Why CCG?

- CCG is mildly context-sensitive (like TAG)
  - Captures crossing dependencies, but is still efficiently parseable
- CCG has a flexible constituent structure:
  - Simple, unified treatment of extraction and coordination (verbs have the same lexical category in main clauses as in relative clauses, wh-questions, right-node-raising, etc.)
  - Allows incremental processing (psycholinguistic motivation)
  - Integration with prosody/information structure
- CCG has a transparent syntax-semantics interface
- CCG requires no traces or null elements

Overview

- Part I: The basics (local dependencies)
  - The ingredients: categories, rules, derivations
  - Simple syntactic phenomena: modification, simple coordination
  - Syntactic derivations and semantic interpretations
- Part II: The interesting stuff (long-range dependencies)
  - binding theory, control and raising
  - extraction and coordination
  - scrambling

CCG: The machinery

- Categories specify subcat lists of words/constituents.
- Combinatory rules specify how constituents can combine.
- The lexicon specifies which categories a word can have.
- Derivations spell out process of combining constituents.
CCG rules

These are really rule schemas, ie.:

Function application:

\[
\begin{align*}
X &\rightarrow Y \quad Y \rightarrow X \\
\end{align*}
\]

Type raising:

\[
\begin{align*}
X &\rightarrow T \setminus (T \setminus X) \\
X &\rightarrow T \setminus (T \setminus X) \\
\end{align*}
\]

Function composition:

\[
\begin{align*}
X/Y &\rightarrow X/Z \\
Y/Z &\rightarrow X/Z \\
\end{align*}
\]

Coordination:

\[
\begin{align*}
X \text{ conj } X \\
\end{align*}
\]

These are order-preserving rules (context-free only). More later...

CCG derivations

\[
\begin{array}{ccc}
\text{I like coffee} \\
\text{NP (S NP) NP NP}
\end{array}
\]

Type-raising and composition permit alternative derivations.

This is an example of incremental derivation.

Here, I like is a constituent.

(If you don’t like that, we’ll later see examples where that makes a lot more sense.)

But in general, not every substring can be a constituent.

The syntax-semantics interface

Every syntactic rule has a semantic interpretation:

Function application

\[
\begin{align*}
X/Y &\rightarrow X \\
Y &\rightarrow X \\
\end{align*}
\]

Function composition

\[
\begin{align*}
X/Y &\rightarrow X/Z \\
Y/Z &\rightarrow X/Z \\
\end{align*}
\]

Type-raising

\[
\begin{align*}
X &\rightarrow a \\
T &\rightarrow (T \setminus X) \\
\end{align*}
\]

Derivations and interpretations

- Syntactic derivations...
  - describe constituency
  - account for unbounded dependencies that arise through extraction and coordination (but don’t require traces to do so)
  - are not a level of representation in the theory, just a record of the process which builds the interpretation (the interface between spoken form and its meaning)

- Semantic interpretations...
  - account for bounded dependencies that arise in binding, raising and control (c-command is defined here)

Part II: The interesting parts

A. Bounded dependencies
Binding theory in CCG

- Dominance and c-command are:
  - defined over predicate-argument structure
  - specified by lexical entries

- Reflexives and reciprocals have “anaphoric” (+ANA) categories:
  (corresponding to pro-terms in the predicate-argument structure)
  - himself: NP_{ANA, lex, self}
  - each other: NP_{ANA, other}

- +ANA arguments of verbs have different semantics:
  - see: (S[NS]\(\lambda x. x \mid y\)) \((\lambda x. y \mid x)\)
  - talk: (S[NS]\(\lambda x. x \mid y\)) \((\lambda y. x \mid y)\)

B. Unbounded dependencies

Another example

<table>
<thead>
<tr>
<th>whom</th>
<th>John</th>
<th>persuades</th>
<th>rarely</th>
<th>to go</th>
</tr>
</thead>
<tbody>
<tr>
<td>N(\lambda x. (\lambda y. y))</td>
<td>N(\lambda x. (\lambda y. y))</td>
<td>N(\lambda x. (\lambda y. y))</td>
<td>N(\lambda x. (\lambda y. y))</td>
<td>N(\lambda x. (\lambda y. y))</td>
</tr>
</tbody>
</table>

This requires crossing composition rules:

- Forward crossing composition: \(X/Y \not\Rightarrow X/Z\)
- Backward crossing composition: \(Y/Z \not\Rightarrow X/Z\)

Tough movement

easy to please

<table>
<thead>
<tr>
<th>S(\lambda x. (\lambda y. y))</th>
<th>S(\lambda x. (\lambda y. y))</th>
<th>S(\lambda x. (\lambda y. y))</th>
<th>S(\lambda x. (\lambda y. y))</th>
</tr>
</thead>
</table>

Parasitic gaps

- also specified in the lexical entry of the verb:
  - tries: \((\lambda x. \lambda y. \text{try}(\lambda x)(\lambda y)(\lambda y))\)
  - persuades: \((\lambda x. \lambda y. \text{persuade}(\lambda x)(\lambda y)(\lambda y))\)

- Modals and auxiliaries are like subject control:
  - might: \((\lambda x. \lambda y. \text{might}(\lambda x)(\lambda y)(\lambda y))\)

Control and raising

- Use type-raising and composition to form “incomplete” constituents.
- Wh-words subcategorize for “incomplete” constituents:

\[
\begin{array}{c}
\text{that} \\
\text{John} \\
\text{buys}
\end{array}
\begin{array}{c}
\text{NP} \\
\text{S[NS]} \\
\text{S[NS]} \\
\text{S[NS]}
\end{array}
\Rightarrow 
\begin{array}{c}
\text{NP} \\
\text{S[NS]} \\
\text{S[NS]} \\
\text{S[NS]}
\end{array}
\]

Questions

<table>
<thead>
<tr>
<th>does</th>
<th>he</th>
<th>seem</th>
<th>to</th>
<th>like</th>
<th>coffee?</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S[NS](\lambda x. (\lambda y. y)))</td>
<td>(S[NS](\lambda x. (\lambda y. y)))</td>
<td>(S[NS](\lambda x. (\lambda y. y)))</td>
<td>(S[NS](\lambda x. (\lambda y. y)))</td>
<td>(S[NS](\lambda x. (\lambda y. y)))</td>
<td>(S[NS](\lambda x. (\lambda y. y)))</td>
</tr>
</tbody>
</table>

Wh-extraction

Use type-raising and composition to form “incomplete” constituents.

- This requires a new combinator rule:
  - Backward crossing substitution: \(Y/Z \Rightarrow X/Z \not\Rightarrow X/\lambda s.f(x)\)
Right node raising

- Use type-raising and composition to form "incomplete" constituents.
- RNR is just coordination of such "incomplete" constituents:

\[
\begin{array}{c}
\text{John buys and Mary sells (coffee)} \\
\text{\( \frac{\text{NP}}{\text{S}(\text{S}[\text{NP}] / \text{NP})} \)} \text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \\
\text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \\
\end{array}
\]

Scrambling

Verb has standard category. Use type-raising and (crossing) composition:

\[
\begin{array}{c}
\text{Er gab dem Polizisten eine Blume} \\
\text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \\
\text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \\
\end{array}
\]

Argument cluster coordination

- Use type-raising and composition to combine the argument clusters.

\[
\begin{array}{c}
\text{I give her flowers and him whisky} \\
\text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \\
\text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \\
\end{array}
\]

Long-distance scrambling

Relative clause: eine Karte, die er der Presse hatte zustecken lassen.

\[
\begin{array}{c}
\text{a map, which he (to) the press has pass on let.} \\
\text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \\
\text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \\
\end{array}
\]

Canonical form: er hatte eine Karte der Presse zustecken lassen.

\[
\begin{array}{c}
\text{he has a map (to) the press pass on let.} \\
\text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \\
\text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \text{\( \frac{\text{NP}}{\text{S}[\text{NP}] / \text{NP}} \)} \\
\end{array}
\]

What I didn’t get to...

- Extensions: eg. Multi-modal CCG (Baldridge)
- CCG corpora:
  - English CCGbank: translation of the Penn Treebank into CCG (my PhD).
  - German CCGbank: translation of German Tiger corpus into CCG.
  - Turkish: translation of Turkish dependency corpus into CCG (Cakici).
- CCG parsers:
  - StatCCG: generative statistical CCG parser (my PhD)
  - Clark and Curran: loglinear CCG parser and supertagger
  - sem2CCG: generates DRSs from C&C's output (Bos).
  - OpenCCG: open source implementation for parsing, generation etc.

More on CCG

M. Steedman
The Syntactic Process
MIT press, 2000

M. Steedman
Surface Structure and Interpretation
MIT press, 1996

http://groups.inf.ed.ac.uk/ccg