Topics in Computational Linguistics — Parsing and Generation —

Stephan Oepen
Dan Flickinger

Center for the Study of Language and Information
Stanford University — Spring Quarter 2003

{oe|dan}@csli.stanford.edu

http://lingo.stanford.edu/courses/03/pg/
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)
And What is Parsing and Generation Then?

... arguably the most traditional stream of computational linguistics research  
(Martin Kay)

The Grammar of Norwegian

```
S → NP VP  { VP ( NP ) }
VP → V NP  { V ( NP ) }
VP → VP PP  { PP ( VP ) }
PP → P NP  { P ( NP ) }
NP → "Per"  { Peter }
NP → "Jon"  { John }
NP → "Oslo"  { Oslo }
V → "ser"  { λbλa see ( a, b ) }
P → "i"  { λdλc in ( c, d ) }
```

![Parse tree for "Jon ser Per i Oslo"](image_url)
Meaning Composition (Overly Simplified)

\{ \text{in (see (John, Peter), Oslo)} \}

\{ \lambda_a \text{ in (see (a, Peter), Oslo)} \}

\{ \lambda_a \text{ see (a, Peter)} \}

\{ \lambda_c \text{ in (c, Oslo)} \}

\{ \lambda_b \lambda_a \text{ see (a, b)} \}

\{ \text{Peter} \}

\{ \lambda_d \lambda_c \text{ in (c, d)} \}

\{ \text{Oslo} \}

\text{Jon}

\text{VP} \rightarrow \text{V NP} \quad \{ \text{V (NP)} \}
What We Are About to Do (and Why)

Course Outline

• familiarize with common parsing, unification, and generation algorithms;
• learn how to efficiently process context-free and unification grammars;
• implement, adapt, and debug core NLP procedures in Common-Lisp;
• solve weekly excercises: immediate gratification (at risk of late hours).

Why Grammar-Based Parsing and Generation

• research formalize linguistic knowledge; use computer programs to validate hypotheses: parsers and generators relate form to meaning;
• education teach analyses in formal grammar; experience ubiquitous ambiguity in natural languages; support student experimentation;
• applications embed grammar-based natural language analysis in research prototypes and natural language processing applications.
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.

\[
\begin{align*}
\text{defun } \! (n) &= \\
&= \begin{cases} 
1 & \text{for } n = 0 \\
 n \times (n - 1)! & \text{for } n > 0 
\end{cases}
\end{align*}
\]
Comments on Relevant Literature (1 of 2)

Common-Lisp


Parsing and Generation

Comments on Relevant Literature (2 of 2)

**Unification-Based Parsing**


