Combinatory Categorial Grammar

Pavel Kalvoda

April 22, 2014

Overview

A note on terminology

Combinatory means combinational, related to combination, being able to combine or be combined, as in ‘combinatory logic’.

Categories & (Pure) Categorial Grammar

Phrase-structure grammar

Rules

$$(1)\ \mathsf{S} \rightarrow \mathsf{NP\ VP}$$
$$(2)\ \mathsf{NP} \rightarrow \mathsf{A\ N} \mid \mathsf{N}$$
$$(3)\ \mathsf{VP} \rightarrow \mathsf{V\ NP}$$

Terminals

$$(a)\ \mathsf{A} \rightarrow \text{living}$$
$$(b)\ \mathsf{N} \rightarrow \text{people} \mid \text{food}$$
$$(c)\ \mathsf{V} \rightarrow \text{need}$$

Categorial Grammar

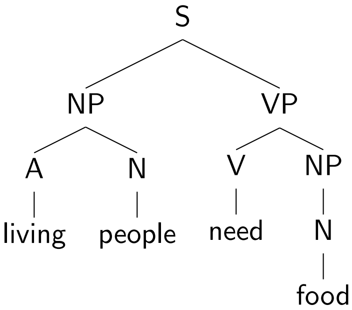
Application rules

$$(>)\quad X/Y \quad Y \Rightarrow X$$
$$(<)\quad Y \quad X\backslash Y \Rightarrow X$$

Terminals & categories

$$\text{living} := NP/N$$
$$\text{people} := N$$
$$\text{food} := NP$$
$$\text{need} := (S\backslash NP)/NP$$

Phrase-structure grammar derivation tree



Categorial grammar derivation

$$\cfrac{\cfrac{\cfrac{\textit{living}}{NP/N} \quad \cfrac{\textit{people}}{N}}{NP}\,> \qquad \cfrac{\cfrac{\textit{need}}{(S\backslash NP)/NP} \quad \cfrac{\textit{food}}{NP}}{S\backslash NP}\,>}{S}\,<$$

Categories & (Pure) Categorial Grammar

- Categories describe syntactical & grammatical properties of constituents
- They are referred to as ‘syntactic types’
- There are two kinds of types
  - Functional types – A, V
  - Atomic types – N, NP
- The choice is arbitrary, but “verbs are functions” is a well established concept

	Phrase-structure	**Categorial**
Rules	Explicit	Generic
Derivation	Terminals & nonterminals	Categories only
Expression–type association	Part of grammar	In corpus

Table : Phrase-structure grammars vs Categorial grammars

Another example: Transitive and intransitive verbs

Transitive verbs

(S\NP)/NP

$$\dfrac{\dfrac{\textit{John}}{NP} \qquad \dfrac{\dfrac{\textit{likes}}{(S\backslash NP)/NP} \qquad \dfrac{\textit{potatoes}}{NP}}{S\backslash NP}\,>}{S}\,<$$

Intransitive verbs

S\NP

$$\dfrac{\dfrac{\textit{John}}{NP} \qquad \dfrac{\textit{sleeps}}{S\backslash NP}}{S}\,<$$

From CG to CCG

> "Pure CG (Ajdukiewicz 1935, Bar-Hillel 1953) limits syntactic combination to rules of functional application of functions to arguments to the right or left. [...] This restriction limits expressivity to the level of context-free grammar, and CCG generalizes the context-free core by introducing further rules for combining categories."
>
> — Steedman and Baldridge, Combinatory Categorial Grammar

CCG: Slash types

- A slash has one of four feature values ($\star, \times, \diamond, \cdot$)
- Slash type imposes limits on possible combination
  - Formalized by application/combination rules
- Written as a subscript ($/_\star, /_\times, \backslash_\diamond, \backslash_\cdot$)

CCG: Slash types – $\star$

- The most restrictive
- Equivalent to the simple slash in CG
- Supertype of all other slash types

Rules

$$
\begin{array}{ll}
(>) & X/_{\star}Y \;\; Y \Rightarrow X \\
(<) & Y \;\; X\backslash_{\star}Y \Rightarrow X
\end{array}
$$

Interlude: Building the logical form

CCG allows you to associate functions with the rules. These functions can then be used to generate the logical representation.

Syntax

$$< \text{expression} >:= [< \text{category} >: \lambda < \text{paramter} > [...]\,.] < \text{body} >$$

Extended rules

$$\begin{array}{llll} (>) & X/_\star Y : f \quad Y : a & \Rightarrow X : fa \\ (<) & Y : a \quad X\backslash_\star Y : f & \Rightarrow X : fa \end{array}$$

Interlude: Building the logical form

$$\dfrac{\dfrac{Marcel}{NP : marcel'} \quad \dfrac{\dfrac{proved}{(S\backslash NP)/NP : \lambda x \lambda y.prove'xy} \quad \dfrac{completeness}{NP : completeness'}}{S\backslash NP : \lambda y.prove'completeness'y}>}{S : prove'completeness'marcel'}<$$

The expressions are left-associative
(prove'completeness'marcel' = (prove'completeness')marcel')

CCG: Slash types – $\times$

- Allows limited permutation
- Subtype of $\star$

Rules

$$(> \mathbf{B}_\times) \quad X/_\times Y : f \;\; Y\backslash_\times Z : g \Rightarrow X\backslash_\times Z : \lambda z.f(gz)$$
$$(< \mathbf{B}_\times) \quad Y/_\times Z : g \;\; X\backslash_\times Y : f \Rightarrow X/_\times Z : \lambda z.f(gz)$$

CCG: Slash types – $\diamond$

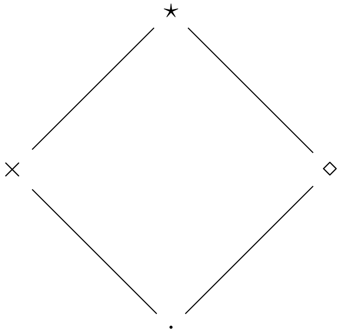
- Allows associativity (composition)
- Subtype of $\star$
- Can be iterated for a fixed n

Rules

$$(>\mathbf{B}) \quad X/_\diamond Y : f \;\; Y/_\diamond Z : g \Rightarrow X/_\diamond Z : \lambda z.f(gz)$$
$$(<\mathbf{B}) \quad Y\backslash_\diamond Z : g \;\; X\backslash_\diamond Y : f \Rightarrow X\backslash_\diamond Z : \lambda z.f(gz)$$

CCG: Slash types – $\cdot$

- Allows any of the former applications
- Subtype of both $\diamond$ and $\times$



CCG: Type raising

"Combinatory grammars also include type-raising rules, which turn arguments into functions over functions-over-such-arguments."

Rules

$$(> \mathbf{T}) \quad \mathrm{X} : a \Rightarrow \mathrm{T}/_i(\mathrm{T}\backslash_i \mathrm{X}) : \lambda f.fa$$

Where X is a primitive category

- Mimics case marking

CCG: Type raising

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathit{Marcel}}{NP}}{S/(S\backslash NP)}{>}\mathbf{T} \quad \dfrac{\mathit{proved}}{(S\backslash NP)/NP}}{S/NP}{>}\mathbf{B} \quad \dfrac{\dfrac{\mathit{and}}{(X\backslash_{\star}X)/_{\star}X} \quad \dfrac{\dfrac{\dfrac{\mathit{I}}{NP}}{S/(S\backslash NP)}{>}\mathbf{T} \quad \dfrac{\mathit{disproved}}{(S\backslash NP)/NP}}{S/NP}{>}\mathbf{B}}{(S/NP)\backslash_{\star}(S/NP)}{>}}{S/NP}{<} \quad \dfrac{\mathit{completeness}}{NP}}{S}{>}$$

CCG: Type raising

Question

Can we derive this sentence without type raising?

CCG: Type raising - cases and free word order

- There are approaches for languages with free word order [Karttunen, 1986]
- Most of them treat NPs as functors (because of flexion)
- Relying on prepositions as indicators of some cases doesn't always work
- ... especially when there are no prepositions

CCG: Type raising - cases and free word order

Finnish inessive:
-*ssa* indicates "in that place"

Nom		Ines	
kaupunki	city	kaupungissa	in city
kylä	village	kylässä	in village
huone	room	huoneessa	in room

Table : -*ssa*

All constituents of NP clusters take the same suffix: *suuri valkoinen talo* → *suuressa valkoisessa talossa* (in a big white house)

Other properties and relationships can be expressed in similar way: *suuressa valkoisessa talossamme* (in our big white house)

More in [Karttunen, 1986]

CCG: Combinatory principles

Adjacency, Consistency

The Principle of Inheritance
If the category that results from the application of a combinatory rule is a function category, then the slash type of a given argument in that category will be the same as the one(s) of the corresponding argument(s) in the input function(s).

$$X/Y \quad Y \Rightarrow Z$$
$$X/_{\diamond}Y \quad Y/_{\diamond}Z \quad \Rightarrow X/_{\times}Z$$

CCG: Expressive power

Question

Is CCG context-free?

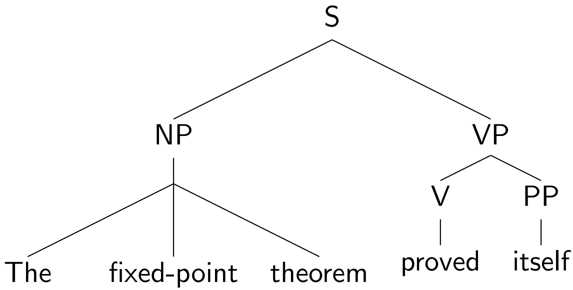
See [Vijay-Shanker and Weir, 1994]

OpenCCG demonstration

Bounded constructions

- Reflexivization
- Dative-shift
- Raising
- Object and Subject Control

Bounded constructions: Reflexivization



Bounded constructions: Reflexivization

$$\text{proved} := (S\backslash NP_{3sn})\backslash_{LEX}((S\backslash NP_{3sn})/NP) : \lambda p \lambda y.p(ana' y)y$$

$$\dfrac{\dfrac{The\ fixed-point\ theorem}{S/(S\backslash NP_{3sn}) : fptheorem'} \quad \dfrac{\dfrac{proved}{(S\backslash NP_{arg})/NP : \lambda x \lambda y.prove' xy} \quad \dfrac{itself}{(S\backslash NP_{3sn})\backslash((S\backslash NP_{3sn})/NP) : \lambda p \lambda y.p(ana' y)y}}{S\backslash NP_{3sn} : \lambda y.prove'(ana' y)y}<}{S : prove'(ana' fptheorem')fptheorem'}>$$

- It's a clitic!
- * *"Itself proved the fixed-point theorem"* is disallowed by the Principle of Inheritance
- Limitations of syntactical/lexical approach: *I got the book! – Can I see* ***it****?*
- Very similar approach to dative shifts

Bounded constructions: Raising

Modal verbs and verbs that behave like modals act on almost-complete sentences

(38) seems $:= (S\backslash NP)/(S_{TO}\backslash NP) : \lambda p\lambda y.seem'(py)$

The primitive *seem'* is a modal or intensional operator which the interpretation composes with the complement predicate, thus:

(39)

$$
\begin{array}{cccc}
\text{Marcel} & \text{seems} & \text{to} & \text{drink} \\
\hline
S/(S\backslash NP) & (S\backslash NP)/(S_{TO}\backslash NP) & (S_{TO}\backslash NP)/(S_{INF}\backslash NP) & S_{INF}\backslash NP \\
: \lambda p.p\ marcel' & : \lambda p\lambda y.seem'(py) & : \lambda p.p & : drink'
\end{array}
$$

$$\xrightarrow{\quad > \quad} S_{TO}\backslash NP : drink'$$

$$\xrightarrow{\quad > \quad} S\backslash NP : \lambda y.seem'(drink'y)$$

$$\xrightarrow{\quad > \quad} S : seem'(drink'marcel')$$

Bounded constructions: Object Control

Some verbs control the infinitival complement's subject through the object.

- I persuaded Marcel to take a bath.
- I persuaded Marcel to bathe himself.

persuaded $:= ((S\backslash NP)/(S_{TO}\backslash NP))/NP : \lambda x \lambda p \lambda y.persuade'(p(ana'x))xy$

persuade'(bathe'(ana'(ana'marcel'))(ana'marcel'))marcel'me'

Bounded constructions: Subject Control

Some verbs control the subject reference.

- John promised me _ to go away.
- John ordered me _ to go away.

Coordination

(69)

$$
\begin{array}{cccccc}
\text{give} & \text{Walt} & \text{the salt} & \text{and} & \text{Malcolm} & \text{the talcum} \\
\overline{DTV} & \overline{TV\backslash_{\diamond}DTV}^{<\mathbf{T}} & \overline{VP\backslash TV}^{<\mathbf{T}} & \overline{(X\backslash_{\star}X)/_{\star}X} & \overline{TV\backslash_{\diamond}DTV}^{<\mathbf{T}} & \overline{VP\backslash TV}^{<\mathbf{T}} \\
 & \multicolumn{2}{c}{\overline{VP\backslash_{\diamond}DTV}^{<\mathbf{B}}} & & \multicolumn{2}{c}{\overline{VP\backslash_{\diamond}DTV}^{<\mathbf{B}}} \\
 & & & \multicolumn{3}{c}{\overline{(VP\backslash_{\diamond}DTV)\backslash_{\star}(VP\backslash_{\diamond}DTV)}^{>}} \\
 & \multicolumn{5}{c}{\overline{VP\backslash_{\diamond}DTV}^{<}} \\
\multicolumn{6}{c}{\overline{VP}^{<}}
\end{array}
$$

Word order

(93) a. Kyooju-ga komonjo-o gakusee-ni kasita.
Professor-NOM manuscript-ACC student-DAT lent-PAST.CONCL
'The professor lent the manuscript to the student.'

b. Kyooju-ga komonjo-o gakusee-ni kasita.

$$
\begin{array}{cccc}
\text{Kyooju-ga} & \text{komonjo-o} & \text{gakusee-ni} & \text{kasita.} \\
\overline{S/VP}^{>\mathbf{T}} & \overline{VP/TV}^{>\mathbf{T}} & \overline{TV/DTV}^{>\mathbf{T}} & \overline{DTV} \\
\multicolumn{2}{c}{\overline{\qquad S/TV \qquad}^{>\mathbf{B}}} & & \\
\multicolumn{3}{c}{\overline{\qquad S/DTV) \qquad}^{>\mathbf{B}}} & \\
\multicolumn{4}{c}{\overline{\qquad\qquad S \qquad\qquad}^{>}}
\end{array}
$$

In this case there is another derivation for the argument cluster:

(94)

$$
\begin{array}{cccc}
\text{Kyooju-ga} & \text{komonjo-o} & \text{gakusee-ni} & \text{kasita.} \\
\overline{S/VP}^{>\mathbf{T}} & \overline{VP/TV}^{>\mathbf{T}} & \overline{TV/DTV}^{>\mathbf{T}} & \overline{DTV} \\
 & \multicolumn{2}{c}{\overline{\qquad VP/DTV \qquad}^{>\mathbf{B}}} & \\
\multicolumn{3}{c}{\overline{\qquad S/DTV \qquad}^{>\mathbf{B}}} & \\
\multicolumn{4}{c}{\overline{\qquad\qquad S \qquad\qquad}^{>}}
\end{array}
$$

Intonation

(107) Q: I know who proved soundness. But who proved COMPLETENESS?

A: (MARCEL) (proved COMPLETENESS).

H* L L+H* LH%

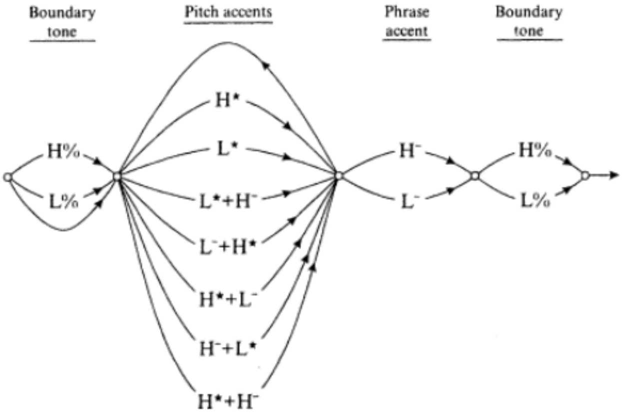
(108) Q: I know which result Marcel PREDICTED. But which result did Marcel PROVE?

A: (Marcel PROVED) (COMPLETENESS).

L+H*LH% H* LL%

(111) proved := $(S_\theta \backslash NP_\theta)/NP_\theta : \lambda x \lambda y. * prove' xy$

Intonation



Source: [Ladd, 2008]

Implementation

- CCG can be parsed in low polynomial time (quadratic)
  - However, most sentences are regular
  - This is an upper bound
  - Humans can do it in linear time (or better)
  - Statistical optimization
- OpenCCG
  - Parser & realizer
  - Java (and lots of XML, too)
  - Standalone or library
  - LGPL

Applications

- English
  - Dialogs
  - Intonation in generation
  - Generation in for in-car systems
- German
  - Parsing
- Italian, Greek, sign languages
- Most projects seem to be abandoned

More at
`www.utcompling.com/wiki/openccg/projects-using-openccg`

References & other resources

Steedman and Baldridge (2011)
Combinatory Categorial Grammar
http://homepages.inf.ed.ac.uk/steedman/papers/ccg/SteedmanBaldridgeNTSyntax.pdf

A. Brett
Lecture notes for Linguistics 484 (University of Victoria) http://web.uvic.ca/~ling48x/ling484/notes/index.html

Vijay-Shanker, K. and Weir, David J. (1994)
The Equivalence of Four Extensions of Context-Free Grammars. Mathematical Systems Theory 27(6): 511546.

D. R. Ladd (2008)
Intonational Phonology
Cambridge University Press

L. Karttunen (2008)
Radical Lexicalism
https://www.academia.edu/1863598/Radical_Lexicalism

B. Hoffman (1992)
A CCG Approach to Free Word Order Languages