# Comparing Left-Associative Grammar with PSG

Roland Hausser

Universitдt Erlangen-Nьrnberg
Abteilung Computerlinguistik (CLUE)
rrh@linguistik.uni-erlangen.de

This paper presents a basic alternative to phrase structure grammar, called left-associative grammar. While phrase structure grammars (PSG) are based on the principle of possible *substitutions*, left-associative grammar (LAG) is based on possible *continuations*.

In other words, LAG computes language input by always combining a sentence start with a next word into a new sentence start. This procedure continues until there is either no more next word in the input (end of sentence) or the current next word is not compatible with the current sentence start (ungrammatical continuation). If a next word can be combined with a sentence start in more than one way (syntactic ambiguity) or if a next word has more than one reading (lexical ambiguity), the resulting derivation paths are pursued in parallel.

The original motivation for developing LAG was to model the time-linear structure of natural language. It turned out, however, that the principle of computing possible continuations resulted in a language and complexity hierarchy which is orthogonal to that of PSG.

Of the PSG language classes, i.e. regular, context-free, context-sensitive, and recursively enumerable, only the regular and context-free are computationally tractable. However,

> It is no secret that context-free grammars are only a first order approximation to the various mechanisms used for specifying the syntax of modern programming languages.[1]
>
> S. Ginsberg 1980, p.7

Most attempts to arrive at new language classes have consisted in *conservative extensions*, however, which follow context-free PSG too closely. Based on adding certain mechanisms, they result in additional language classes which fit right into the subset relations of the PS-hierarchy. For example, the context-free languages form a proper subset of the tree adjoining languages (TALs),[2] which form a proper subset of the index languages,[3] which in turn form a proper subset of the context-sensitive languages.

That the language classes of PSG are not really natural is shown by formal languages $a^k b^k$, which is context-free, and $a^k b^k c^k$,which is context-sensitive (and thus in a class of exponential

[1] See also M. Harrison 1978, p. 219ff, in the same vein.
[2] A.K. Joshi et al. 1975.
[3] Hopcroft & Ullman 1979, p. 389 f. A pumping lemma for index languages proved T. Hayashi 1973.

complexity). Similarly, $WW^R$ is context-free, while WW is context-sensitive. This comes from the fact that the context-free languages are limited to pairwise inverse structures, like abc...cba, which in turn derives from the PSG rule schema of context-free languages, e.g. S → aSb. In PSG, anything that is not pairwise inverse is at least context-sensitive.

In comparison, let us illustrate the general format of LAG with the definition for $\mathsf{a}^k\mathsf{b}^k$.

### 0.1 LAG for context-free $\mathsf{a}^k\mathsf{b}^k$

LX $=_{def}$ {[a (a)], [b (b)]}
$ST_S =_{def}$ {[(a) {$r_1$, $r_2$}]}
$r_1$: (X) (a) ⇒ (aX) {$r_1$, $r_2$}
$r_2$: (aX) (b) ⇒ (X) {$r_2$}
$ST_F =_{def}$ {[$\varepsilon$ $rp_2$]}.

LX specifies the lexicon of the language, where by a word, e.g. [a (*a*)], is defined as an ordered pair of a surface, here a, and a category, here (*a*). The start state $ST_S$ specifies the kind of first word a derivation can start with, here in terms of the category (*a*), and the rules which can be used to add the next word, here the rule package {$r_1$, $r_2$}. The rules consist of a rule name, e.g. $r_1$, a pattern for the sentence start, e.g. (X), a pattern for the next word, e.g. (a), a pattern for the resulting sentence start, e.g. (aX), and a rule package. The final state $ST_F$ specifies the category of the result, here empty category, and the rule which applied last, represented by its rule package.

In context-free $\mathsf{a}^k\mathsf{b}^k$, the LAG collects the *a*'s in the category (rule $r_1$), and then substracts for each b in the input an *a* in the category. In context-sensitive $\mathsf{a}^k\mathsf{b}^k\mathsf{c}^k$ the procedure is similar, except that for each *a* subtracted at the beginning of the category, a *b* is added at its end, thus recycling the *a*'s as *b*'s to ensure that an equal number of c's follows in the input.

### 0.2 LAG for context-sensitive $\mathsf{a}^k\mathsf{b}^k\mathsf{c}^k$

LX $=_{def}$ {[a (a)], [b (b)], [c (c)]}
$ST_S =_{def}$ {[(a) {$r_1$, $r_2$}]}
$r_1$: (X) (a) ⇒ (aX) {$r_1$, $r_2$}
$r_2$: (aX) (b) ⇒ (Xb) {$r_2$, $r_3$}
$r_3$: (bX) (c) ⇒ (X) {$r_3$}
$ST_F =_{def}$ {[$\varepsilon$ $rp_3$]}.

Note that additional languages like $\mathsf{a}^k\mathsf{b}^k\mathsf{c}^k\mathsf{d}^k$... may be easily defined by extending the LAG 0.2. Also, LAG parses $\mathsf{a}^{i!}$, which is not even an index language, in linear time.

Hausser, R. (1999/2001) *Foundations of Computational Linguistics,*, 2nd Edition 2001, pp. 578, Berlin, New York: Springer-Verlag.