

“An introduction to multiple context free grammars for linguists” (Clark, 2014)

- Some language phenomena can't be modeled with CFGs.
- In 1991, Seki et al. come up with MCFGs, an extension of CFGs that allows us to work with discontinuous constituents or displaced elements.
- MCFGs are mildly context-sensitive grammars, as TAGs and CCGs. They are between Type-1 and Type-0 (Chomsky's hierarchy).
- MCFGs are equivalent to (some sort of) Minimalist Grammars (MGs).
- In his article, Clark provides a “non-technical explanation of MCFGs” and aims to account “how these can give a natural treatment of the types of syntactic phenomena that in mainstream generative grammar have been treated using movement”.

'Top-down' notation

Bottom-up notation

$$L = \{a^n b^n \mid n > 0\}$$

$$S \rightarrow A, S, B$$

$$S \rightarrow A, B$$

$$A \rightarrow a$$

$$B \rightarrow b$$

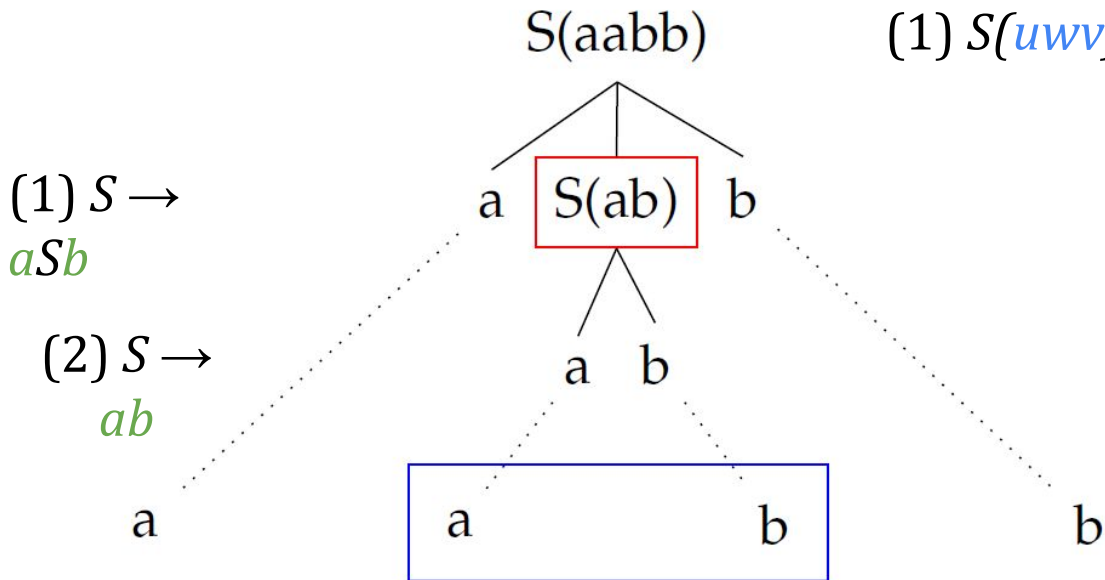
$$S(uvw) \leftarrow A(u), S(w), B(v)$$

$$S(uv) \leftarrow A(u), B(v)$$

$$A(a) \leftarrow$$

$$B(b) \leftarrow$$

Top-down notation



Bottom-up notation

$$(1) S(uvw) \leftarrow A(u), S(w), B(v)$$

$u = 'a', w = 'ab', v = 'b'$

$$(2) S(ab)$$

Motivation

In order to manipulate the way in which the yields (i.e. the sequence of words) of two or more constituents are concatenated when they form a more complex constituent, it is useful to refer to them as variables that satisfy a predicate: $P(x)$, where the predicate is a non-terminal symbol.

$$N(uv) \leftarrow P(u)Q(v)$$

or...

$$P \Rightarrow^* u$$

$$Q \Rightarrow^* v$$

$$N \Rightarrow^* uv$$

“In the case of this new format, we express the concatenation explicitly using variables”
(p. 5)

It's worth noting...

- In CFGs, the order of the strings derived by the non-terminals was implicitly encoded. Now it is not.

$$N(uv) \leftarrow P(u)Q(v) \neq N(uv) \leftarrow Q(v)P(u)$$

Linearity

- “Each variable on the right-hand side of the rule occurs exactly once on the left-hand side of the rule – and there aren't any other variables.”
- $N(uv) \leftarrow P(u),Q(v)$ is linear. $N(vu) \leftarrow P(u),Q(v)$ is not.

MCFGs

- Their key novelty is the **two-place predicate**, i.e. a predicate that applies not just to one string, but to a pair of strings.

$$M(u, v)$$

- These two strings are ordered.
- The number of strings which a predicate applies to is called **dimension**.

MCFGs

- We will always want to have some “normal” predicates to generate “full sentences”.

$$S_1(uv) \leftarrow N_2(u, v)$$

- But this is also possible:

$$N_2(au, bv) \leftarrow N_2(u, v)$$

- And finally we need:

$$N_2(a, b).$$

On the notation

$$N_2(ux,vy) \leftarrow P_2(u,v), Q_2(x,y)$$

We can no longer express the concatenation with solely the order of the symbols at the right-hand side.

$$N_2(xu,yv)$$

$$N_2(xuy,v)$$

$$N_2(x,uvy)$$

...

Even if we limit ourselves to **linear** and **non-permuting** rules:

$$N_2(uux,vvy) \leftarrow P_2(u,v), Q_2(x,y) \times$$

$$N_2(vy,ux) \leftarrow P_2(u,v), Q_2(x,y) \times$$

↑ ↑
ordered pair!

Formal definition (Stabler)

let's define an MCFG $G = \langle \Sigma, N, P, S \rangle$ with an infinite set X of variables x_i as follows:

Σ is a finite nonempty set of vocabulary elements

N is a finite nonempty set of categories A , where each has dimension $\dim(A) > 0$

P is a set of rules of the following form, for $n \geq 0$

$$B_0(\alpha_1, \dots, \alpha_{k_0}) \leftarrow B_1(x_{1,1}, \dots, x_{1,k_1}) \dots B_n(x_{n,1}, \dots, x_{n,k_n})$$

where

- for each $0 \leq i \leq n$, $\dim(B_i) = k_i$

- each $x_{i,j}$ is a variable

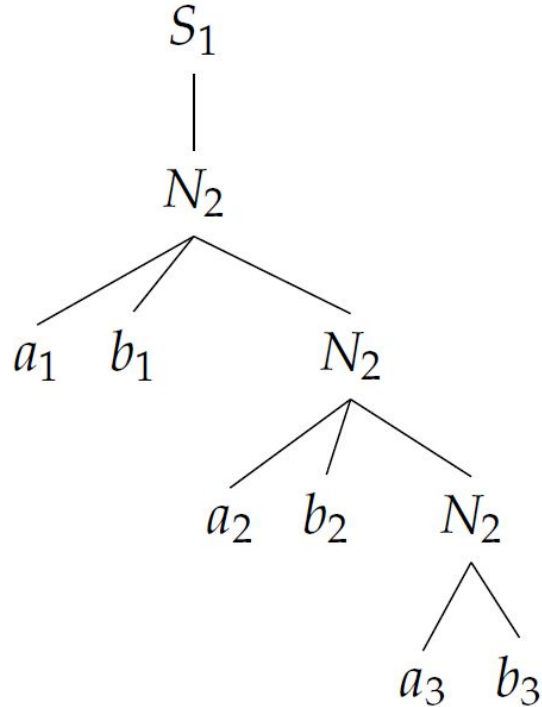
- each $\alpha_i \in (\Sigma \cup X)^*$

- each variable occurs at most once on the right, and at most once on the left

- all variables on the left appear on the right

$S \in N$ the start category, with $\dim(S) = 1$

Again... $L = \{d^n b^n \mid n > 0\}$



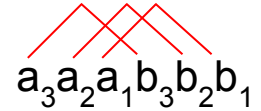
$$S_1(uv) \leftarrow N_2(u, v)$$

$$N_2(au, bv) \leftarrow N_2(u, v)$$

$$N_2(au, bv) \leftarrow N_2(u, v)$$

$$N_2(a, b).$$

same distance!

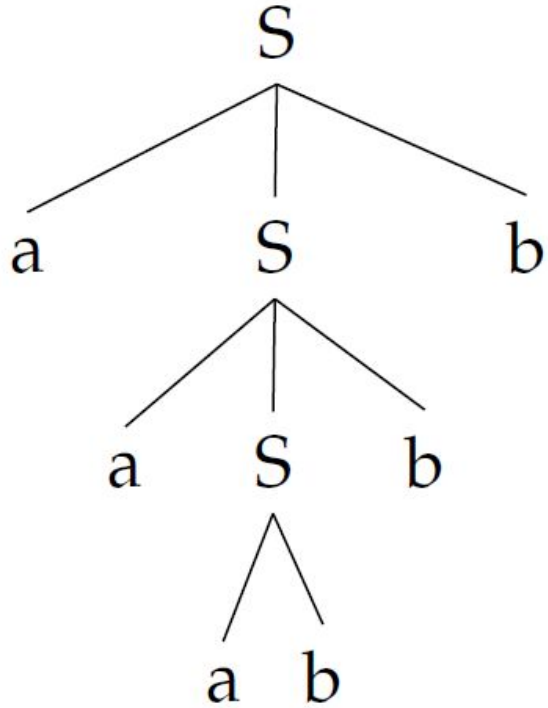


$a_3 a_2 a_1, b_3 b_2 b_1$

$a_2 a_1, b_2 b_1$

a_1, b_1

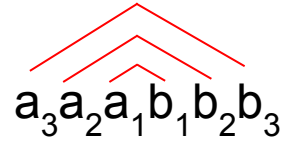
Weak equivalence, but not strong equivalence



$$S \rightarrow aSb$$

$$S \rightarrow aSb$$

$$S \rightarrow ab$$



$$a_2 a_1 b_1 b_2$$


$$a_1 b_1$$

Why (some sort of) context-freeness is preserved

- “The validity of a step in the derivation does not depend on the context that the symbol occurs in, only on the symbol that is being processed. Thinking of it in terms of trees, this means that if we have a valid derivation with a subtree headed with some symbol, whether it is of dimension one or two, we can freely swap that subtree for any other subtree that has the same symbol as its root, and the result will be also be a valid derivation” (p. 13)

Linguistic example: the cross-serial dependencies from Swiss German

... das mer d'chind em Hans es huus lönd hälfe aastrüiche
... that we the children-ACC Hans-DAT house-ACC let help paint



‘... that we let the children help Hans paint the house’

... *mer d'chind de Hans es huus
... we the children-ACC Hans-ACC the house-ACC

✗ $CN_d V_a$

✗ $CN_a N_d V_d V_a$

Rules for cross-dependency structures

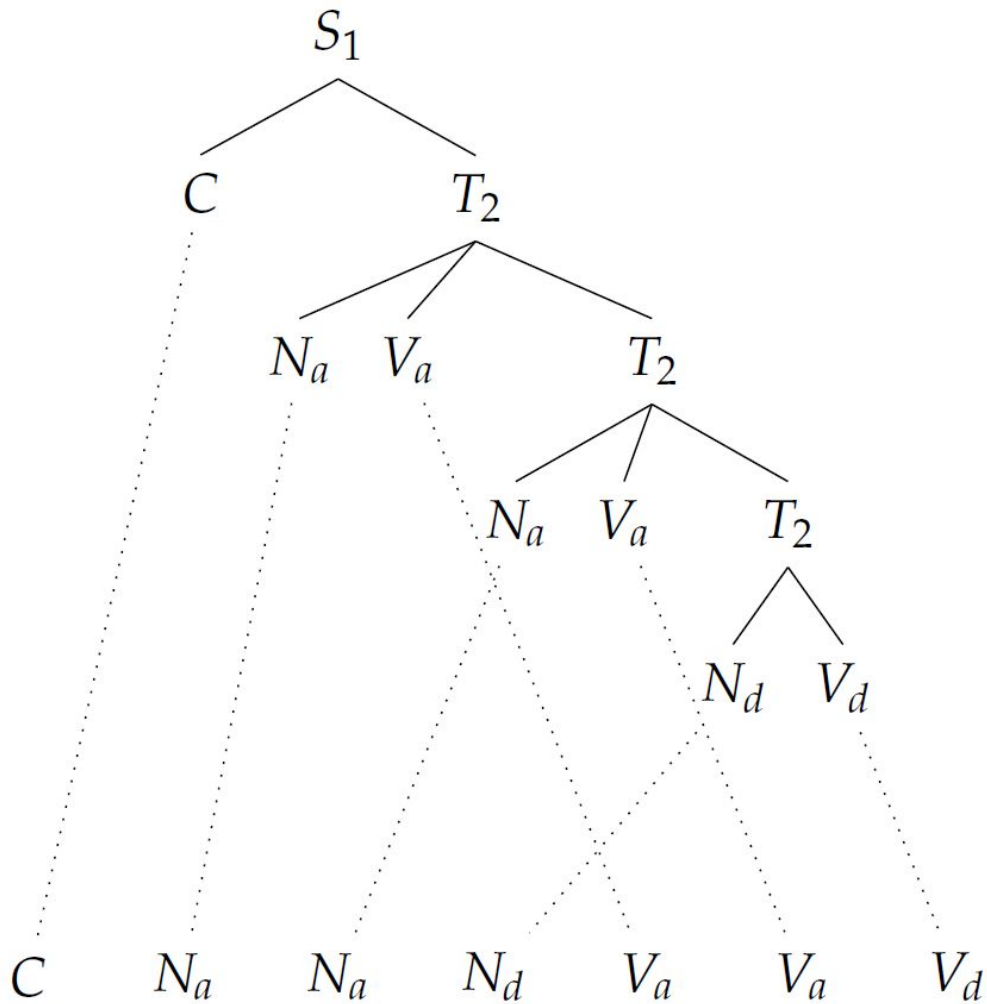
$$(17) \quad S_1(Cuv) \leftarrow T_2(u, v) \quad \vdots \quad S_1(uv) \leftarrow N_2(u, v)$$

$$(18) \quad T_2(N_a, V_a) \quad \vdots \quad N_2(a, b).$$

$$(19) \quad T_2(N_d, V_d)$$

$$(20) \quad T_2(N_a u, V_a v) \leftarrow T_2(u, v) \quad \vdots \quad N_2(au, bv) \leftarrow N_2(u, v)$$

$$(21) \quad T_2(N_d u, V_d v) \leftarrow T_2(u, v)$$

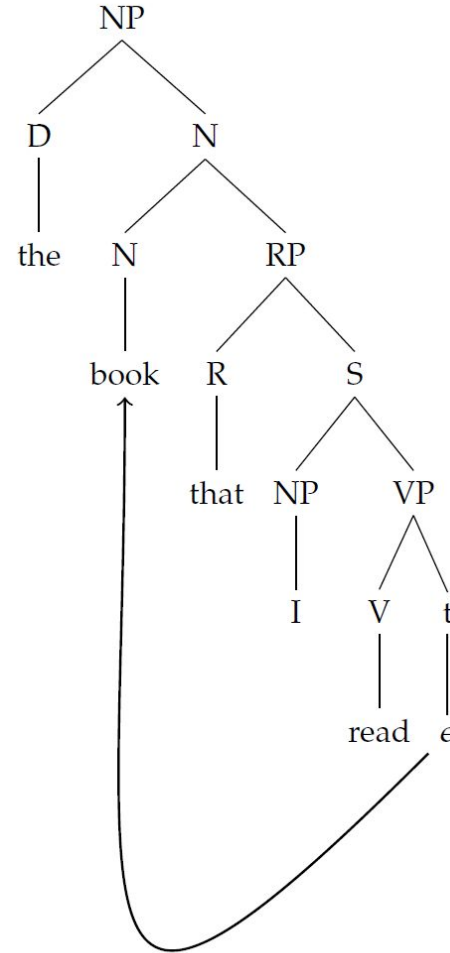


The symbols that are derived at the same step of the derivation (i.e. discontinuous constituents) establish the right dependency.

Relative clauses

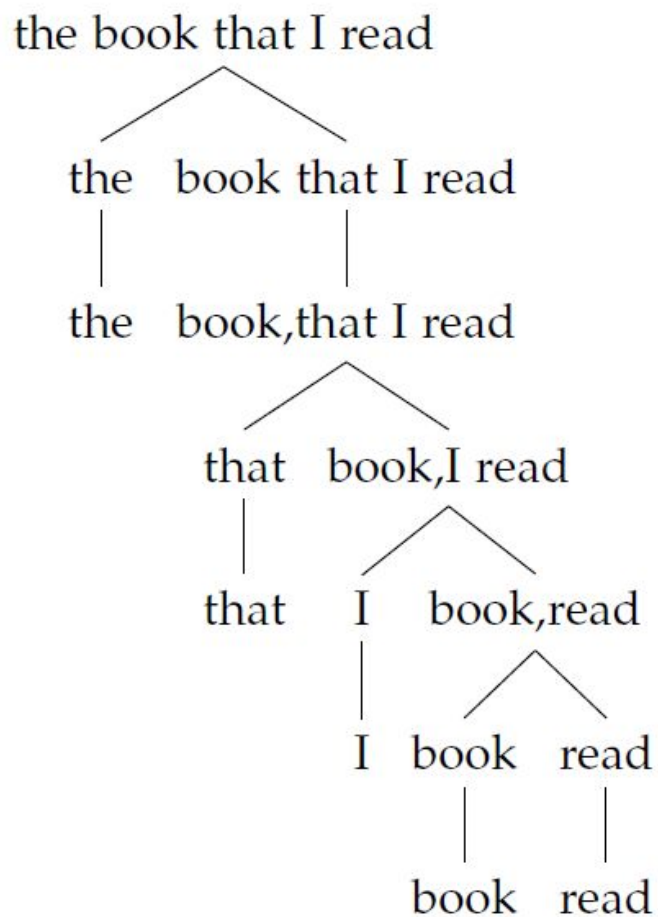
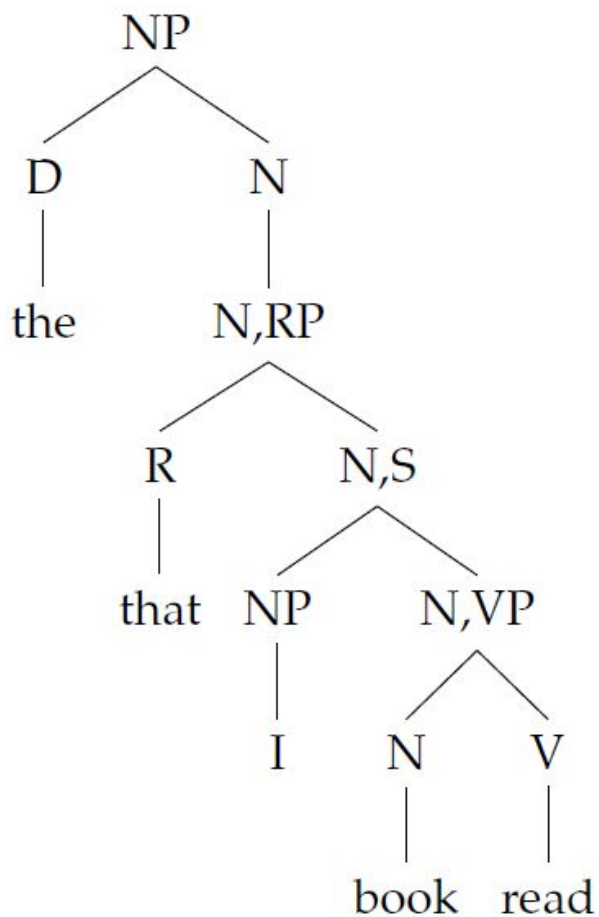
“The book that I read ϵ ”

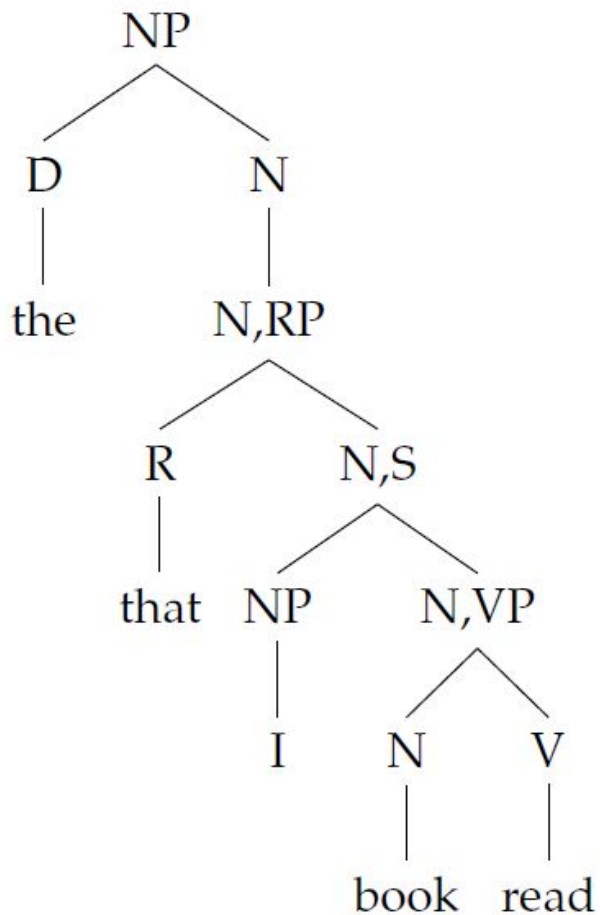
- The displaced constituent ‘moves’ upwards leaving a trace -or a copy.
- It is interpreted in the lower position and pronounced in the higher one.



How movement looks like in MCFGs

1. We derive the element in its 'semantic' (original) position, but instead of concatenating its 'phonetic features,' these are held in a *waiting position*.
2. When the moving string gets to its landing position, it is concatenated with the main string.





$$NP_1(uv) \leftarrow D_1(u), N_1(v)$$

$$N_1(uv) \leftarrow NRP_2(u, v)$$

$$NRP_2(u, wv) \leftarrow R_1(w), NS_2(u, v)$$

$$NS_2(u, wv) \leftarrow NP_1(w), NVP_2(u, v)$$

$$NVP_2(u, v) \leftarrow N_1(u), V_1(v)$$

General rule for moving constituent

$$MX_2(u, wv) \leftarrow Y_1(w), MZ_2(u, v).$$

- We could have infinite rules of this type (“This is the book that I told you Bob said I should read.”, and so on).

Alternative with trace

$$NVP_2(u_1, vu_2) \leftarrow NT_2(u_1, u_2), V_1(v)$$

$$NT_2(u, v) \leftarrow N_1(u), T_1(v)$$

Equivalence between MCFGs and TAGs

MCFGs are closely related to TAGs, but the latter are more restrictive, as their derivation trees will only use well-nested rules.

$$N_2(ux, vy) \leftarrow P_2(u, v), Q_2(x, y) \quad \times$$

$$N_2(ux, yv) \leftarrow P_2(u, v), Q_2(x, y) \quad \checkmark$$

$$N_2(xu, vy) \leftarrow P_2(u, v), Q_2(x, y) \quad \checkmark$$

Equivalence between MCFGs and MGs

Michaelis (2001) noticed that for every MG G , there is an MCFG G' which is not only weakly equivalent but also strongly equivalent in the sense that there is an isomorphism between the derivations of G and G' for every string in $L(G)$.

$$\frac{s :: =f\gamma \quad t \cdot f, t_1 : \alpha_1, \dots, t_k : \alpha_k}{st : \gamma, t_1 : \alpha_1, \dots, t_k : \alpha_k}$$



$$\gamma, \alpha_1, \dots, \alpha_k : (st, t_1, \dots, t_k) \leftarrow =f\gamma :: (s) \quad f, \alpha_1, \dots, \alpha_k(t, t_1, \dots, t_k)$$

Stabler's rules of inference

$$\frac{s ::= f\gamma \quad t \cdot f, \alpha_1, \dots, \alpha_k}{st : \gamma, \alpha_1, \dots, \alpha_k} \text{merge1: lexical item selects a non-mover as complement}$$

$$\frac{s := f\gamma, \alpha_1, \dots, \alpha_k \quad t \cdot f, \iota_1, \dots, \iota_l}{ts : \gamma, \alpha_1, \dots, \alpha_k, \iota_1, \dots, \iota_l} \text{merge2: derived item selects a non-mover as specifier}$$

$$\frac{s \cdot = f\gamma, \alpha_1, \dots, \alpha_k \quad t \cdot f\delta, \iota_1, \dots, \iota_l}{s : \gamma, \alpha_1, \dots, \alpha_k, t : \delta, \iota_1, \dots, \iota_l} \text{merge3: any item selects a mover}$$

$$\frac{s : +f\gamma, \alpha_1, \dots, \alpha_{i-1}, t : -f, \alpha_{i+1}, \dots, \alpha_k}{ts : \gamma, \alpha_1, \dots, \alpha_{i-1}, \alpha_{i+1}, \dots, \alpha_k} \text{move1: final move of licensee}$$

$$\frac{s : +f\gamma, \alpha_1, \dots, \alpha_{i-1}, t : -f\delta, \alpha_{i+1}, \dots, \alpha_k}{s : \gamma, \alpha_1, \dots, \alpha_{i-1}, t : \delta, \alpha_{i+1}, \dots, \alpha_k} \text{move2: nonfinal move of licensee}$$

Move-1: final move of licensee

that I read : +k RP book :: -k
book that I read : RP

RP :₁ (book that I read) ← +k RP : , -k ::₂ (book, that I read)

Merge-1: lexical item selects a non-mover as complement

that :: =S +k RP I read : S, book :: -k
that I read : +k RP, book :: -k

+k RP : , -k ::₂ (book, that I read) ← =S +k RP ::₁ (that), S : , -k ::₂ (book, I read)

Merge-2: derived item selects non-mover as specifier

read : =NP S, book :: -k I : NP
S : I read, book :: -k

S : , -k ::₂ (book, I read) ← NP :₁ (I), =NP S :: , -k :₂
(book, read)

Merge-3: any item selects a mover

read :: =N =NP S book :: N -k
read : =NP S, book :: -k

=NP S : , -k ::₂ (book, read) ← =N =NP S ::₁ (read), N -k ::₁
(book)

Time complexity of an MCFG parsing algorithm

- Depends on the **rank r** of a grammar (maximum number of non-terminals that we have on the right hand side of a rule) and **dimension d** (number of chunks of the predicate).
- MCFGs can be parsed in time $n^{(r+1)d}$.
- CFGs (in Chomsky normal form) have $d = 1$ and $r = 2$: n^3 .
- MCFGs with $d = 2$ and $r = 2$: n^6 .

Conclusion

- MCFGs provide a way of dealing with movement and discontinuous constituencies in polynomial time, preserving -to a large degree- the structural descriptions of traditional generative analyses.
 - As a consequence, MGs are computationally tractable as well.
- MCFGs show that there is not necessarily a ‘technical’ difference between formalisms that use movement and formalisms that don’t.
- (Well-nested) MCFGs are weakly equivalent to TAGs and CCGs -among other formalisms-, which means that they have the properties of MCSLs.
- These equivalences “suggest that MCFGs define the right combinatorial operations to model natural language syntax.”

References

- Clark, A. (2014). “An introduction to multiple context free grammars for linguists”. Available at: <https://alexc17.github.io/static/pdfs/mcfgsforlinguists.pdf>
- Michaelis, J. (2001). “Transforming linear context-free rewriting systems into minimalist grammars”. *Logical Aspects of Computational Linguistics*, pp. 228–244. Springer.
- Seki, H., Matsumura, T., Fujii, M., and Kasami, T. “On multiple context-free grammars”. (1991). *Theoretical Computer Science*, 88(2):229, 1991
- Stabler, E. (2013). “Computational Linguistics: Defining, calculating, using, and learning linguistic structures”. Lx185/209 lecture notes. Available at: <https://linguistics.ucla.edu/people/stabler/185-13.pdf>