Grammar Formalisms
Combinatorial Categorial Grammar (CCG)

Laura Kallmeyer, Timm Lichte, Wolfgang Maier
Universität Tübingen

20.06.2007

Outline

What is CCG?

- Combinatory Categorial Grammar (CCG) is a grammar formalism equivalent to Tree Adjoining Grammar, i.e.
  - it is lexicalized
  - it is parsable in polynomial time
  - it can capture cross-serial dependencies
- Just like TAG, CCG is used for grammar writing
- CCG is especially suitable for statistical parsing

In a nutshell

- Categories: specify subcat lists of constituents
- Lexicon: assigns categories (and semantics) to words
- Combinatory rules: specify how constituents can combine
- Derivations: records the derivation history, i.e. how consituents are combined
- Syntax-semantics interface: Categories and rules are paired with a semantic counterpart

CCG categories

There are two kinds of categories:

- Simple categories: S, NP
- Complex categories: Functions returning a result when combined with an **argument**
  - Intransitive verb: S**NP**
  - Transitive verb: (S\NP)/**NP**
  - Adverb: (S\NP)**(S\NP)**
  - Prepositional phrase: (NP\NP)/**NP**

The lexicon

- The CCG lexicon assigns categories to words, i.e. it specifies which categories a word can have.
- Furthermore, the lexicon specifies the semantic counterpart of the syntactic rules, e.g.:
  love (S\NP)/NP $\lambda x \lambda y.loves' xy$
- Combinatory rules determine what happens with the category and the semantics on combination

Functional application

Functional application

- X/Y:f Y:a ⇒ X:fa (fwd functional application, >)
- Y:a X\Y:f ⇒ X:fa (bwd functional application, <)

- Combine a function with its argument:
  1. NP S**NP** → S
     Mary sleeps → Mary sleeps
  2. (S\NP)/**NP** NP → S**NP**
     likes Mary → likes Mary
     NP S**NP** → S
     John likes Mary → John likes Mary
- Direction of the slash indicates position of the argument with respect to the function

A CCG derivation using FA

- The combinatory rule used in each derivation step is usually indicated on the right of the derivation line
- Note especially what happens with the semantic information

$$\dfrac{\dfrac{\dfrac{John}{NP : John'} \quad \dfrac{loves}{(S\backslash NP)/NP : \lambda x \lambda y.loves' xy} \quad \dfrac{Mary}{NP : Mary'}}{S\backslash NP : \lambda y.loves' Mary' y}\,>}{S : loves' Mary' John'}\,<$$

More combinatory rules: Type-raising

Forward type-raising

- X:a ⇒ T/(T\X):$\lambda f.fa$ (>T)

- Type-raising turns an argument into a function (e.g. for case assignment)
  NP ⇒ S/(S**NP**) (nominative)
  NP ⇒ (S\NP)/((S\NP)/**NP**) (accusative)
- This must be used e.g. in the case of wh-movement
- Example with semantics follows

More combinatory rules: Functional composition

Functional composition

- X/Y:f Y/Z:g $\Rightarrow_B$ X/Z:$\lambda x.f(gx)$ (>B)

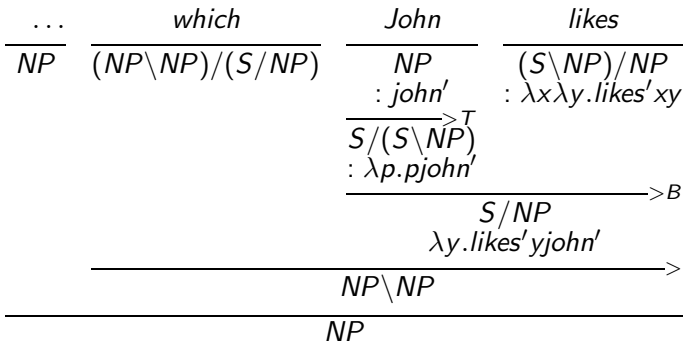
- Functional composition composes two complex categories (two functions):
  - (S\NP)/**PP** **PP** /NP ⇒ (S\NP) /NP
  - S/(**S\NP**) (**S\NP**) /NP ⇒ S /NP
- Example with semantics follows

Restriction

- Both type-raising and function composition are supposed to be only used when syntactically necessary
- Derivations respecting this restriction are called *normal-form* derivations

Functional composition and type-raising

An example using both of them:



Semantics with CCG

CCG offers a syntax-semantics interface.

- Every syntactic category and rule has a semantic counterpart
- The lexicon is used to pair words with syntactic categories and semantic interpretations:
  love (S\NP)/NP $\lambda x \lambda y.loves' xy$
- We have already seen example derivation with semantics

Raising and Control recap

Verbs that subcategorize for to-infinitives show differing properties with respect to their **semantic and syntactic influence on the subject** of the to-infinitives.

- Control verbs / Equi verbs (*try*, *persuade*)
- Raising verbs (*seem*, *expect*)

Control verbs recap

Control verbs establish the coreference between their subject/object and the unexpressed subject (PRO) of their sentential complement.
(PRO control)

(1) a. John tried [PRO to leave]. (subject control)

b. John persuaded him [PRO to leave]. (object control)

c. *There tries [PRO to be disorder after a revolution].

⇒ Control verbs assign semantic role to the controller!

Subject Control - CCG analysis

(2) John wants to leave. (subject control)

- Choose the categories:
  - *John* := NP:*john'*
  - *wants* := (S\NP)/(S_{TO}\NP):$\lambda p \lambda y.want'(p(ana'(y)))y$
  - *to leave* := (S_{TO}\NP):$\lambda x.leave' x$

Subject Control - CCG analysis

- The analysis:

$$\dfrac{\dfrac{\textit{John}}{\underset{:\,\textit{john}'}{NP}} \quad \dfrac{\dfrac{\textit{wants}}{\underset{:\,\lambda p\lambda y.\textit{want}'(p(\textit{ana}'(y)))y}{(S\backslash NP)/(S_{TO}\backslash NP)}} \quad \dfrac{\textit{to} \quad \textit{leave}}{\underset{:\,\lambda x.\textit{leave}'x}{(S_{TO}\backslash NP)}}}{\underset{:\,\lambda y.\textit{want}'(\textit{leave}'(\textit{ana}'(y))y)}{(S\backslash NP)}}>}{\underset{:\,\textit{want}'(\textit{leave}'(\textit{ana}'(\textit{john}'))\textit{john}')}{S}}<$$

Object Control - CCG analysis

(3) John persuaded him __ to leave (object control)

- Choose the categories:
  - *John* := NP
  - *him* := NP
  - *persuaded* := ((S\NP)/(S_{TO}\NP))/NP
    :$\lambda x\lambda p\lambda y.\textit{persuade}'(p(\textit{ana}'(x)))xy$
  - *to leave* := (S_{TO}\NP)

Object Control - CCG analysis

- The analysis:

$$\dfrac{\dfrac{\textit{John}}{\underset{:\,\textit{john}'}{NP}} \quad \dfrac{\dfrac{\dfrac{\textit{persuaded}}{\underset{:\,\lambda x\lambda p\lambda y.\textit{persuade}'(p(\textit{ana}'(x)))xy}{((S\backslash NP)/(S_{TO}\backslash NP))/NP}} \quad \dfrac{\textit{Mary}}{\underset{:\,\textit{mary}}{NP}}}{\underset{:\,\lambda p\lambda y.\textit{persuade}'(p(\textit{ana}'(\textit{mary}')))\textit{mary}'y}{(S\backslash NP)/(S_{TO}\backslash NP)}}> \quad \dfrac{\textit{to} \quad \textit{leave}}{\underset{:\,\lambda z.\textit{leave}'z}{S_{TO}\backslash NP}}}{\underset{:\,\lambda y.\textit{persuade}'(\textit{leave}'(\textit{ana}'(\textit{mary}')))\textit{mary}'y}{S\backslash NP}}>}{\underset{:\,\textit{persuade}'(\textit{leave}'(\textit{ana}'(\textit{mary}')))\textit{mary}'\textit{john}'}{S}}<$$

Raising verbs recap

Raising verbs determine case and agreement properties of the subject of the (non-finite) sentential complement. Semantically, however, the "raised" constituent is no immediate part of the argument structure of the raising verb.

(4) a. [John] seems [to leave]. (subject raising)
b. John expects [her to leave]. (object raising)
c. [There] seems [to be disorder after a revolution].
d. John expected [it to rain].

⇒ don't assign a semantic role to the raised constituent (raising of expletive *it/there*)

(5) John seems unhappy.
*John tries unhappy.

⇒ allow for **small clauses**

Subject raising - CCG analysis

(6) John seems to leave. (subject raising)

- Choose the categories:
  - *John* := NP:$john'$
  - *seems* := (S\NP)/(S$_{TO}$\NP):$\lambda p \lambda y.seem'(py)$
  - *to leave* := (S$_{TO}$\NP):$\lambda x.leave'x$

Subject raising - CCG analysis

- The analysis:

$$\dfrac{\dfrac{\textit{John}}{\underset{:\ john'}{NP}} \quad \dfrac{\dfrac{\textit{seems}}{\underset{:\ \lambda p \lambda y.seem'(py)}{(S\backslash NP)/(S_{TO}\backslash NP)}} \quad \dfrac{\textit{to} \quad \textit{leave}}{\underset{:\ \lambda x.leave'x}{(S_{TO}\backslash NP)}}}{\underset{\lambda y.seem'(leave'y)}{S\backslash NP}}\,{>}}{\underset{:\ seem'(leave'john)}{S}}\,{<}$$

WH-extraction

- Recap: In TAG, generally, for a verb with a WH extracted subject, a new tree is introduced
- How does CCG proceed?

Example for wh extraction

(7) The beer which John said Mary ordered

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\textit{beer}}{N} \quad \dfrac{\textit{that}}{(N\backslash N)/(S/NP)} \quad \dfrac{\dfrac{\textit{John}}{NP}}{S/(S\backslash NP)}\,{>^T} \quad \dfrac{\textit{said}}{(S\backslash NP)/S} \quad \dfrac{\dfrac{\textit{Mary}}{NP}}{S/(S\backslash NP)}\,{>^T} \quad \dfrac{\textit{ordered}}{(S\backslash NP)/NP}}{S/S}\,{>B} \quad \dfrac{}{S/NP}\,{>B}}{N\backslash N}\,{>}}{N}\,{<}$$

Alternative?

However, why not simply do this:

$$\frac{\ldots \quad \underline{John} \quad \underline{said} \quad \underline{Mary} \quad \underline{ordered}}{\dfrac{\dfrac{}{(S/NP)/(S/NP)} \quad \dfrac{}{S/NP}}{S/NP}\,{>}}$$

Alternative?

However, why not simply do this:

$$\frac{\ldots \quad \underline{John} \quad \underline{said} \quad \underline{Mary} \quad \underline{ordered}}{\dfrac{\dfrac{}{(S/NP)/(S/NP)} \quad \dfrac{}{S/NP}}{S/NP}\,{>}}$$

- Blow-up of the lexicon!
- We would need to introduce a category for each verb like *said* which would enable it to be used in this specific construction.
- The use of functional composition keeps degree of generality high

Cross-serial dependencies in CCG

- One of the most-cited reasons for using a formalism more powerful than CFG: Availability of cross-serial dependencies
- In CCG, a new kind of functional composition is needed:

Forward crossed composition

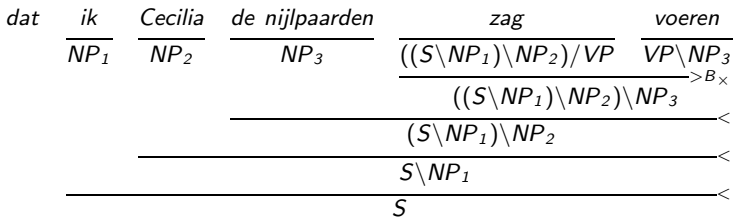
- $X/Y \quad Y\backslash Z \quad \Rightarrow_{B_\times} \quad X\backslash Z \qquad (>B_\times)$

An example for cross-serial dependencies

(8) dat Jan Cecilia de nijlpaarden zag voeren
that Jan Cecilia the hippos saw feed
that Jan saw Cecilia feed the hippos

The derivation follows...

An example for cross-serial dependencies



CCG and NLP kind of applications

- CCG has been extensively used for wide-coverage parsing.
- Since CCG derivations are binary, standard chart parsing techniques can be used
- Furthermore, research has been done on
  - Statistical models for treebank-based CCG parsing
  - Supertagging
  - Efficient organization of the lexicon
  - Creation of CCG lexica and corpora

  (among others)

The "problem" with lexicalized grammar formalisms

- Lexicalized Formalisms (such as TAG and CCG) aim at associating lexical items with the syntactical/grammatical information they contribute
- Many lexical items are assigned to the same structures which in turn share many common parts
  $\rightsquigarrow$ no structuring, redundancy!
- Redundancy is problematic in the context of
  - **grammar maintenance**: becomes difficult and error-prone, due to lack of abstraction
  - **parsing**: Efficient lexicon access is usually crucial for parsing speed (esp when dealing with big grammars)

Organization of the lexicon

Goal: Factorizing and/or hierarchizing common information in the lexicon and re-using it at different places.
Common technique for CCG: Inheritance-based lexicon

- Known from Attribute-Value-Grammar (AVG)
- Idea: add a tree-shaped type inheritance hierarchy of lexical categories
- Its items are grouped based on the information they share: Types inherit all of their properties to their subtypes.

Supertagging (TAG)

In the context of Tree-Adjoining Grammar (Bangalore, 1997):

- In practice, the time needed for parsing TAG depends on efficient tree selection
- This becomes especially important when using large-scale highly ambiguous grammars with many trees (XTAG)
- Supertagging: Prior to parsing, each lexical item of the input is tagged on a statistcial basis with an elementary tree

Supertagging (CCG)

This idea can also be applied on Combinatory Categorial Grammars (Clark, 2002)

- Words can possibly be assigned many different categories $\rightsquigarrow$ too much work for the parser (ambiguity, efficiency)
- Solution: Let (HMM-based) tagger assign categories beforehand ("almost parsing")

Creation of CCG resources

- The automatic creation of CCG grammars has been thoroughly investigated
- Treebanks can be used as the data source
- The similarity between CCG and CFG derivation eases grammar extraction
- Extracted grammars are used to develop statistical parsing models

References

Please see the reference list on the course webpage at
`http://www.sfb441.uni-tuebingen.de/emmy/gf/references.pdf`