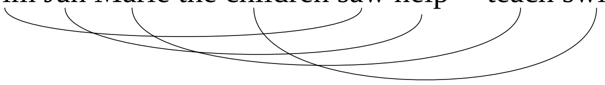
Comparing Lexicalized Grammar Formalisms in an Empirically Adequate Way: The Notion of Generative Attachment Capacity

Laura Kallmeyer
SFB 441 – University of Tübingen, Germany
lk@sfs.uni-tuebingen.de

The work presented here addresses the question of how to determine whether a grammar formalism is powerful enough to describe natural languages. The expressive power of a formalism can be characterized in terms of i) the string languages it generates (*weak generative capacity (WGC)*) or ii) the tree languages it generates (*strong generative capacity (SGC)*). The notion of WGC is not enough to determine whether a formalism is adequate for natural languages. We argue that even SGC is problematic since the sets of trees a grammar formalism for natural languages should be able to generate is difficult to determine. The concrete syntactic structures assumed for natural languages depend very much on theoretical stipulations and empirical evidence for syntactic structures is rather hard to obtain. Therefore, for lexicalized formalisms, we propose to consider the ability to generate certain strings together with specific predicate argument dependencies as a criterion for adequacy for natural languages.

Weak, strong and derivational generative capacity The WGC, i.e., the sets of string languages a formalism generates, is not enough to determine whether a formalism is powerful enough for natural languages: CFG can generate the string set of cross-serial dependencies in Dutch. But (see Fig. 1) these strings do not show the desired dependencies (i.e., arguments are added to predicates they do not depend on).

(1) ... dat Wim Jan Marie de kinderen zag helpen leren zwemmen
... that Wim Jan Marie the children saw help teach swim



'... that Wim saw Jan help Marie teach the children to swim'

Besides WGC, SGC was introduced, a characterization of the expressive power of a formalism by the set of tree languages it generates. Computationally this is a very useful notion. Bresnan et al. (1982) use the notion of SGC to argue that CFGs are not able to describe the cross-serial dependencies in Dutch. We think however that the set of trees necessary for natural languages is difficult to determine for two reasons:

S → NP S zwemen
S → NP S leren
S → NP S helpen
S → NP zag
NP → Wim
NP → Jan
NP → Marie
NP → de kinderen

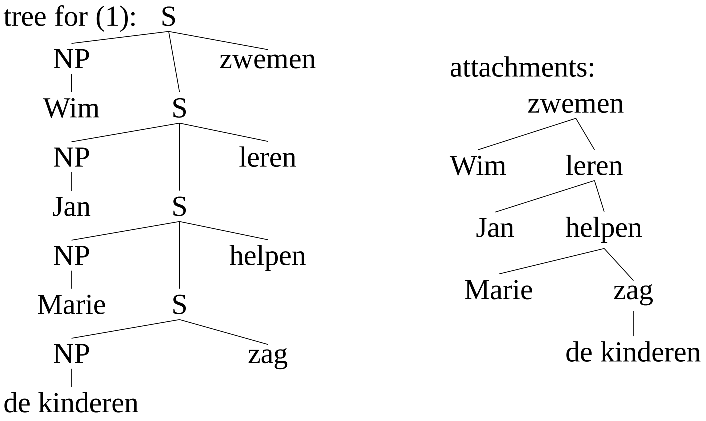


Figure 1: CFG for cross-serial dependencies displaying the wrong attachments

1. the question whether a formalism is adequate for natural languages should be decided as independent as possible from any concrete linguistic theory. The choice of syntactic structures however depends very much on the theory behind. Therefore each argument that a certain formalism cannot generate natural languages because it cannot generate the syntactic structures needed is a little weak since it does not show that it is in general not possible (by adopting a different syntactic theory) to find a natural language grammar using the formalism in question. 2. the requirements for a grammar formalism for natural languages should, if possible, be empirically observable. With constituent tests, syntactic structure is observable to a certain degree but in some cases syntactic structure follows only from theoretical stipulations without a solid empirical foundation. Even constituent tests make theoretical stipulations. Therefore we think SGC not adequate to determine whether a formalism can generate natural languages.

For generative grammars, Becker et al. (1992) introduce the *derivational generative capacity (DGC)*, a characterization in terms of so-called *indexed languages*. In these grammars, for each string a derivation with a certain number of steps is performed. Each lexical item in the string receives an index, the number of the derivation step in which it was added. The sets of indexed strings one can generate with the formalism determines the DGC. We think that this is not adequate for natural languages either since rather than considering which items are added in which derivation step one should consider which item is added to (i.e., is argument of) which item.

Generative attachment capacity We therefore introduce the notion of *generative attachment capacity (GAC)* for lexicalized generative grammars (i.e., the grammar consists of a set of elementary objects associated with lexical items; larger objects are derived by putting the elementary objects together): For each derivation of some string w, the *attachment structure* is a graph containing as nodes the occurrences of terminal symbols t in w such that there is an edge between t_1 and t_2 iff one of them was added to the other in the course of the derivation. The set of sets of pairs of strings and corresponding attachment structures a formalism can generate determines its GAC.

For natural languages, the attachment structures should contain all predicate argument links. I.e., if we use the notion of GAC the question whether a formalism is adequate for natural languages amounts to asking whether it can generate all strings with the correct predicate argument dependencies. This is a very useful characterization since 1. it is empirically more appropriate than SCG because linguistic evidence for predicate argument dependencies is much easier to obtain than for syntactic structures, and 2. attachment structures are crucial for semantics since they are at the heart of the syntax-semantics interface.

With the GAC, the CFG in Fig. 1 is not adequate for Dutch cross-serial dependencies since it does not generate the dependencies in Fig. 2. We can in fact show that lexicalized CFGs can generate $\{n^k v^k \,|\, k > 0\}$ only with nested dependencies but not with crossed dependencies as in Fig. 2: assume that there is a lexicalized CFG for the language in Fig. 2. Then there is a nonterminal A in the rule introducing v_i where n_i gets attached. I.e., these rules are of the form $X \longrightarrow w_1 A w_2 v w_3$ or $X \longrightarrow w_1 v w_2 A w_3$ with $X \in N, w_1, w_2, w_3 \in (N \cup T)^*$. Replacing these rules with new rules $X \longrightarrow w_1 t w_2 t w_3$ for all $t \in T$ gives a new CFG that generates the copy language $\{ww \,|\, w \in T^*\}$. Contradiction since the copy language is not context-free.

String language $\{n^k v^k \,|\, k > 0\}$
with the following
attachment structure
for $w = n_1 n_2 \dots n_k v_1 v_2 \dots v_k$
(indices mark different
occurrences of a terminal):

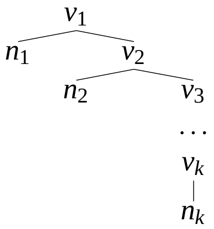


Figure 2: Language necessary for cross-serial dependencies

This work was much influenced by similar discussions in the context of Tree Adjoining Grammars (TAG), and the notion of DGC is actually very close to GAC. Under certain assumptions (e.g., the possibility to precompile substitution or, in other words, to replace preterminals in a CFG in advance by the corresponding terminals) it amounts to the same. However, in some grammar formalisms one cannot make these assumptions and then GAC is a better criterion for the expressive power needed.

References

Becker, T., O. Rambow, and M. Niv (1992). The Derivationel Generative Power of Formal Systems or Scrambling is Beyond LCFRS. Technical Report IRCS-92-38, Institute for Research in Cognitive Science, University of Pennsylvania.

Bresnan, J., R. M. Kaplan, S. Peters, and A. Zaenen (1982). Cross-serial dependencies in Dutch. *Linguistic Inquiry*, **13**(4):613–635.