Syntactic Formalisms for Parsing Natural Languages

Aleš Horák, Miloš Jakubíček, Vojtěch Kovář
(based on slides by Juyeon Kang)

ia161@nlp.fi.muni.cz

Autumn 2013

Parsing with CCG

Outline

- Categorial Grammar is
  - : a lexicalized theory of grammar along with other theories of grammar such as HPSG, TAG, LFG, ...
  - : linguistically and computationally attractive
    $\longrightarrow$ language invariant combination rules, high efficient parsing

Main idea in CG and *application operation*

- All natural language consists of operators and of operands.
  - Operator (functor) and operand (argument)
  - Application: (operator(operand))
  - Categorial type: typed operator and operand

1. A-B categorial system

The product of the directional adaptation by Bar-Hillel (1953) of Ajdukiewicz's calculus of syntactic connection (Ajdukiewicz, 1935)

Definition 1 (AB categories).

Given *A*, a finite set of *atomic categories*, the **set of categories C** is the smallest set such that:

- $A \subseteq C$
- $(X \backslash Y), (X/Y) \in C$ *if* $X, Y \in C$

1. A-B categorial system

- **Categories** (type): primitive categories and derivative categories
  - Primitive: S for sentence, N for nominal phrase
  - Derivative: $S/N, N/N, (S\backslash N)/N, NN/N, S/S \ldots$
- Forward($>$) and backward ($<$) **functional application**

$$a.\ X/Y\ Y \Rightarrow X \qquad (>)$$
$$b.\ Y\ X\backslash Y \Rightarrow X \qquad (<)$$

1. A-B categorial system

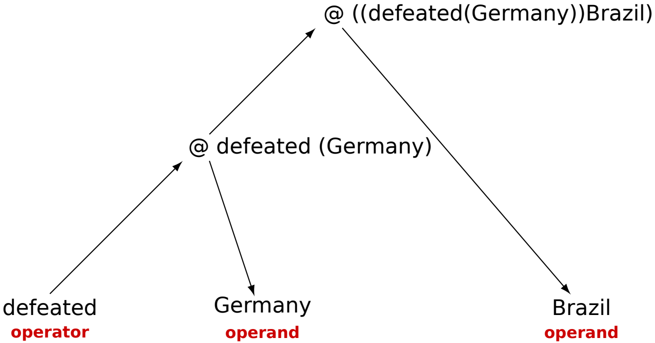
- **Calculus on types** in CG are analogue to **algebraic operations**

$$x/y \; y \rightarrow x \quad \approx \quad 3/5 * 5 = 3$$

$$\begin{array}{ccc}
\textit{Brazil} & \textit{defeated} & \textit{Germany} \\
\hline
n & (s\backslash n)/n & n \\
\cline{2-3}
 & \multicolumn{2}{c}{s\backslash n \quad >} \\
\hline
\multicolumn{3}{c}{s \quad <}
\end{array}$$

1. A-B categorial system

Applicative tree of *Brazil defeated Germany*



Limitation of AB system

1 Relative construction

a. team_i that t_i defeated Germany
b. team_i that Brazil defeated t_i

a'. that $(n\backslash n)/(s\backslash n)$ — team $[\text{that}]_{(n\backslash n)/(s\backslash n)}$ $[\text{defeated Germany}]_{s\backslash n}$

b'. that $(n\backslash n)/(s/n)$ — team $[\text{that}]_{(n\backslash n)/(s/n)}$ $[\text{Brazil defeated}]_{s/n}$

team	that	Brazil	defeated
	$(n\backslash n)/(s/n)$	n	$(s\backslash n)/n$

$\longleftarrow$ (?)

3 Many others complex phenomena

- Coordination, object extraction, phrasal verbs, ...

4 AB's generative power is too weak – *context-free*

2. Lambek calculus (Lambek, 1958, 1961)

the calculus of **syntactic types**
still *context-free*

The axioms of Lambek calculus are the following:

1. $x \to x$
2. $(xy)z \to x(yz) \to (xy)z$ (the axioms 1, 2 with inference rules, 3, 4, 5)
3. If $xy \to z$ then $x \to z/y$, if $xy \to z$ then $y \to x\backslash z$;
4. If $x \to z/y$ then $xy \to z$, if $y \to x\backslash z$ then $xy \to z$;
5. If $x \to y$ and $y \to z$ then $x \to z$.

2. Lambek calculus (Lambek, 1958, 1961)

The rules obtained from the previous axioms are the following:

1. Hypothesis: if x and y are types, then x/y and $y\backslash x$ are types.
2. Application rules : $(x/y)y \rightarrow x, y(y\backslash x) \rightarrow x$
   ex: *Poor John works.*
3. *Associativity rule :* $(x\backslash y)/z \leftrightarrow x\backslash(y/z)$
   ex: John likes Jane.
4. Composition rules : $(x/y)(y/z) \rightarrow x/z, (x\backslash y)(y\backslash z) \rightarrow x\backslash z$
   ex: *He likes him.*
   $s/(n\backslash s)n\backslash s/n$
5. *Type-raising rules :* $x \rightarrow y/(x/y), x \rightarrow (y/x)\backslash y$

3. Combinatory Categorial Grammar

- Developed originally by M. Steedman (1988, 1990, 2000, ...)
- Combinatory Categorial Grammar (CCG) is a grammar formalism equivalent to Tree Adjoining Grammar, i.e.
  - it is lexicalized
  - it is parsable in polynomial time (See Vijay-Shanker and Weir, 1990)
  - it can capture cross-serial dependencies
- Just like TAG, CCG is used for grammar writing
- CCG is especially suitable for statistical parsing

3. Combinatory Categorial Grammar

- several of the **combinators which Curry and Feys** (1958) use to define **the λ-calculus** and applicative systems in general are of considerable syntactic interest (Steedman, 1988)
- The relationships of these combinators to terms of the λ-calculus are defined by the following equivalences (Steedman, 2000b):

$$a.\mathbf{B}fg \equiv \lambda x.f(gx) \text{ ... composition}$$
$$b.\mathbf{T}x \equiv \lambda f.fx \text{ ... type-raising}$$
$$c.\mathbf{S}fg \equiv \lambda x.fx(gx) \text{ ... substitution}$$

CCG categories

- Atomic categories: S, N, NP, PP, TV...
- Complex categories are built recursively from atomic categories and slashes
- Example complex categories for verbs:
  - intransitive verb: $S\backslash NP$ walked
  - transitive verb: $(S\backslash NP)/NP$ respected
  - ditransitive verb: $((S\backslash NP)/NP)/NP$ gave

Lexical categories in CCG

- An elementary syntactic structure – a lexical category – is assigned to each word in a sentence, eg:

  walked: $S\backslash NP$ ‘give me an NP to my left and I return a sentence’

- Think of the lexical category for a verb as a function: NP is the argument, S the result, and the slash indicates the direction of the argument

The typed lexicon item

- 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 $(S\backslash NP)/NP \lambda x \lambda y.\mathrm{loves}' xy$
- Combinatory rules determine what happens with the category and the semantics on combination

The typed lexicon item

- Attribution of types to lexical items: examples

Predicate

ex: is as an identificator of nominal

as an *operator of predication* from a nominal $\longrightarrow (S\backslash NP)/NP$

from an adjective $\longrightarrow (S\backslash NP)/(N/N)$

from an adverb $\longrightarrow (S\backslash NP)/(S\backslash NP)\backslash(S\backslash NP)$

from a preposition $\longrightarrow (S\backslash NP)/((S\backslash NP)\backslash(S\backslash NP)/NP)$

ex: verbs

$\longrightarrow$ unary $(S\backslash NP)$

$\longrightarrow$ binary $(S\backslash NP)/NP$

$\longrightarrow$ ternary $(S\backslash NP)/NP/NP$

The typed lexicon item

Adverbs

Adverb of verb

$(S\backslash NP)/(S\backslash NP)$

$(S\backslash NP)/NP/(S\backslash NP)/NP$

Adverb of adverb

$(S\backslash NP)/(S\backslash NP)/(S\backslash NP)/(S\backslash NP)$

$(S\backslash NP)/NP/(S\backslash NP)/NP/(S\backslash NP)/NP/(S\backslash NP)/NP$

Adverb of adjective

$(N/N)/(N/N)$

$(N\backslash N)/(N\backslash N)$

Adverb of proposition

S/S

Adverb: operator of determination of type (X/X)

The typed lexicon item

Preposition

Prep. 1:
constructor of adverbial phrase

$(S\backslash NP)\backslash(S\backslash NP)/NP$
$(S/S)/NP$
$(S/S)/N$

Prep. 2:
constructor of adjectival phrase

$(N\backslash N)/NP$
$(N\backslash N)/N$

Preposition: constructor of determination of type (X/X)

Dictionary of typed words

Syntactic categories	Syntactic types	Lexical entries
Nom.	N	*Olivia, apple...*
Completed nom.	NP	*an apple, the school*
Pron.	NP	*She, he...*
Adj.	$(N/N), (N\backslash N)$	***pretty** woman,...*
Adv.	$(N/N)/(N/N)$,	***very** delicious,...*
	$(S\backslash NP)\backslash(S\backslash NP)$...	
Vb	$(S\backslash NP), (S\backslash NP)/NP$...	*run, give...*
Prep.	$(S\backslash NP)\backslash(S\backslash NP)/NP$	*run in the park,*
	$(NP\backslash NP)/NP$...	*book of John, ...*
Relative	$(S\backslash NP)/S$...	*I believe that...*

Combinatorial categorial rules

- Functional application $(>, <)$
- Functional composition $(> \mathbf{B}, < \mathbf{B})$
- Type-raising $(< \mathbf{T}, > \mathbf{T})$
- Distribution $(< \mathbf{S}, > \mathbf{S})$
- Coordination $(< \Phi, > \Phi)$

Functional application (FA)

$X/Y : f \quad Y : a \Rightarrow X : fa$ (forward functional application, $>$)

$Y : a \quad X\backslash Y : f \Rightarrow X : fa$ (backward functional application, $<$)

- Combine a function with its argument:

$$\dfrac{NP \quad S\backslash NP}{S}<$$

Mary sleeps → (sleeps (Mary))

$$\dfrac{NP \quad \dfrac{(S\backslash NP)/NP \quad NP}{S\backslash NP}>}{S}<$$

$S\backslash NP$ → (likes (Mary))

John likes Mary → ((likes (Mary))John)

- Direction of the slash indicates position of the argument with respect to the function

Derivation in CCG

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

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

Function composition (FC)

Generalized forward composition $(> \boldsymbol{Bn})$

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

- Functional composition composes two complex categories (two functions):

$$(S\backslash NP)/PP \quad (PP/NP) \Rightarrow_B (S\backslash NP)/NP$$
$$S/(S\backslash NP) \quad (S\backslash NP)/NP \Rightarrow_B S/NP$$

$$\begin{array}{ccc}
\text{birds} & \text{like} & \text{bugs} \\
\hline
NP & (S\backslash NP)/NP & NP \\
\hline
S/(S\backslash NP) \quad {>T} & & \\
\hline
\multicolumn{2}{c}{S/NP} \quad {>B} & \\
\hline
\multicolumn{3}{c}{S} \quad {>}
\end{array}$$

Function composition (FC)

Generalized backward composition $(< \boldsymbol{Bn})$

$$Y\backslash Z : f \qquad X\backslash Y : g \Rightarrow_B X\backslash Z : x.f(gx) \qquad (< B)$$

The referee gave	Unsal	a card	and	Rivaldo	the ball
$(s/np)/np$	np	np	$(X\backslash X)/X$	np	np
	$<$T	$<$T		$<$T	$<$T
	$(s/np)\backslash((s/np)/np)$	$s\backslash(s/np)$		$(s/np)\backslash((s/np)/np)$	$s\backslash(s/np)$

Unsal + a card: $<$B

$s\backslash((s/np)/np$

Rivaldo + the ball: $<$B

$s\backslash((s/np)/np$

Unsal a card and Rivaldo the ball: $< \Phi >$

$s\backslash((s/np)/np$

The referee gave … the ball: $<$

s

Type-raising (T)

Forward type-raising $(> \boldsymbol{T})$

$$X : a \Rightarrow T/(T\backslash X) : \lambda f.fa \qquad (> T)$$

- Type-raising turns an <u>argument</u> into a <u>function</u> (e.g. for case assignment)

$$NP \Rightarrow S/(S\backslash NP) \text{ (nominative)}$$

$$\dfrac{\dfrac{\textit{birds}}{NP} \qquad \dfrac{\textit{fly}}{S\backslash NP}}{S}<$$

$$\dfrac{\dfrac{\dfrac{\textit{birds}}{NP}}{S/(S\backslash NP)}>T \qquad \dfrac{\textit{fly}}{S\backslash NP}}{S}>$$

- This must be used *e.g. in the case of WH-questions*

Example of functional composition $(> \boldsymbol{B})$ and type-raising (T)

team	*that*	*I*	*thought*	*that*	*Brazil*	*defeated*
n	$(n\backslash n)/(s/np)$	np	$(s\backslash np)/s$	s/s	np	$(s\backslash np)/np$

$$\frac{np}{s/(s\backslash np)}\ {>}\mathrm{T}$$

$$\frac{s/(s\backslash np) \quad (s\backslash np)/s}{s/s}\ {>}\mathrm{B}$$

$$\frac{np}{s/(s\backslash np)}\ {>}\mathrm{T}$$

$$\frac{s/(s\backslash np) \quad (s\backslash np)/np}{s/np}\ {>}\mathrm{B}$$

$$\frac{s/s \quad s/np}{s/np}\ {>}\mathrm{B}$$

$$\frac{s/s \quad s/np}{s/np}\ {>}\mathrm{B}$$

$$\frac{(n\backslash n)/(s/np) \quad s/np}{n\backslash n}\ {>}$$

$$\frac{n \quad n\backslash n}{n}\ {<}$$

Example of functional composition $(> B)$ and type-raising (T)

Backward type-raising $(< T)$

$$X : a \Rightarrow T\backslash(T/X) : \lambda f.fa \qquad (< T)$$

- Type-raising turns an argument into a function (e.g. for case assignment)

$$NP \Rightarrow (S\backslash NP)\backslash((S\backslash NP)/NP) \text{ (accusative)}$$

The referee gave	*Unsal*	*a card*	*and*	*Rivaldo*	*the ball*
$(s/np)/np$	np	np	$(X\backslash X)/X$	np	np
	$<T$: $(s/np)\backslash((s/np)/np)$	$<T$: $s\backslash(s/np)$		$<T$: $(s/np)\backslash((s/np)/np)$	$<T$: $s\backslash(s/np)$

$$\dfrac{(s/np)\backslash((s/np)/np) \quad s\backslash(s/np)}{s\backslash((s/np)/np)} < B \qquad \dfrac{(s/np)\backslash((s/np)/np) \quad s\backslash(s/np)}{s\backslash((s/np)/np)} < B$$

$$\dfrac{s\backslash((s/np)/np) \quad (X\backslash X)/X \quad s\backslash((s/np)/np)}{s\backslash((s/np)/np)} < \Phi >$$

$$\dfrac{(s/np)/np \quad s\backslash((s/np)/np)}{s} <$$

Coordination (&)

$$X\ CONJ\ X \Rightarrow_{\Phi} X \qquad (Coordination(\Phi))$$

$$\begin{array}{cccccc}
\textit{give} & \textit{a dog} & \textit{a bone} & \textit{and} & \textit{a policeman} & \textit{a flower} \\
\overline{(VP/NP)/NP} & \overline{(VP/NP)\backslash((VP/NP)/NP)}{<}T & \overline{VP\backslash(VP/NP)}{<}T & \overline{\text{conj}} & \overline{(VP/NP)\backslash((VP/NP)/NP)}{<}T & \overline{VP\backslash(VP/NP)}{<}T
\end{array}$$

$$\textit{a dog} + \textit{a bone}: \overline{VP\backslash((VP/NP)/NP)}{<}B \qquad \textit{a policeman} + \textit{a flower}: \overline{VP\backslash((VP/NP)/NP)}{<}B$$

$$\overline{VP\backslash((VP/NP)/NP)}{<\&>}$$

$$\overline{VP}{<}$$

Substitution (S)

Forward substitution ($> \boldsymbol{S}$)

$$(X/Y)/Z \; Y/Z \Rightarrow_S X/Z$$

- Application to parasitic gap such as the following:

a. team that I persuaded every detractor of to support

team	that	I	persuaded	every detractor of	to support
	$(n\backslash n)/(s/np)$	np	$((s\backslash np)/(s\backslash np))/np$	np/np	$(s\backslash np)/np$

$np \Rightarrow_{>T} s/(s\backslash np)$

$((s\backslash np)/(s\backslash np))/np \;\; np/np \Rightarrow_{>B} ((s\backslash np)/(s\backslash np))/np$

$((s\backslash np)/(s\backslash np))/np \;\; (s\backslash np)/np \Rightarrow_{>S} (s\backslash np)/np$

$s/(s\backslash np) \;\; (s\backslash np)/np \Rightarrow_{>B} s/np$

$(n\backslash n)/(s/np) \;\; s/np \Rightarrow_{>} n\backslash n$

Substitution (S)

Backward crossed substitution ($< S\times$)

$$Y/Z \ (X\backslash Y)/Z \Rightarrow_S X/Z$$

- Application to parasitic gap such as the following:

a. *John watched without enjoying the game between Germany and Paraguay.*

b. game that John watched without enjoying

game that John [watched]$_{(s\backslash np)/np}$ [without enjoying]$_{((s\backslash np)\backslash(s\backslash np))/np}$

$$
\begin{array}{llllll}
\textit{game} & \textit{that} & \textit{John} & \textit{watched} & \textit{without enjoying} & \\
\hline
 & (n\backslash n)/(s/np) & np & (s\backslash np)/np & ((s\backslash np)\backslash(s\backslash np))/np & \\
 & & \overline{s/(s\backslash np)}^{>T} & \multicolumn{2}{c}{\overline{(s\backslash np)/np}^{<S_\times}} & \\
 & & \multicolumn{3}{c}{\overline{s/(s\backslash np)}^{>B}} & \\
 & \multicolumn{4}{c}{\overline{n\backslash n}^{>}} &
\end{array}
$$

Limit on possible rules

- The Principle of Adjacency:

Combinatory rules may only apply to entities which are linguistically realised and adjacent.

- The Principle of Directional Consistency:

All syntactic combinatory rules must be consistent with the directionality of the principal function. ex: $X\backslash Y\ Y \not=> X$

- The Principle of Directional Inheritance:

If the category that results from the application of a combinatory rule is a function category, then the slash defining directionality for a given argument in that category will be the same as the one defining directionality for the corresponding arguments in the input functions. ex: $X/Y\ Y/Z \not=> X\backslash Z$.

Semantic in CCG

- CCG offers a syntax-semantics interface.
- The lexical categories are augmented with an explicit identification of their semantic interpretation and the rules of functional application are accordingly expanded with an explicit semantics.
- Every syntactic category and rule has a semantic counterpart.
- The lexicon is used to pair words with syntactic categories and semantic interpretations:

$$love\ (S\backslash NP)/NP \Rightarrow \lambda x \lambda y.loves'xy$$

Semantic in CCG

- The semantic interpretation of all combinatory rules is fully determined by the **Principle of Type Transparency**:
  - **Categories**: All syntactic categories reflect the semantic type of the associated logical form.
  - **Rules**: All syntactic combinatory rules are type-transparent versions of one of a small number of semantic operations over functions including application, composition, and type-raising.

Semantic in CCG

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

- the semantic type of the reduction is the same as its syntactic type, here functional application.

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

Semantic in CCG

CCG with semantics : *Mary will copy and file without reading these articles*

Mary will	*copy*	*and*	*file*	*without*	*reading*	*these articles*
S/VP :p.Mary' λp.will'	VP/NP :copy'	CONJ :and'	VP/NP :file'	(VP\VP)/VPing λp.λq.without'pq	VPing/NP :read'	NP :articles'

without + reading, $>B$:
(VP\VP)/VPing
:λx.λq.without'(read' x)q

file + (without reading), $<S$:
VP/NP
:λx.without'(read'x)(file'x)

copy + and + (file without reading), $<\Phi>$:
VP/NP
:λx.and'(without'(read'x)(file'x))(copy'x)

(copy and file without reading) + these articles, $<$:
VP
:and'(without')(read'articles')(file'articles'))(copy'articles')

Mary will + (copy and file without reading these articles), $>$:
S
:will'(and'(without')(read'articles')(file'articles'))(copy'articles'))mary'

Parsing a sentence in CCG

Step 1: tokenization

Step 2: tagging the concatenated lexicon

Step 3:

- calculate on types attributed to the concatenated lexicons by applying the adequate combinatorial rules
- eliminate the applied combinators (we will see how to do on next week)

Step 4: finding the parsing results presented in the form of an operator/operand structure (predicate -argument structure)

Parsing a sentence in CCG

Example: *I requested and would prefer musicals*

STEP 1 : tokenization/lemmatization $\rightarrow$ *ex) POS Tagger, tokenizer, lemmatizer*

a. *I-requested-and-would-prefer-musicals*
b. *I-request-ed-and-would-prefer-musical-s*

STEP 2 : tagging the concatenated expressions $\rightarrow$ *ex) Supertagger, Inventory of typed words*

I	NP
Requested	$(S\backslash NP)/NP$
And	$CONJ$
Would	$(S\backslash NP)/VP$
Prefer	VP/NP
musicals	NP

Parsing a sentence in CCG

STEP 3 : categorial calculus

a. apply the type-raising rules → *Subject Type-raising* $(> T)$
$NP : a \Rightarrow T/(T\backslash NP) : Ta$

b. apply the functional composition rules → *Forward Composition:* $(> B)$
$X/Y : f \quad Y/Z : g \Rightarrow X/Z : Bfg$

c. apply the coordination rules → *Coordination:* $(< \& >)$
$X\ conj\ X \Rightarrow X$

	I-	requested-	and-	would-	prefer-	musicals	
1/	NP	$(S\backslash NP)/NP$	CONJ	$(S\backslash NP)/VP$	VP/NP	NP	
2/	$S/(S\backslash NP)$	$(S\backslash NP)/NP$	CONJ	$(S\backslash NP)/VP$	VP/NP	NP	$(>T)$
3/	$S/(S\backslash NP)$	$(S\backslash NP)/NP$	CONJ	$(S\backslash NP)/NP$		NP	$(>B)$
4/	$S/(S\backslash NP)$	$(S\backslash NP)/NP$				NP	$(> \Phi)$
5/	$S/(S\backslash NP)$	$(S\backslash NP)/NP$				NP	$(>B)$
6/	S/NP					NP	$(>)$
7/	**S**						

Parsing a sentence in CCG

STEP 4 : semantic representation (predicate-argument structure)

I requested and would prefer musicals

1/ :i' :request' :and' : will' :prefer' : musicals'

2/ :λf.f I'

3/ : λx.λy.will'(prefer'x)y

4/ : λxλy.and'(will'(prefer'x)y)(request'xy)

5/ : λxλy.and'(will'(prefer'x)y)(request'xy)

6/ :λy.and'(would'(prefer' musicals')y)(request' musicals' y)

7/S: and'(will'(prefer' musicals') i')(request' musicals' i')

Variation of CCG : Multi-modal CCG (Baldridge, 2002)

- Modalized CCG system
- Combination of Categorial Type Logic (CTL, Morrill, 1994; Moortgat, 1997) into the CCG (Steedman, 2000)
- Rules restrictions by introducing the modalities: *, x, $\bullet$, $\Diamond$
- Modalized functional composition rules

$$\begin{array}{llcl} (>B) & X/_{\Diamond}Y \quad Y/_{\Diamond}Z & \Rightarrow & X/_{\Diamond}Z \\ (<B) & X\backslash_{\Diamond}Y \quad Y\backslash_{\Diamond}Z & \Rightarrow & X\backslash_{\Diamond}Z \end{array}$$

- Invite you to read the paper “*Multi-Modal CCG*” of (Baldridge and M.Kruijff, 2003)

The positions of several formalisms on the Chomsky hierarchy