Combinatory Categorial Grammar 2

Vera Demberg

Universität des Saarlandes

Grammatikformalismen SS 2013

4. Juni 2013

Table of Contents

Zur Erinnerung: Combinatory Categorial Grammar

Mark Steedman (Univ. of Edinburgh)

- Kann folgende Phänomene erklären:
  - Bindung / Kontrolle (Reflexivpronomen)
  - Lange Abhängigkeiten (Relativsätze)
  - Koordination
  - Kreuzende Dependenzen (Holländisches Beispiel; $\rightarrow$ mild kontextabhängig)
- direkte Syntax-Semantik Schnittstelle

Table of Contents

CCG Combinatory Rules

Forward and Backward Application (with Semantics):

Forward Application: $X/Y : f \quad Y : a \quad \Rightarrow_{>} \quad X : fa$

Backward Application: $Y : a \quad X\backslash Y : f \quad \Rightarrow_{<} \quad X : fa$

Example

Marcel	proved	completeness
$NP_{3sm} : marcel'$	$(S\backslash NP_{3s})/NP : \lambda x \lambda y.prove'xy$	$NP : completeness'$

$$\frac{(S\backslash NP_{3s})/NP : \lambda x \lambda y.prove'xy \quad NP : completeness'}{S\backslash NP_{3s} : \lambda y.prove'\,completeness'\,y}>$$

$$\frac{NP_{3sm} : marcel' \quad S\backslash NP_{3s} : \lambda y.prove'\,completeness'\,y}{S : prove'\,completeness'\,marcel'}<$$

CCG Combinatory Rules

Coordination:	X:g	CONJ:b	X:f	$\Rightarrow_{<\phi>}$	X: $\lambda\ldots b(f\ldots)(g\ldots)$
Forward Composition:	$X/Y : f$		$Y/Z : g$	$\Rightarrow_{>B}$	$X/Z : \lambda z.f(gz)$
Backward Composition:	$Y\backslash Z : g$		$X\backslash Y : f$	$\Rightarrow_{<B}$	$X\backslash Z : \lambda z.f(gz)$

Marcel — $NP : marcel'$

conjectured — $(S\backslash NP)/NP : conjecture'$

and — $(X\backslash_{\star}X)/_{\star}X : and'$

might — $(S\backslash NP)/VP : might'$

prove — $VP/NP : prove'$

completeness — $NP : completeness'$

might + prove $(>B)$: $(S\backslash NP)/NP : \lambda x\lambda y.might'(prove'x)y$

and + [might prove] $(>)$: $((S\backslash NP)/NP)\backslash_{\star}((S\backslash NP)/NP) : \lambda tv\lambda x\lambda y.and'(might'(prove'x)y)(tv\,xy)$

conjectured + [and might prove] $(<)$: $(S\backslash NP)/NP : \lambda x\lambda y.and'(might'(prove'x)y)(conjecture'xy)$

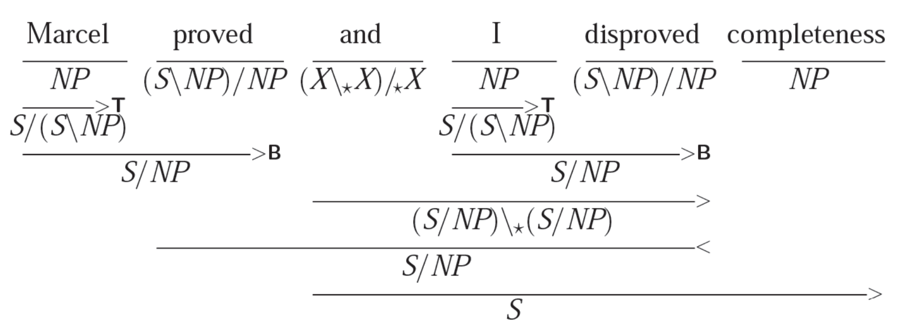
[conjectured and might prove] + completeness $(>)$: $S\backslash NP : \lambda y.and'(might'(prove'completeness')y)(conjecture'completeness'y)$

Marcel + [conjectured and might prove completeness] $(<)$: $S : and'(might'(prove'completeness')marcel')(conjecture'completeness'marcel')$

CCG Combinatory Rules

Forward Type-raising: $X : a \quad \Rightarrow_T \quad T/(T\backslash X) : \lambda f.fa$

Example



All rules for English

Forward Application:	$X/Y: f$	$Y: a$	$\Rightarrow_{>}$	$X: fa$
Backward Application:	$Y: a$	$X\backslash Y: f$	$\Rightarrow_{<}$	$X: fa$
Coordination:	X:g CONJ:b	X:f	$\Rightarrow_{<\phi>}$	X: $\lambda ... b(f...)(g...)$
Forward Composition:	$X/Y: f$	$Y/Z: g$	$\Rightarrow_{>B}$	$X/Z: \lambda z.f(gz)$
Backw Composition:	$Y\backslash Z: g$	$X\backslash Y: f$	$\Rightarrow_{<B}$	$X\backslash Z: \lambda z.f(gz)$
Forw Gen Composition:	X/Y: f	(Y/Z)/$\$_1$		
		$: ...\lambda z.gz...$	$\Rightarrow_{>B^n}$	(X/Z)/$\$_1$ $: ....f(gz...)$
Backw Crossed Comp:	Y/Z: g	X$\backslash$Y: f	$\Rightarrow_{<B_x}$	X/Z: $\lambda z.f(gz)$
Forward Type-raising:	$X: a$		$\Rightarrow_{T}$	$T/(T\backslash X): \lambda f.fa$

Reflexivpronomen

Mary	washed	herself
NP_{sf}	$(S\backslash NP_{sG})/NP$	$(S\backslash NP_{sf})\backslash((S\backslash NP_{sf})/NP)$
$: mary'$	$: \lambda x.\lambda y.wash' xy$	$: \lambda p.\lambda z.p(ana' z)z$

Objektkontrolle

Peter persuaded Marcel to bathe Mary.

Lexikon

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

to :=$(S_{TO}\backslash NP)/(S_{INF}\backslash NP) : \lambda p.p$

bathe := $(S_{INF}\backslash NP)/NP : \lambda v.\lambda w.bathe'\ vw$

Table of Contents

The Strict Competence Hypothesis

Strong Competence Hypothesis (Bresnan and Kaplan, 1982)

The Strong Competence Hypothesis of asserts that there exists a direct correspondence between the rules of a grammar and the operations performed by the human language processor.

Competence

Competence is the **'ideal' language system** that makes it possible for speakers to produce and understand an infinite number of sentences in their language, and to distinguish grammatical sentences from ungrammatical sentences.

Performance

Linguistic performance is governed by **principles of cognitive structure** such as memory limitations, distractions, shifts of attention and interest, and (random or characteristic) errors.

The Strict Competence Hypothesis

Strong Competence Hypothesis (Bresnan and Kaplan, 1982)

The Strong Competence Hypothesis of asserts that there exists a direct correspondence between the rules of a grammar and the operations performed by the human language processor.

Competence

Competence is the **'ideal' language system** that makes it possible for speakers to produce and understand an infinite number of sentences in their language, and to distinguish grammatical sentences from ungrammatical sentences.

Performance

Linguistic performance is governed by **principles of cognitive structure** such as memory limitations, distractions, shifts of attention and interest, and (random or characteristic) errors.

The Strict Competence Hypothesis

Strong Competence Hypothesis (Bresnan and Kaplan, 1982)

The Strong Competence Hypothesis of asserts that there exists a direct correspondence between the rules of a grammar and the operations performed by the human language processor.

+

Rule-to-Rule Assumption (Bach, 1976)

Each syntactic rule corresponds to a rule of semantic interpretation.
(⇒ entities combined by syntactic rules must be semantically interpretable)

=

Strict Competence Hypothesis (Steedman, 1992)

Structures manipulated by the processor are isomorphic to the constituents listed in the grammar.

The Strict Competence Hypothesis

Strong Competence Hypothesis (Bresnan and Kaplan, 1982)

The Strong Competence Hypothesis of asserts that there exists a direct correspondence between the rules of a grammar and the operations performed by the human language processor.

$+$

Rule-to-Rule Assumption (Bach, 1976)

Each syntactic rule corresponds to a rule of semantic interpretation.
($\Rightarrow$ entities combined by syntactic rules must be semantically interpretable)

$=$

Strict Competence Hypothesis (Steedman, 1992)

Structures manipulated by the processor are isomorphic to the constituents listed in the grammar.

The Strict Competence Hypothesis

Strong Competence Hypothesis (Bresnan and Kaplan, 1982)

The Strong Competence Hypothesis of asserts that there exists a direct correspondence between the rules of a grammar and the operations performed by the human language processor.

$+$

Rule-to-Rule Assumption (Bach, 1976)

Each syntactic rule corresponds to a rule of semantic interpretation.
($\Rightarrow$ entities combined by syntactic rules must be semantically interpretable)

$=$

Strict Competence Hypothesis (Steedman, 1992)

Structures manipulated by the processor are isomorphic to the constituents listed in the grammar.

Constituents in CCG

Flexible Konsituentenstruktur

Spurious ambiguity

There are 24 different ways of deriving:

Peter caught a big cat.

aber gut für inkrementelle Verarbeitung.

CCG und inkrementelle Verarbeitung

Beispiel ähnlich zu *The horse raced past the barn fell*.

a) The doctor sent for the patients arrived. (schwierig)

b) The flowers sent for the patients arrived. (leichter)

- Wenn b einfacher ist, bedeutet dies, dass wir schon bei "sent" bemerkt haben, dass *flowers* nicht der Agent von "send" sein kann.
  $\Rightarrow$ **Inkrementelle Interpretation bei "the flowers sent"**
- "the flowers sent" ist eine CCG Konstituente.
- CCG kann daher erklären, warum b für Menschen einfacher zu verarbeiten ist.

CCG und inkrementelle Verarbeitung

Beispiel ähnlich zu *The horse raced past the barn fell*.

a) The doctor sent for the patients arrived. (schwierig)

b) The flowers sent for the patients arrived. (leichter)

- Wenn b einfacher ist, bedeutet dies, dass wir schon bei “sent” bemerkt haben, dass *flowers* nicht der Agent von “send” sein kann.
  $\Rightarrow$ **Inkrementelle Interpretation bei “the flowers sent”**
- “the flowers sent” ist eine CCG Konstituente.
- CCG kann daher erklären, warum b für Menschen einfacher zu verarbeiten ist.

So how does that work?

```
  the        flowers              sent          for     the        patient
--------   -----------        ------------    ------  --------   -----------
 NP/N:          N:             (S\NP)/PP:     PP/NP:    NP/N:         N:
^P.def'P   ^x.flowers'x      ^y^x.summon'yx    ^x.x   ^P.def'P   ^x.patient'x
 -------------------->0
 NP:def'(^x.flowers'x)
 ---------------------->T
      S/(S\NP):
 ^P.P(def'(^x.flowers'x))
    ---------------------------------->1
    S/PP:^y.summon'y(def'(^x.flowers'x))
              ---------------------------------->1
              S/NP:^y.summon'y(def'(^x.flowers'x))
                     ----------------------------------------->1
                     S/N:^P.summon'(def'P)(def'(^x.flowers'x))
                            ------------------------------------------------>0
                            S:summon'(def'(^x.patient'x))(def'(^y.flowers'y))
```

Figure: Incremental CCG derivation (Figure taken from McConville's PhD thesis.)

So how does that work?

```
  the        flowers              sent               for      the        patient
--------   ------------   ---------------------    ------   --------   ------------
 NP/N:          N:              (N\N)/PP:           PP/NP:    NP/N:          N:
^P.def'P   ^x.flowers'x   ^z^P^y.Py&send'zy sb'    ^x.x    ^P.def'P   ^x.patient'x
         ---------------->T
            N/(N\N):
         ^Q.^x.Q(flowers')x
  --------------------->1
       NP/(N\N):
  ^Q.def'(^x.Qflowers'x)
     ---------------------------------------->1
     NP/PP:^z.def'(^y.flowers'y&send'zy sb')
               ---------------------------------------->1
               NP/NP:^z.def'(^y.flowers'y&send'zy sb')
                    ---------------------------------------->1
                    NP/N:^P.def'(^y.flowers'y&send'(def'P)y sb')
                              ------------------------------------------------------>0
                              NP:def'(^y.flowers'y&send'(def'(^x.patient'x))y sb')
```

Figure: Incremental CCG derivation (Figure taken from McConville's PhD thesis.)

Table of Contents

All rules for English

Forward Application:	$X/Y: f$	$Y: a$	$\Rightarrow_{>}$	$X: fa$
Backward Application:	$Y: a$	$X\backslash Y: f$	$\Rightarrow_{<}$	$X: fa$
Coordination:	X:g CONJ:b	X:f	$\Rightarrow_{<\phi>}$	X: $\lambda ... b(f...)(g...)$
Forward Composition:	$X/Y: f$	$Y/Z: g$	$\Rightarrow_{>B}$	$X/Z: \lambda z.f(gz)$
Backw Composition:	$Y\backslash Z: g$	$X\backslash Y: f$	$\Rightarrow_{<B}$	$X\backslash Z: \lambda z.f(gz)$
Forw Gen Composition:	X/Y: f	(Y/Z)/$\$_1$		
		$: ...\lambda z.gz...$	$\Rightarrow_{>B^n}$	(X/Z)/$\$_1$ $: ....f(gz...)$
Backw Crossed Comp:	Y/Z: g	X\Y: f	$\Rightarrow_{<B_x}$	X/Z: $\lambda z.f(gz)$
Forward Type-raising:	$X: a$		$\Rightarrow_{T}$	$T/(T\backslash X): \lambda f.fa$

Crossing composition rules

The crossing functional composition rules

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

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

I	introduced	to Marcel	my very heavy friends
$S/(S\backslash NP)$: $\lambda p.p\ me'$	$((S\backslash NP)/PP_{TO})/NP$: $\lambda x\lambda y\lambda z.introduce'yxz$	$S\backslash(S/PP_{TO})$: $\lambda q.q\ marcel'$	$S\backslash(S/NP)$: $\lambda r.r\ friends'$

$$\frac{S/(S\backslash NP) : \lambda p.p\ me' \quad ((S\backslash NP)/PP_{TO})/NP : \lambda x\lambda y\lambda z.introduce'yxz}{(S/PP_{TO})/NP : \lambda x\lambda y.introduce'yx\ me'}\ {>}\mathbf{B}^2$$

$$\frac{(S/PP_{TO})/NP : \lambda x\lambda y.introduce'yx\ me' \quad S\backslash(S/PP_{TO}) : \lambda q.q\ marcel'}{S/NP : \lambda x.introduce'marcel'x\ me'}\ {<}\mathbf{B}_{\times}$$

$$\frac{S/NP : \lambda x.introduce'marcel'x\ me' \quad S\backslash(S/NP) : \lambda r.r\ friends'}{S : introduce'marcel'friends'me'}\ {<}$$

Swiss German / Dutch

The crossing functional composition rules

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

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

das	mer	em Hans	es huus	hälfed	aastriiche
that	we−NOM	Hans−DAT	the house−ACC	helped	paint
	$NP^{\uparrow}_{nom}$	$NP^{\uparrow}_{dat}$	$NP^{\uparrow}_{acc}$	$((S_{+SUB}\backslash NP_{nom})\backslash NP_{dat})/VP$	$VP\backslash NP_{acc}$

hälfed + aastriiche $\;>\mathbf{B}_{\times}$: $((S_{+SUB}\backslash NP_{nom})\backslash NP_{dat})\backslash NP_{acc}$

es huus + … $\;>$: $(S_{+SUB}\backslash NP_{nom})\backslash NP_{dat}$

em Hans + … $\;>$: $S_{+SUB}\backslash NP_{nom}$

mer + … $\;>$: S_{+SUB}

"that we helped Hans paint the house"