax

- Sag, Ivan A. Tom Wasow, and Emily M. Bender: *Syntactic Theory. A Formal Introduction (2<sup>nd</sup> Edition)*. Stanford, CA: CSLI Publications (2003);
- Pollard, Carl and Sag, Ivan: *Head-Driven Phrase Structure Grammar*. Chicago, IL and London, UK: University of Chicago Press (1994).
- Shieber, Stuart: *An Introduction to Unification-Based Approaches to Grammar*. Stanford, CA: CSLI Publications (1986).

## The Linguistic Knowledge Builder

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

