Introduction to Cognitive Science: Notes

VI: Universal Grammar is Transparent to Planning

- Readings for this section: *A very short introduction to CCG (see weekly syllabus page).

Categorial Grammar

- Categorial Grammar replaces PS rules by lexical categories and general combinatory rules (**Lexicalization**):

(1) $S \rightarrow NP \quad VP$

$VP \rightarrow TV \quad NP$

$TV \rightarrow \{proved, finds, \ldots\}$

- Categories:

(2) proved := $(S\backslash NP)/NP$

(3) think := $(S\backslash NP)/_{\diamond}S$

Categorial Grammar

- Categorial Grammar replaces PS rules by lexical categories and general combinatory rules (**Lexicalization**):

(1) $S \rightarrow NP \quad VP$

$VP \rightarrow TV \quad NP$

$TV \rightarrow \{proved, finds, \ldots\}$

- Categories:

(2) proved := $(S\backslash NP)/NP$: $prove'$

(3) think := $(S\backslash NP)/_{\diamond}S$: $think'$

Applicative Derivation

- Functional Application

$$\frac{X/_{\star}Y \qquad Y}{X}\ > \qquad \frac{Y \qquad X\backslash_{\star}Y}{X}\ <$$

- (4)

$$\dfrac{\dfrac{\text{Marcel}}{NP} \qquad \dfrac{\dfrac{\text{proved}}{(S\backslash NP)/NP} \qquad \dfrac{\text{completeness}}{NP}}{S\backslash NP}\,>}{S}\,<$$

(5)

$$\dfrac{\dfrac{\text{I}}{NP} \qquad \dfrac{\dfrac{\text{think}}{(S\backslash NP)/_{\diamond}S} \qquad \dfrac{\dfrac{\text{Marcel}}{NP} \qquad \dfrac{\dfrac{\text{proved}}{(S\backslash NP)/NP} \qquad \dfrac{\text{completeness}}{NP}}{S\backslash NP}\,>}{S}\,<}{S\backslash NP}\,>}{S}\,<$$

Applicative Derivation

- Functional Application

$$\frac{X/_{\star}Y : f \quad Y : g}{X : f(g)}\ > \qquad \frac{Y : g \quad X\backslash_{\star}Y : f}{X : f(g)}\ <$$

- (4)

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

(5)

$$\dfrac{\dfrac{\text{I}}{NP : i'} \quad \dfrac{\dfrac{\text{think}}{(S\backslash NP)/_{\diamond}S : think'} \quad \dfrac{\dfrac{\dfrac{\text{Marcel}}{NP : marcel'} \quad \dfrac{\dfrac{\text{proved}}{(S\backslash NP)/NP : prove'} \quad \dfrac{\text{completeness}}{NP : completeness'}}{S\backslash NP : \lambda y.prove'\,completeness'\,y}>}{S : prove'\,completeness'\,marcel'}<}{S\backslash NP : think'\,(prove'\,completeness'\,marcel')}>}{S : think'\,(prove'\,completeness'\,marcel')\,i'}<$$

Combinatory Categorial Grammar (CCG)

- Combinatory Rules:

$$\frac{X/_{\star}Y \qquad Y}{X} > \qquad \frac{Y \qquad X\backslash_{\star}Y}{X} <$$

$$\frac{X/_{\diamond}Y \qquad Y/_{\diamond}Z}{X/_{\diamond}Z} >\mathbf{B} \qquad \frac{Y\backslash_{\diamond}Z \qquad X\backslash_{\diamond}Y}{X\backslash_{\diamond}Z} <\mathbf{B}$$

$$\frac{X/_{\times}Y \qquad Y\backslash_{\times}Z}{X\backslash_{\times}Z} >\mathbf{B}_{\times} \qquad \frac{Y/_{\times}Z \qquad X\backslash_{\times}Y}{X/_{\times}Z} <\mathbf{B}_{\times}$$

- All arguments are type-raised via the lexicon:

$$\frac{X}{\mathrm{T}/(\mathrm{T}\backslash X)} >\mathbf{T} \qquad \frac{X}{\mathrm{T}\backslash(\mathrm{T}/X)} <\mathbf{T}$$

Combinatory Categorial Grammar (CCG)

- Combinatory Rules:

$$\frac{X/_{\star}Y : f \quad Y : g}{X : f(g)} > \qquad \frac{Y : g \quad X\backslash_{\star}Y : f}{X : f(g)} <$$

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

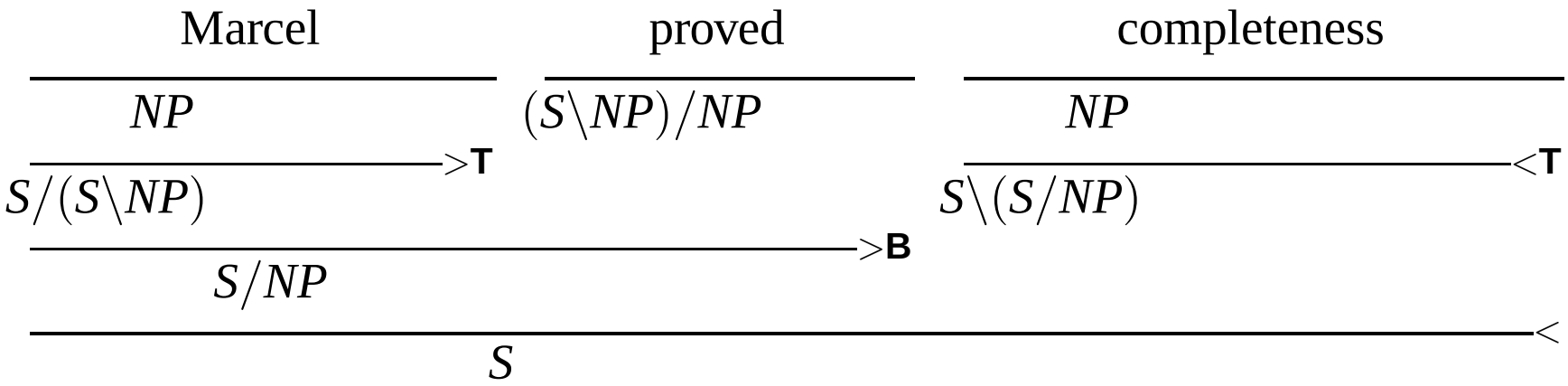
$$\frac{X/_{\times}Y : f \quad Y\backslash_{\times}Z : g}{X\backslash_{\times}Z : \lambda z.f(g(z))} >\mathbf{B}_{\times} \qquad \frac{Y/_{\times}Z : g \quad X\backslash_{\times}Y : f}{X/_{\times}Z : \lambda z.f(g(z))} <\mathbf{B}_{\times}$$

- All arguments are type-raised via the lexicon:

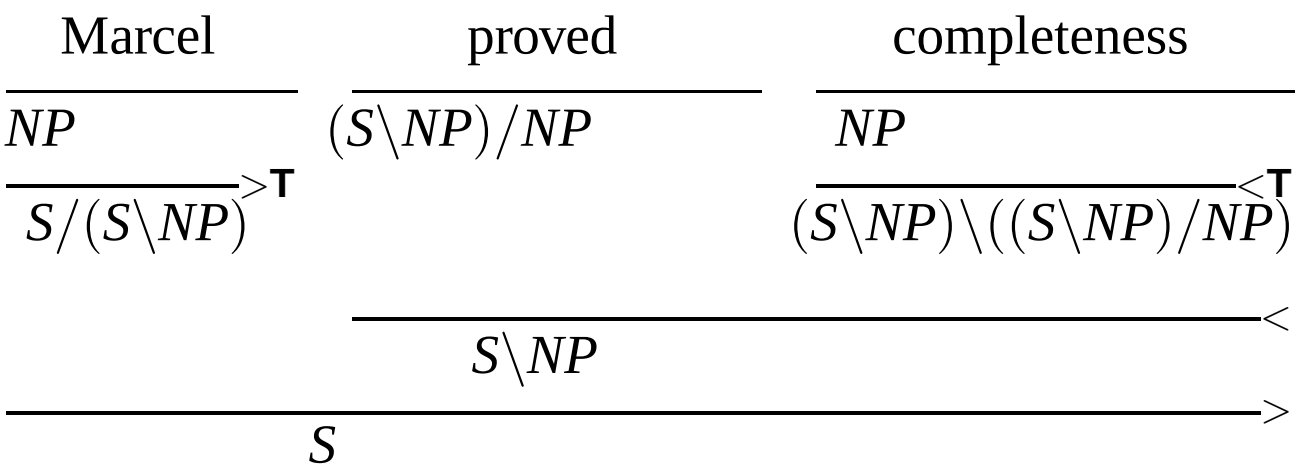
$$\frac{X : x}{\mathrm{T}/(\mathrm{T}\backslash X)\lambda f.f(x)} >\mathbf{T} \qquad \frac{X : x}{\mathrm{T}\backslash(\mathrm{T}/X) : \lambda f.f(x)} <\mathbf{T}$$

Combinatory Derivation

(6)



(7)



Combinatory Derivation

(6)

$$\dfrac{\dfrac{\dfrac{\text{Marcel}}{NP : marcel'}}{\dfrac{S/(S\backslash NP) : \lambda f.f\ marcel'}{>\mathbf{T}}} \quad \dfrac{\dfrac{\text{proved}}{(S\backslash NP)/NP : prove'}}{} \quad \dfrac{\dfrac{\text{completeness}}{NP : completeness'}}{\dfrac{S\backslash(S/NP) : \lambda p.p\ completeness'}{<\mathbf{T}}}}{\dfrac{S/NP : \lambda x.prove'\ x\ marcel'}{>\mathbf{B}} \quad \dfrac{}{}}$$

$$\dfrac{}{<}$$

$$S : prove'\ completeness'\ marcel'$$

(7)

$$\dfrac{\dfrac{\dfrac{\text{Marcel}}{NP : marcel'}}{\dfrac{S/(S\backslash NP)}{: \lambda f.f\ marcel'}>\mathbf{T}} \quad \dfrac{\dfrac{\text{proved}}{(S\backslash NP)/NP : prove'}}{} \quad \dfrac{\dfrac{\text{completeness}}{NP : completeness'}}{\dfrac{(S\backslash NP)\backslash((S\backslash NP)/NP)}{: \lambda p.p\ completeness'}<\mathbf{T}}}{\dfrac{S\backslash NP : \lambda y.prove'\ completeness'\ y}{<}}$$

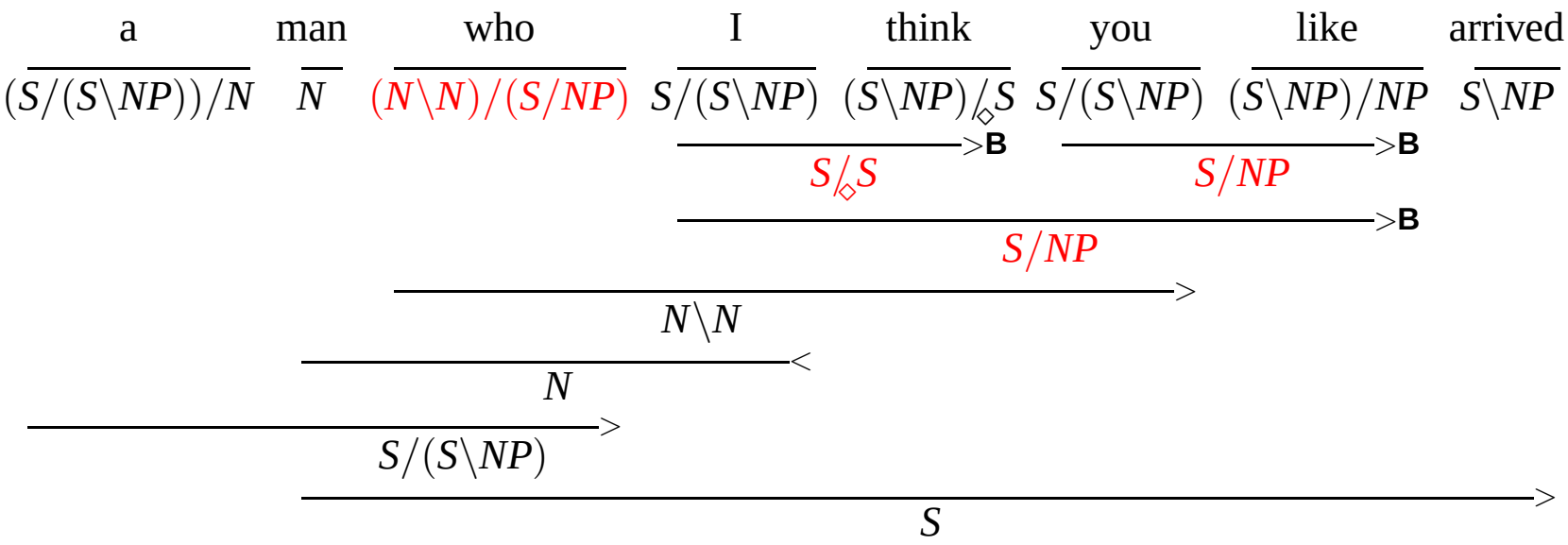
$$\dfrac{}{>}$$

$$S : prove'\ completeness'\ marcel'$$

Linguistic Predictions: Unbounded "Movement"

- The combination of type-raising and composition allows derivation to project lexical function-argument relations onto "unbounded" constructions such as relative clauses and coordinate structures, without transformational rules:

(8)



Predictions: Argument-Cluster Coordination

- The following construction is predicted on arguments of symmetry.

(9) give a teacher an apple and a policeman a flower

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{DTV}\ \dfrac{}{TV\backslash DTV}\,{<}\mathbf{T}\ \dfrac{}{VP\backslash TV}\,{<}\mathbf{T}}{VP\backslash DTV}\,{<}\mathbf{B}\quad \dfrac{}{CONJ}\quad \dfrac{\dfrac{}{TV\backslash DTV}\,{<}\mathbf{T}\quad \dfrac{}{VP\backslash TV}\,{<}\mathbf{T}}{VP\backslash DTV}\,{<}\mathbf{B}}{VP\backslash DTV}\,{<}\Phi{>}}{VP}\,{<}$$

- $VP = S\backslash NP;\ TV = (S\backslash NP)/NP;\ DTV = ((S\backslash NP)/NP)/NP$

- The derivation of utterance(9) is isomorphic to the process of composing a plan for another's action from the affordances of teachers, apples, (etc.), in a situation affording the plan by a speaker who desires its side-effects.

Syntax = Type-Raising and Composition

- CCGs combination of type-raising and composition yields a “mildly context sensitive” permuting and rebracketing calculus closely tuned to the needs of natural grammar.
- The argument cluster coordination construction (9) is an example of a universal tendency for “deletion under coordination” to respect basic word order: in all languages, if arguments are on the left of the verb then argument clusters coordinate on the left, if arguments are to the right of the verb then argument clusters coordinate to the right of the verb (Ross 1970):

(10) SVO: *SO and SVO SVO and SO

VSO: *SO and VSO VSO and SO

SOV: SO and SOV *SOV and SO

These Things are Out There in the Treebank

- Full Object Relatives (570 in WSJ treebank)
- Reduced Object Relatives (1070 in WSJ treebank)
- Argument Cluster Coordination (230 in WSJ treebank):

```
(S (NP-SBJ It)
   (VP (MD could)
       (VP (VP (VB cost)
               (NP-1 taxpayers)
               (NP-2 $ 15 million))
           (CC and)
           (VP (NP=1 BPC residents)
               (NP=2 $ 1 million)))))
```

These Things are Out There (contd.)

- Parasitic Gaps (at least 6 in WSJ treebank):

```
(S (NP-SBJ Hong Kong's uneasy relationship with China)
   (VP (MD will)
       (VP (VP (VB constrain)
               (NP (-NONE- *RNR*-1)))
           (PRN (: --)
                (IN though)
                (VP (RB not)
                    (VB inhibit)
                    (NP (-NONE- *RNR*-1)))
                (: --))
           (NP-1 long-term economic growth))))
```

CCG is Only Just Trans-Context Free

- CCG is provably weakly equivalent to Linear Indexed Grammar (LIG) Joshi *et al.* (1991).
- Hence it is not merely "Mildly Context Sensitive" (Joshi 1988) but rather "Nearly Context Free," or "Type 1.$\dot{9}$" in the Extended Chomsky Hierarchy.

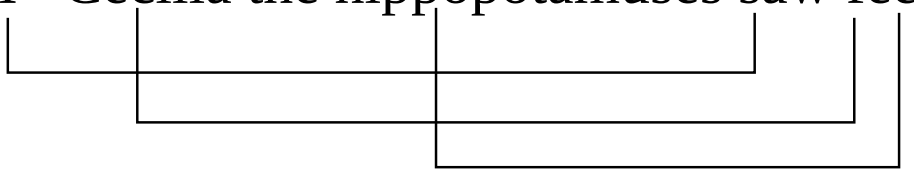
Language Type	**Automaton**	**Rule-types**	**Exemplar**
Type 0: RE	Universal Turing Machine	$\alpha \rightarrow \beta$	
Type 1: CS	Linear Bound Automaton (LBA)	$\phi A \psi \rightarrow \phi \alpha \psi$	a^{2^n}
"Type 1.99̇: LI"	Embedded PDA (EPDA)	$A_{[(i),\ldots]} \rightarrow \phi B_{[(i),\ldots]} \psi$	$a^n b^n c^n$
Type 2: CF	Push-Down Automaton (PDA)	$A \rightarrow \alpha$	$a^n b^n$
Type 3: FS	Finite-state Automaton (FSA)	$A \rightarrow \begin{cases} \text{a B} \\ \text{a} \end{cases}$	a^n

A Trans-Context Free Natural Language

- CCG can capture unboundedly crossed dependencies in Dutch:

... omdat ik Cecilia de nijlpaarden zag voeren.

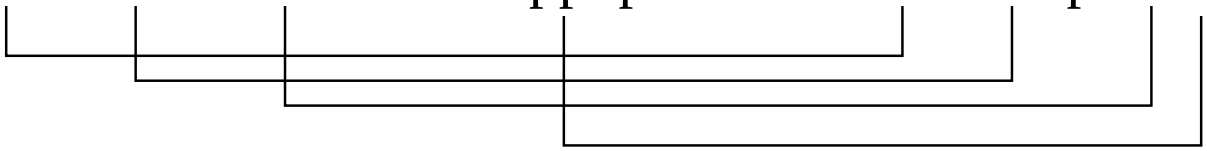
... because I Cecilia the hippopotamuses saw feed



'... because I saw Cecilia feed the hippopotamuses.'

... omdat ik Cecilia Henk de nijlpaarden zag helpen voeren.

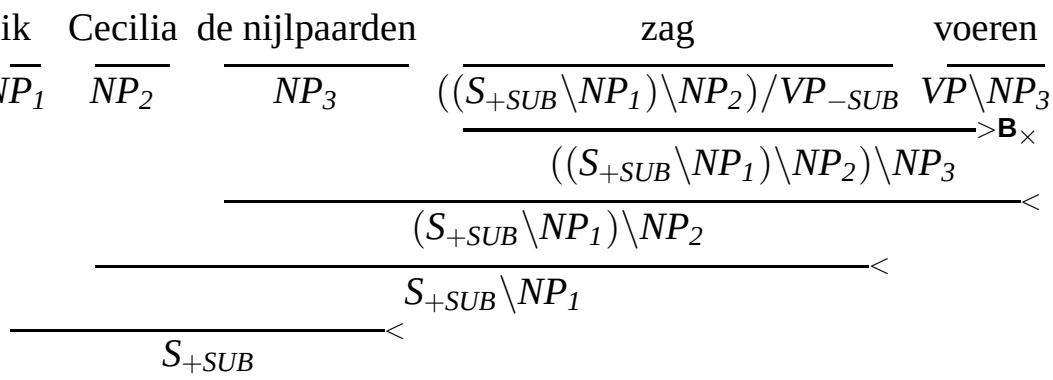
... because I Cecilia Henk the hippopotamuses saw help feed



'... because I saw Cecilia help Henk feed the hippopotamuses.'

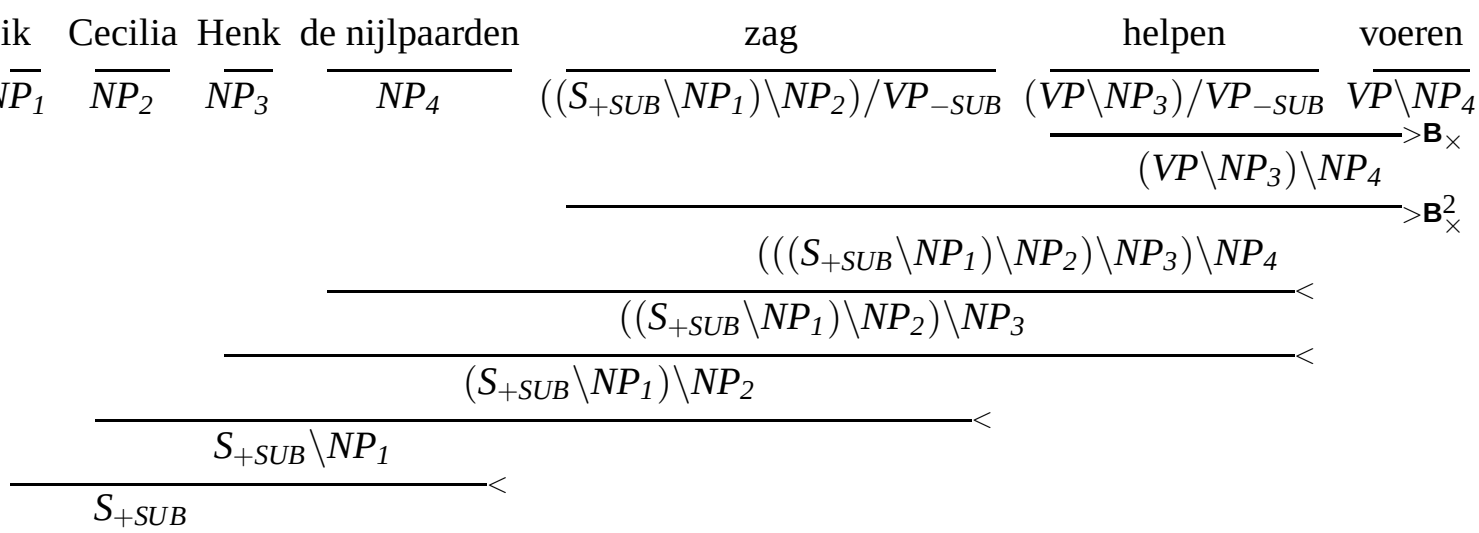
A Trans-Context Free Natural Language

(11) dat ik Cecilia de nijlpaarden zag voeren



A Trans-Context Free Natural Language

(12) dat ik Cecilia Henk de nijlpaarden zag helpen voeren



- The parallel involvement of type-raising **T** and composition **B** in planning and grammar suggest that the latter is evolutionarily and developmentally a transparent attachment to the former.

CCG is Just Trans-Context Free (contd.)

- It has polynomial parsing complexity (Vijay-Shanker and Weir 1990)
- Hence it has nice “Divide and Conquer” algorithms, like CKY, and Dynamic Programming.
- For real-life sized examples like parsing the newspaper, such algorithms must be statistically optimized.

References

Joshi, Aravind, 1988. "Tree Adjoining Grammars." In David Dowty, Lauri Karttunen, and Arnold Zwicky (eds.), *Natural Language Parsing*, Cambridge: Cambridge University Press. 206–250.

Joshi, Aravind, Vijay-Shanker, K., and Weir, David, 1991. "The Convergence of Mildly Context-Sensitive Formalisms." In Peter Sells, Stuart Shieber, and Tom Wasow (eds.), *Processing of Linguistic Structure*, Cambridge, MA: MIT Press. 31–81.

Ross, John Robert, 1970. "Gapping and the Order of Constituents." In Manfred Bierwisch and Karl Heidolph (eds.), *Progress in Linguistics*, The Hague: Mouton. 249–259.

Vijay-Shanker, K. and Weir, David, 1990. "Polynomial Time Parsing of Combinatory Categorial Grammars." In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics, Pittsburgh*. San Francisco, CA: Morgan Kaufmann, 1–8.