Formal Mechanisms for Movement and Surface Structure in Dutch

Annius Groenink
Centrum voor Wiskunde en Informatica
e-mail: `Annius.Groenink@cwi.nl`

Abstract

It has been argued many times that syntactical movement, in the form of leftward extraposition and wrapping discontinuous constituents, is of such a crucial importance that there is a need to develop grammar formalisms aimed solely at a clear and concise treatment of these phenomena.
Along the guiding lines of a number of examples of Dutch verb phrase structure, I first discuss head grammar (HG) and show that it inherently lacks the strong generative to describe even very simple fragments of complete Dutch sentences. I then make, in three steps, a progression from linear context-free rewriting systems (LCFRS) to literal movement grammar (LMG), and show that in contrast to the general feeling about LCFRS, the resulting formalism is an attractive and adequate tool for describing concrete fragments of configurational languages with a nontrivial surface structure.

1 Introduction

In recent years there has been considerable interest for 'light' grammar formalisms; most notably *tree adjoining grammar* (TAG), but also for *linear indexed grammar* (LIG) as a platform for implementing TAG parsing methods (Vijay-Shanker and Weir 1994) and as a basis for restricted unification grammars (Keller and Weir 1995), and to a lesser degree for *head grammar* (HG, Pollard 1984), its formal generalizations *linear context-free rewriting systems* (LCFRS, Weir 1988) and *parallel multiple context-free grammars* (PMCFG, Kaji et al. 1992), and finally *extraposition grammar* (XG, Pereira 1981).

These systems have in common that they are elementary extensions or progressions of the context-free grammars; indeed most of the formalisms mentioned generate *mildly context-sensitive* languages. One of the motivations for such grammar formalisms is that in order to study the real theory-independent nature of certain, especially structurally oriented, linguistic phenomena, it is interesting to investigate precisely how much formal power is necessary for an adequate description. Another motivation often mentioned is a dissatisfaction with the sometimes very *ad hoc* or feebly theoretically founded methods of describing movement found in popular feature-based frameworks, such as slash threading. For example, Pereira in (Pereira 1981) gives the following motivation for the XG formalism:

> "The importance of these [extraposition] constructions, even in simplified subsets of natural language, such as those used in database interfaces, suggests that a grammar formalism should be able to express them in a clear and concise manner."

Nevertheless, formalisms such as head grammar (and especially the weakly equivalent tree adjoining grammars) seem to be primarily interesting to abstract formal language theory, and as such have not been backed up by many examples of their concrete use in linguistic descriptions. Where such descriptions *are* given (e.g. in (Pollard 1984)), they are largely aimed at an account of English, or minimal, strongly isolated fragments of more complex languages such as Dutch. The complexity of English surface structure is too limited to give a faithful account of the adequacy of a surface structure description method. This paper shows that this can lead to (1) exceedingly *ad hoc* conceptions of discontinuous constituency and (2) formalisms that lack the (strong) generative capacity to give satisfactory structural descriptions of more surface-complex configurational languages such as Dutch and German.

The emphasis in this paper is on head grammar (HG), two extensions (LCFRS, LMG), and the view on linguistic structure imposed by those formalisms.

2 Extraposition and Discontinuous Constituency

Before a discussion of discontinuous constituency and syntactical movement, it is worth noting that it is a concept which strongly depends on the notion of a constituent itself. In transformational accounts of language, the meaning of the word constituent varies depending on what structure one is looking at (s-structure, d-structure, LF). In LFG, the sentence corresponding to a *c-structure* (assuming that the **c** stands for constituent) is read off the tree from left to right, hence a constituent in LFG always seems to form a contiguous substring of a sentence, and the existence of *discontinuous* constituents is meaningless if not contradictory.

The most wide-spread examples of what the literature calls a discontinuous constituent are probably phrases such as *hard to motivate* in

Frank is a **hard** person **to motivate**. (1)

The underlying idea is apparently that there is a generally accepted tree structure inspired by functional concepts such as heads, complements and modifiers

(X-bar theory if the reader wishes). Constituents are then obtained by putting together the words at the leaves of arbitrary subtrees. A sensible account of English will recognize *hard to motivate* as an adjectival phrase that modifies the nominal projection *person*, hence *person* and *hard to motivate* must be represented by separate subtrees, and they must be separate constituents.

Another way of reasoning is that we want the adjectival phrase *hard to motivate* in sentence (1) to be represented in the same way as in (2).

Frank is **hard to motivate**. (2)

There are roughly two explanations in existence. The transformational view is that *to motivate* is *extraposed* or *moved* rightward out of its deep-structural position. The other explanation calls it a phenomenon of *discontinuous constituency* and says that the constituent *hard to motivate* will *wrap* itself around a noun phrase when it modifies it.

Now consider the following sentences.

Did Eve **eat the apple** (3)

Eve **did eat the apple** (4)

An elegant description would probably consider the phrase headed by *eat* as a complement to *did*, and assign the same (deep-) structural analysis to *did eat the apple* in both examples. Again, this might equally well be explained as movement of the verb *did* or by saying that *did eat the apple* is a discontinuous VP constituent that can wrap itself around its subject.

The same remarks can be made about topicalized sentences. The following are borrowed from (Pollard 1984) on head grammar:

Smith **sent Jones to Minsk** (5)

Minsk, Smith **sent Jones to** (6)

Jones, Smith **sent to Minsk** (7)

Although it may be harsh to strictly pose that *sent Jones to Minsk* is a constituent in all three sentences, it seems reasonable to suppose that a grammar formalism designed to adequately describe movement and discontinuity should use its structural capacities to give a model of these sentences. However, the head grammar fragment sketched in (Pollard 1984) reverts (necessarily, as we will soon see) to a SLASH feature to treat topicalization.

3 Crossed Dependencies in Head Grammar

(Pollard 1984) presents an extensive treatment of discontinuous constituency in English, including examples such as *Frank is a hard person to motivate.* Pollard also gives a description of Dutch crossed dependencies in an appendix (a similar HG analysis is discussed in this paper). However, the accounts of English and Dutch are both limited: the fragment of English discussed does not treat auxiliary inversion, and that of Dutch only talks about the verb phrase, leaving verb second phenomena undiscussed.

Except for the very simple case of subject-auxiliary inversion, English is a language with a highly 'local' verb structure: sentences with embedded verbal projections always consist of verbs immediately followed by their complements:

$$\text{John } [_{\mathrm{VP}}[_{\overline{\mathrm{V}}} \text{ saw Mary } [_{\overline{\mathrm{V}}} \text{ teach Fred } [_{\overline{\mathrm{V}}} \text{ to swim }]]]] \tag{8}$$

For the study of movement and discontinuity phenomena, Dutch is a much more interesting language. Dutch is very strict about which surface forms are acceptable, yet it shows a great diversity in verb phrase order. The most frequent order is that of *crossed dependencies*: the verb phrase is split up into a *nominal cluster* and a *verb cluster*:

$$\ldots \text{ dat Jan } [_{\mathrm{VP}}[_{\mathrm{NC}} \text{ Marie Fred }] \; [_{\mathrm{VC}} \text{ zag leren zwemmen }]] \tag{9}$$

that ... saw teach swim

... that John saw Mary teach Fred to swim

However odd the *surface* structure of Dutch may be, the underlying functional head-complement structure is still present—be it that each $\overline{\mathrm{V}}$ is somehow divided into two parts: one that selects to appear in the NC, and one that selects to appear in the VC. This is exactly reflected in the way (Pollard 1984) describes Dutch crossed dependencies, and reflects the way in which the methods proposed in this paper will describe movement and discontinuity phenomena in general: the choice of constituents is fully motivated by their deep-structural characteristics, and in order to obtain the correct surface forms, we split up the *yield* of constituents into a number of *clusters* which select to appear in different positions in the generated sentence. We hence more or less abandon the concept of surface *structure*, taking deep structure as a point of departure and viewing surface forms as obtained from this deep structure by concatenation of the different clusters yielded by its constituents. This is equivalent to saying that the concepts of *movement* and *discontinuity* are not sufficiently general, and we should rather think of producing surface forms as giving rules for *placement* of clusters or "subconstituents" with no meaningful structural representation of their own.

Head Grammar assigns a special role to the position of the *head* of a constituent in the construction of its surface form from a deep-structural representation. It splits up the *yield* of a nonterminal into two parts, which can appear at different places in the derived string.

Definition. A (modified) head grammar (HG) is a tuple (N, T, S, P), where the productions in P are of the form $A \to \langle w_1, w_2 \rangle$ where $A \in N$, $w_1, w_2 \in T^*$ or $A \to f(B_1, B_2)$ where $B_1, B_2 \in N$, and the *yield function*, f, is one of the function symbols *wrap*, *concat*$_1$ or *concat*$_2$. A head grammar G recognizes pairs of terminal strings, as follows:

Base case If $A \rightarrow \langle w_1, w_2 \rangle$ is a grammar rule, then $A \overset{G}{\Longrightarrow} \langle w_1, w_2 \rangle$

Inductive case If $A \rightarrow f(B_1, B_2)$ is a rule in G, $B_1 \overset{G}{\Longrightarrow} \langle u_1, u_2 \rangle$ and $B_2 \overset{G}{\Longrightarrow} \langle v_1, v_2 \rangle$, then $A \overset{G}{\Longrightarrow} f(\langle u_1, u_2 \rangle, \langle v_1, v_2 \rangle)$ where

$$\begin{array}{lcl} \mathit{wrap}(\langle v_1, v_2 \rangle, \langle w_1, w_2 \rangle) & = & \langle w_1 v_1,\ v_2 w_2 \rangle \\ \mathit{concat}_1(\langle v_1, v_2 \rangle, \langle w_1, w_2 \rangle) & = & \langle v_1,\ v_2 w_1 w_2 \rangle \\ \mathit{concat}_2(\langle v_1, v_2 \rangle, \langle w_1, w_2 \rangle) & = & \langle v_1 v_2 w_1,\ w_2 \rangle \end{array}$$

The underlying intuition is that a tuple $\langle w_1, w_2 \rangle$, represents a constituent $w_1 w_2$ whose *head* is the first terminal of w_2.

$$\begin{array}{lcl} \overline{\mathrm{V}} & \rightarrow & \mathit{concat}_2(\mathrm{NP},\ \mathrm{VT}) \\ \mathrm{VT} & \rightarrow & \mathit{wrap}(\mathrm{VR},\ \overline{\mathrm{V}}) \\ & & \\ \mathrm{NP} & \rightarrow & \langle \lambda, \mathtt{Marie} \rangle \mid \langle \lambda, \mathtt{koffie} \rangle \\ \mathrm{VT} & \rightarrow & \langle \lambda, \mathtt{drinken} \rangle \\ \mathrm{VR} & \rightarrow & \langle \lambda, \mathtt{zag} \rangle \end{array}$$

Figure 1: HG for transitive and raising verbs in Dutch.

The HG shown in figure 1 gives an analysis, similar to the one found in (Pollard 1984), of crossed dependencies in Dutch. The head of a $\overline{\mathrm{V}}$ is its leading verb, so the yield of a $\overline{\mathrm{V}}$ is a tuple whose first component is a sequence of NPs (the nominal cluster) and whose second component is a series of verbs (the verb cluster). The first of the noun phrases is the direct object of the head verb. The derivation tree for the verb phrase *Marie koffie zag drinken* (saw Mary drink coffee) is shown in figure 2. This HG analysis of the cross-serial verb phrase emphasizes elegantly both the underlying functional or deep structure of the verb phrase, and the way the surface form of a Dutch VP is constructed. It is therefore surprising to see that no attempts have been made to apply the very same method to other similar phenomena, such as verb second forms and leftward nominal extraposition.

The question *why* this has not been done is easily answered. (Pollard 1984) *needs* a SLASH feature to model leftward extraposition, because the head grammar formalism is too weak to treat both verbial and nominal discontinuity at once. The same is true for fronting of the head verb (Dutch verb second or English auxiliary inversion).

Suppose for example that want to extend the account of the VP to produce full Dutch sentential forms (verb second in (10b) and (10c) and topicalization

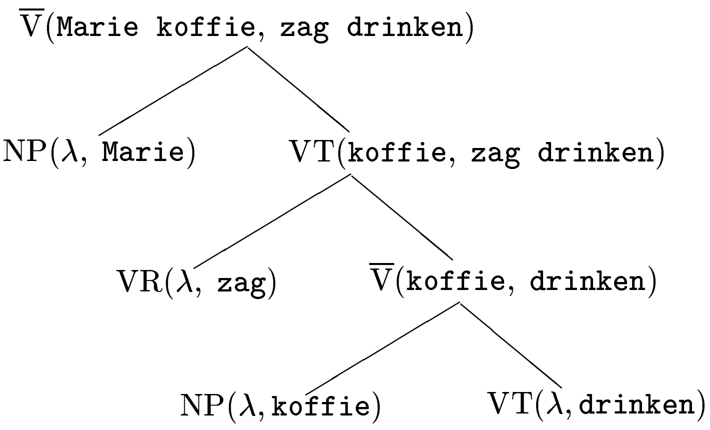

Figure 2: HG derivation.

of the first object in (10d)):

(10)

a. . . . dat Jan **Marie koffie zag drinken**
. . . that John saw Mary drink coffee

b. Jan **zag Marie koffie drinken**
John saw Mary drink coffee

c. **Zag** Jan **Marie koffie drinken**?
Did John see Mary drink coffee

d. **Wie zag** Jan **koffie drinken**?
Who did John see drink coffee?

If we want these four examples to get the same VP analysis, we see that we need to split the VP up into at least *three* components, i.e. the finite verb *zag*, its direct object *wie/Marie* and the remaining frame *koffie drinken*, in order to produce the surface forms through concatenation and wrapping.

So an extension of head grammar which splits up the yield of a constituent into more than two parts seems to provide the required power for describing more complex surface structure phenomena, at the expense of losing the linguistically flavoured motivation in terms of the role of heads. This is precisely what we will do in the next section.

4 Linear Context-Free Rewriting Systems

Linear context-free rewriting systems (LCFRS, Weir 1988) are a generalization of head grammar to arbitrary linear, non-erasing operations over tuples of arbitrary arity:

Definition. A *linear context-free rewriting system* (LCFRS) is a tuple

(N, T, μ, S, P) where N, T and S are as for HG; $\mu : N \rightarrow \mathbf{N}$ is the *similarity type* that assigns an arity to each nonterminal, $\mu(S) = 1$ and P is a set of *productions* of the form

$$A \quad \rightarrow \quad f(B_1, \ldots, B_m)$$

where $m \geq 0$, $A, B_1, \ldots, B_m \in N$, and the yield function f is a *linear, nonerasing* function over tuples of terminal words, that is, $f : ((T^*)^{\mu(B_1)}, \ldots, (T^*)^{\mu(B_m)}) \rightarrow (T^*)^{\mu(A)}$ can be defined symbolically as

$$f(\langle x_1^1, \ldots, x_{\mu(B_1)}^1 \rangle, \ \ldots, \ \langle x_1^m, \ldots, x_{\mu(B_m)}^m \rangle) \ = \ \langle t_1, \ldots, t_{\mu(A)} \rangle$$

where t_k are strings over terminals and the variables x_j^i, and each of the x_j^i appears precisely once in $t_1, \ldots, t_{\mu(A)}$.

LCFRS derivation is exactly analoguous to HG derivation as defined in the previous section, be it that we apply arbitrary linear and nonerasing operations to arbitrary tuples of terminal words.

Linear context-free rewriting systems are generally thought of as a generalization of the HG family in a formal language setting, and it is rarely found in the literature as a tool for describing natural language. First of all they are considered difficult to work with. Furthermore, by allowing arbitrary tuples and arbitrary operations, LCFRS have lost the very linguistically based status of HG, where the division of a constituent into two components is determined by the position of its head.

$$\begin{array}{lllll}
\mathrm{S} & \rightarrow & f_1(\mathrm{NP}, \overline{\mathrm{V}}) & \text{where} & f_1(p, \langle n,\ m,\ v,\ w \rangle) \ = \ \langle \mathtt{dat}\ p\ nm\ vw \rangle) \\
\mathrm{S} & \rightarrow & f_2(\mathrm{NP}, \overline{\mathrm{V}}) & \text{where} & f_2(p, \langle n,\ m,\ v,\ w \rangle) \ = \ \langle p\ v\ nm\ w \rangle) \\
\mathrm{S} & \rightarrow & f_3(\mathrm{NP}, \overline{\mathrm{V}}) & \text{where} & f_3(p, \langle n,\ m,\ v,\ w \rangle) \ = \ \langle v\ p\ nm\ w \rangle) \\
\mathrm{S} & \rightarrow & f_4(\mathrm{NP}, \overline{\mathrm{V}}) & \text{where} & f_4(p, \langle n,\ m,\ v,\ w \rangle) \ = \ \langle n\ v\ p\ m\ w \rangle) \\
 & & & & \\
\overline{\mathrm{V}} & \rightarrow & f_5(\mathrm{VT}, \mathrm{NP}) & \text{where} & f_5(t, p) \ = \ \langle p,\ \lambda,\ t,\ \lambda \rangle) \\
\overline{\mathrm{V}} & \rightarrow & f_6(\mathrm{VR}, \mathrm{NP}, \overline{\mathrm{V}}) & \text{where} & f_6(r, p, \langle n,\ m,\ v,\ w \rangle) \ = \ \langle p,\ nm,\ r,\ vw \rangle) \\
 & & & & \\
\mathrm{NP} & \rightarrow & \langle \mathtt{Jan} \rangle \mid \langle \mathtt{Marie} \rangle \mid \langle \mathtt{koffie} \rangle & & \\
\mathrm{VT} & \rightarrow & \langle \mathtt{drinken} \rangle & & \\
\mathrm{VR} & \rightarrow & \langle \mathtt{zag} \rangle & &
\end{array}$$

Figure 3: LCFRS for Dutch crossed dependencies, verb second and topicalization.

To illustrate why LCFRS is generally considered as a "difficult" grammar formalism, figure 3 shows[1] a very basic LCFRS which describes the Dutch cross-

[1] The grouping of the variable pairs nm and vw is suggestive notation without a formal status—it serves to stress that these couples, when taken together, represent the nominal cluster and verb cluster from the previously given HG grammar.

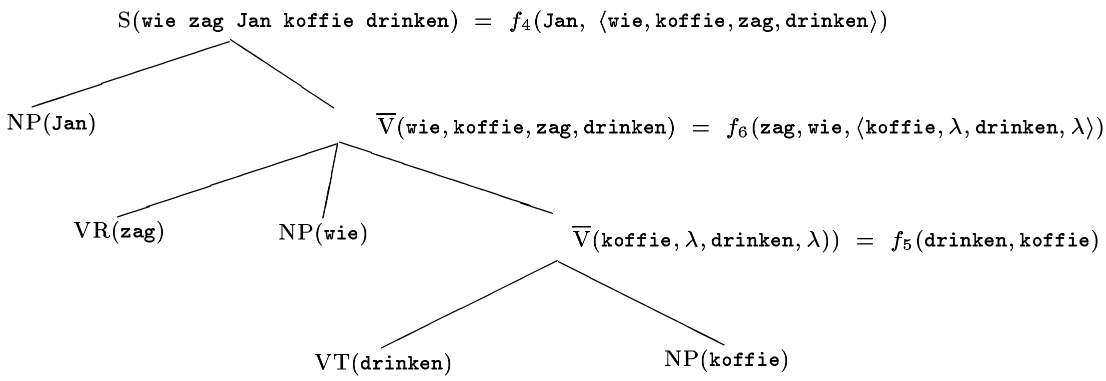

Figure 4: LCFRS derivation of *Wie zag Jan koffie drinken?*

serial VP and four sentential forms including verb second and topicalization. It is a straightforward extension of the HG from the previous section, in that it still divides the verb phrase into a *nominal cluster* and a *verb cluster*; but it splits up both clusters into two components. A verb phrase is now a four-tuple $\langle n, m, v, w\rangle$ consisting of a direct object n or the head of the nominal cluster, the rest of the nominal cluster m, the head verb v, and the rest of the verb cluster w. An example derivation is given in figure 4.[2]

Because an LCFRS production is divided into a "context-free production" and the definition of the yield function, it takes some time to understand a grammar; one has to identify which variables in the function definition are referring to which elements of the RHS of the context-free production. Nevertheless this paper will show, in three steps, how we can modify LCFRS so as to obtain a very attractive tool for the description of movement constructions. The first step is to eliminate the *yield functions* as elements of the grammar.

Step 1. Alternative definition. An LCFRS in *definite clause notation* is a tuple (N, T, V, μ, S, P) with N, T, μ and S as in the standard definition; V is a set of *variable symbols* disjoint with N and T, and the productions $R \in P$ are of the form

$$A(t_1, \ldots, t_{\mu(A)}) \;\texttt{:-}\; B_1(x_1^1, \ldots, x_{\mu(B_1)}^1),\ \ldots,\ B_m(x_1^m, \ldots, x_{\mu(B_m)}^m)$$

where t_i and x_j^i satisfy the same conditions as in the previous definition of LCFRS.[3]

[2] Note that the SOV structure from the HG analysis has been changed to SVO in the LCFRS example—preferable for reasons of uniformity as this is the underlying structure of English; the order of the elements on the RHS of an LCFRS production is irrelevant, so the distinction SVO/SOV in an LCFRS setting is no more than an issue of cosmetics.

[3] The use of the `:-` symbol in preference to $\rightarrow$ is motivated as follows: a Prolog clause can be inferred easily by replacing each term in a production with two integer indices. This corresponds to the construction which translates CPG as defined below into ILFP (Rounds 1988) in the proof that CPG has a fixed recognition problem in PTIME (Groenink 1995a).

The alternative definition provides, in a much more compact notation, the information of an LCFRS, except that the *names* of the yield functions have disappeared. Since the choice of yield functions in LCFRS does not seem to be so well motivated (it is unrestricted) as it is in head grammar, this can hardly be considered a disadvantage. The alternative notation *does* add an essential amount of readability to a grammar. In fact it looks considerably more like an annotated context-free grammar, be it that the order of the items on the RHS of a production is irrelevant. Figure 5 shows the alternative form of the LCFRS from figure 3.

$$
\begin{array}{lll}
\mathrm{S}(\mathtt{dat}\; p\; nm\; vw) & :- & \mathrm{NP}(p),\ \overline{\mathrm{V}}(n,\ m,\ v,\ w) \\
\mathrm{S}(p\; v\; nm\; w) & :- & \mathrm{NP}(p),\ \overline{\mathrm{V}}(n,\ m,\ v,\ w) \\
\mathrm{S}(v\; p\; nm\; w) & :- & \mathrm{NP}(p),\ \overline{\mathrm{V}}(n,\ m,\ v,\ w) \\
\mathrm{S}(n\; v\; p\; m\; w) & :- & \mathrm{NP}(p),\ \overline{\mathrm{V}}(n,\ m,\ v,\ w) \\
 & & \\
\overline{\mathrm{V}}(p,\ \lambda,\ t,\ \lambda) & :- & \mathrm{VT}(t),\ \mathrm{NP}(p) \\
\overline{\mathrm{V}}(p,\ nm,\ r,\ vw) & :- & \mathrm{VR}(r),\ \mathrm{NP}(p),\ \overline{\mathrm{V}}(n,\ m,\ v,\ w) \\
 & & \\
\mathrm{NP}(\mathtt{Jan}). & & \\
\mathrm{VT}(\mathtt{drinken}). & & \\
\ldots & &
\end{array}
$$

Figure 5: The LCFRS in definite clause notation.

The second step is the only step which elevates the generative capacity of the formalism (but not the complexity of fixed recognition, see (Groenink 1995a)). We will present it here without providing formal background. The more mathematically inclined reader is referred to (Groenink 1995c) in which the formal properties of CPG are discussed at length.

Step 2. Definition. A *simple concatenative predicate grammar* (CPG) is obtained by relaxing the definition of LCFRS as follows. For a production

$$A(t_1, \ldots, t_{\mu(A)}) :- B_1(x_1^1, \ldots, x_{\mu(B_1)}^1),\ \ldots,\ B_m(x_1^m, \ldots, x_{\mu(B_m)}^m)$$

we merely require that each of the variables x_j^i occurs *at least once* in $t_1, \ldots, t_{\mu(A)}$.

An informal characterization of a CPG is that it is an LCFRS in which the yield functions are replaced by arbitrary yield *relations*. It is shown in (Groenink 1995a) that the language recognized by an arbitrary given simple CPG[4] is recognisable in time polynomial in terms of the size of the input. Although CPG preserve polynomial time recognition, they are considerably stronger

[4]Note that this is a property, complexity of *fixed recognition*, of a class of languages rather than of a grammar formalism.

than LCFRS. They are e.g. capable of describing the Chinese number names of Radzinski (1991) (see Groenink (1995a)).

Here we will illustrate the added strength by an account of co-ordination. By adding the following rule, which is no longer a valid LCFRS rule because of the shared variable v on its right hand side:

$$\mathrm{S}(p\ v\ n_1 m_1\ w_1\ \texttt{en}\ n_2 m_2\ w_2)\ \ \texttt{:-}\ \ \mathrm{NP}(p),\ \overline{\mathrm{V}}(n_1, m_1, v, w_1),\ \overline{\mathrm{V}}(n_2, m_2, v, w_2)$$

to the grammar in figure 5, we can account for sentences such as

Jan zag Marie koffie drinken en Fred een koekje eten. (11)
saw coffee drink and a biscuit eat
John saw Mary drink coffee and (saw) Fred eat a biscuit.

5 Literal Movement Grammar

The third step in the extension of LCFRS is again one of notation. The grammars shown in the previous section leave a strong linguistic notion implicit: the notion of a strictly left-to-right concatenative *backbone* that is never subject to movement. In the case of our examples, the verb phrase contains one extremely "stable" cluster, *viz.* the final verbal cluster. In each of the S-productions, the verb cluster appears last in the full sentence.

We will now show that a formalism equivalent to simple CPG, the *simple literal movement grammars* (LMG) (Groenink 1995b), again give a considerable improvement in readability to the grammars under investigation, by separating a cluster to which the effect of the yield functions is restricted to simple left-to-right concatenation. A simple explanation of the effect of a translation to LMG is that as far as possible, the part of the sentence that is *not* subject to movement is treated in exactly the same way as it is in a context free grammar.

Step 3. Definition. A simple *literal movement grammar* is a tuple (N, T, V, μ, S, P), where N, T, V, μ and S are as for definite clause LCFRS;

- A *term* $t \in (T \cup V)^*$ is any sequence of terminals and variables.
- An *item* is one of the following:

  A terminal $\mathsf{a} \in T$

  A variable $x \in V$

  A simple predicate $A(x_1, \ldots, x_n)$ where $A \in N$ and $x_1, \ldots, x_n \in V$.

  A slashed predicate $(A(x_1, \ldots, x_n)/y)$ where $y \in V$.

- A production $R \in P$ is of the form

  $$A(t_1, \ldots, t_n)\ \rightarrow\ \Psi_1 \cdots \Psi_n$$

  where $A \in N$, $t_1, \ldots, t_n$ are terms and $\Psi_1, \ldots, \Psi_n$ are items, such that any variable occurring in a simple or slashed predicate Ψ_i occurs either as another item Ψ_j on the RHS, or in one of the terms t_i.

An *instantiated predicate* is a nonterminal A with $\mu(A)$ terminal arguments $A(w_1, \ldots, w_n)$ or a slashed form $A(w_1, \ldots, w_n)/v$. By substituting a terminal word w for each of the variables x in an LMG production we obtain an *instantiated production*

$$A(w_1, \ldots, w_n) \rightarrow \alpha$$

where α is a sequence of terminal symbols and instantiated predicates.

An LMG G recognizes a word w if $S \overset{G}{\Longrightarrow} w$ can be derived by the following inductive system:

Base case If $A(w_1, \ldots, w_n) \rightarrow \alpha$ is an instantiation of a production in G, then

$$A(w_1, \ldots, w_n) \overset{G}{\Longrightarrow} \alpha$$

Inductive steps

$$\frac{A(w_1, \ldots, w_n) \overset{G}{\Longrightarrow} \beta\, B(v_1, \ldots, v_m)\, \gamma \quad B(v_1, \ldots, v_m) \overset{G}{\Longrightarrow} u}{A(w_1, \ldots, w_n) \overset{G}{\Longrightarrow} \beta \;\, u \;\, \gamma}$$

$$\frac{A(w_1, \ldots, w_n) \overset{G}{\Longrightarrow} \beta \;\, (B(v_1, \ldots, v_m)/u) \;\, \gamma \quad B(v_1, \ldots, v_m) \overset{G}{\Longrightarrow} u}{A(w_1, \ldots, w_n) \overset{G}{\Longrightarrow} \beta \;\, \gamma}$$

While the formal definition of LMG is a bit clumsy, and it is easy to show that it is in fact merely a different notation for simple CPG (Groenink 1995c), grammars in the LMG system are generally elegant and understandable.

$$\begin{array}{lll}
\text{Rel} & \rightarrow & \texttt{dat}\ \text{NP}\ nm\ \ v\ \ \overline{\text{V}}(n, m, v) \\
\text{S} & \rightarrow & \text{NP}\ \ v\ \ nm\ \overline{\text{V}}(n, m, v) \\
\text{S} & \rightarrow & v\ \ \text{NP}\ \ nm\ \overline{\text{V}}(n, m, v) \\
\text{S} & \rightarrow & n\ \ v\ \ \text{NP}\ \ m\ \ \overline{\text{V}}(n, m, v) \\
 & & \\
\overline{\text{V}}(p,\ \lambda,\ t) & \rightarrow & (\text{VT}/t)\ \ (\text{NP}/p) \\
\overline{\text{V}}(p,\ nm,\ r) & \rightarrow & (\text{VR}/r)\ \ (\text{NP}/p)\ \ v\ \ \overline{\text{V}}(n, m, v) \\
 & & \\
\text{NP} & \rightarrow & \texttt{Jan} \\
\text{VT} & \rightarrow & \texttt{drinken} \\
\text{VR} & \rightarrow & \texttt{zag} \\
\ldots & &
\end{array}$$

Figure 6: The definite clause LCFRS as a literal movement grammar.

The LMG in figure 6 is equivalent to the definite clause LCFRS from figure 5. A $\overline{\text{V}}$ in the new grammar takes one argument less—the verb cluster which we have already argued is really part of the *surface backbone* of the sentence, is now produced as the *yield* of the $\overline{\mathrm{V}}$. In other words, a predicate $\overline{\mathrm{V}}(n, m, v)$ will recognize a $\overline{\mathrm{V}}$ constituent *missing* the clusters n, m and v. These arguments are terminal strings which, in the informal justification of the grammars we write, *select for (optional) extraposition.*

6 An LMG Account of Dutch Verb Structure

The grammars discussed so far treat transitive and raising verbs, S-relative, declarative and interrogative sentences, and topicalization of the first object. The following two LMG fragments extend the grammar from the previous sections with accounts of a number of fairly sophisticated forms of Dutch verb order.

Fragment 1

As in English, not only the first object in the VP can be topicalized, as in (12), but also any other.

Wie zag Jan koffie drinken? (12)
who saw
Who does John see drink coffee?

Wat zag Jan Marie drinken? (13)
what
What does John see Mary drink?

Furthermore, not all verb phrases have an object at all, and so far there was only a single bar level $\overline{\mathrm{V}}$. Introduce a category VP as follows:

VP(n, v) produces a verb phrase missing the clusters

n : empty or a single noun phrase that strictly selects for topicalized position.

v : empty or a single finite verb that strictly selects for first or second position.

The following sentential productions can now be stated.[5]

(s-rel)	S	$\rightarrow$	`dat` NP VP(λ, λ)
(s-decl)	S	$\rightarrow$	NP v VP(λ, v)
(s-inter)	S	$\rightarrow$	v NP VP(λ, v)
(s-topic)	S	$\rightarrow$	n v NP VP(n, v)

[5]These productions use a construction which falls outside the scope of simple LMG: they use an empty string in predicates on the right hand side of a production. However, this extension is obtained for free: a production A $\rightarrow$ B(λ) can be replaced by the two simple LMG productions A $\rightarrow x$ Empty(x) B(x) and Empty$(\lambda) \rightarrow \lambda$.

The verb phrase immediately dominates one or more $\overline{\mathrm{V}}$'s, which are roughly as in the previous section:

> $\overline{\mathrm{V}}(n,m,v)$ produces a level one verbal projection missing the clusters
>
> n : empty or a single noun phrase that strictly selects for topicalized position.
>
> m : a series of objects that selects to be placed within the VP but left of the yield of the $\overline{\mathrm{V}}$.
>
> v : the heading verb

This definition allows us to incorporate a limited account of binary conjunction at VP level:

(vp-v2)	$\mathrm{VP}(n,v)$	$\rightarrow$	$m\ \overline{\mathrm{V}}(n,m,v)$
(vp-fin)	$\mathrm{VP}(n,\lambda)$	$\rightarrow$	$m\ v\ \overline{\mathrm{V}}(n,m,v)$
(vp-v2-conj)	$\mathrm{VP}(n,v)$	$\rightarrow$	$m_1\ \overline{\mathrm{V}}(n,m_1,v)$ `en` $m_2\ \overline{\mathrm{V}}(n,m_2,v)$
(vp-fin-conj)	$\mathrm{VP}(n,\lambda)$	$\rightarrow$	$m_1\ v_1\ \overline{\mathrm{V}}(n,m_1,v_1)$ `en` $m_2\ v_2\ \overline{\mathrm{V}}(n,m_2,v_2)$

The $\overline{\mathrm{V}}$ is produced by the following rules:

(vi)	$\overline{\mathrm{V}}(\lambda,\lambda,v)$	$\rightarrow$	(VI/v)
(vt)	$\overline{\mathrm{V}}(\lambda,n,v)$	$\rightarrow$	$(\mathrm{VT}/v)\ (\mathrm{NP}/n)$
(vt-top)	$\overline{\mathrm{V}}(n,\lambda,v)$	$\rightarrow$	$(\mathrm{VT}/v)\ (\mathrm{NP}/n)$
(aux)	$\overline{\mathrm{V}}(n,m,v)$	$\rightarrow$	$(\mathrm{Aux}/v)\ w\ \overline{\mathrm{V}}(n,m,w)$
(vr)	$\overline{\mathrm{V}}(p,nm,v)$	$\rightarrow$	$(\mathrm{VR}/v)\ (\mathrm{NP}/n)\ w\ \overline{\mathrm{V}}(p,m,w)$
(vr-top)	$\overline{\mathrm{V}}(n,m,v)$	$\rightarrow$	$(\mathrm{VR}/v)\ (\mathrm{NP}/n)\ w\ \overline{\mathrm{V}}(\lambda,m,w)$

In all rules, the newly introduced verb becomes the head verb cluster v. Note that only the recursive productions (aux1), (vr1) and (vr1top) yield a nonempty string, that is they instantiate the head verb cluster w of the daughter $\overline{\mathrm{V}}$. For the verb types which introduce an object (VT and VR), there are separate rules for topicalization; (vr) takes an arbitrary daughter $\overline{\mathrm{V}}$ and carries over its topicalized object p, and (vr-top) takes a $\overline{\mathrm{V}}$ which does not select for topicalization (its topic cluster is empty), and puts the object n of the VR in the topic cluster. Figure 7 shows the derivation of *Jan zag Marie Fred leren zwemmen* (John saw Mary teach Fred to swim).

Although the most frequently occurring verb order in Dutch is that of crossed dependencies, there are some exceptions. An *extraposition verb* (VE), like

$$\mathrm{S} \Rightarrow \texttt{Jan zag Marie Fred leren zwemmen}$$
$$\mathrm{NP} \Rightarrow \texttt{J.} \quad \mathrm{VP}(\lambda, \texttt{zag}) \Rightarrow \texttt{Marie Fred leren zwemmen}$$
$$\overline{\mathrm{V}}(\lambda, \texttt{Marie Fred}, \texttt{zag}) \Rightarrow \texttt{leren zwemmen}$$
$$\mathrm{VR} \Rightarrow \texttt{zag} \quad \mathrm{NP} \Rightarrow \texttt{M.} \quad \overline{\mathrm{V}}(\lambda, \texttt{Fred}, \texttt{leren}) \Rightarrow \texttt{zwemmen}$$
$$\mathrm{VR} \Rightarrow \texttt{leren} \quad \mathrm{NP} \Rightarrow \texttt{F.} \quad \overline{\mathrm{V}}(\lambda, \lambda, \texttt{zwemmen}) \Rightarrow \lambda$$
$$\mathrm{VI} \Rightarrow \texttt{zwemmen}$$

Figure 7: LMG derivation in Fragment 1.

verbieden, selects for a full VP complement to appear to its right:

a. ...dat de dokter [$_{\mathrm{VP}}$ Jan verbiedt [$_{\mathrm{VP}}$ Anne te bezoeken]] (14)
disallows to visit
...that the doctor does not allow John to visit Anne
b. ?...dat de dokter Jan Anne verbiedt te bezoeken

Although the full VP appears right of the extraposition verb *verbiedt*, objects from the daughter VP can still be topicalized:

Wat verbiedt de dokter [$_{\mathrm{VP}}$ Jan [$_{\mathrm{VP}}$ **Anne te geven**?]] (15)
What disallows to give
What does the doctor not allow John to give Anne?

This leads to the following productions for VE:

(ve)	$\overline{\mathrm{V}}(p, n, v)$	$\rightarrow$	$(\mathrm{VE}/v)\ (\mathrm{NP}/n)\ \mathrm{VP}^{+\mathrm{te}}(p, \lambda)$
(ve-top)	$\overline{\mathrm{V}}(n, \lambda, v)$	$\rightarrow$	$(\mathrm{VE}/v)\ (\mathrm{NP}/n)\ \mathrm{VP}^{+\mathrm{te}}(\lambda, \lambda)$

$$\mathrm{S} \Rightarrow \texttt{dat de dokter Jan verbiedt Anne te bezoeken}$$
$$\mathrm{NP} \Rightarrow \texttt{de dokter} \quad \mathrm{VP}(\lambda, \lambda) \Rightarrow \texttt{Jan verbiedt Anne te bezoeken}$$
$$\overline{\mathrm{V}}(\lambda, \texttt{Jan}, \texttt{verbiedt}) \Rightarrow \texttt{Anne te bezoeken}$$
$$\mathrm{VE} \Rightarrow \texttt{verbiedt} \quad \mathrm{NP} \Rightarrow \texttt{Jan} \quad \mathrm{VP}^{+\mathrm{te}}(\lambda, \lambda) \Rightarrow \texttt{Anne te bezoeken}$$
$$\overline{\mathrm{V}}(\lambda, \texttt{Anne}, \texttt{bezoeken}) \Rightarrow \lambda$$
$$\mathrm{VT} \Rightarrow \texttt{bezoeken} \quad \mathrm{NP} \Rightarrow \texttt{Anne}$$

Figure 8: Derivation in fragment 1 of sentence (14a).

Fragment 2

To conclude the illustration, the next fragment splits up the nominal cluster into two parts, obtaining both a limited description of *partial extraposition verbs* and some forms of partial ellipsis in cases of co-ordination.

A partial extraposition verb such as *proberen* is a liberal-minded verb that accepts anything between a full VP and cross-serial order:

a. ...dat Jan probeert [$_{\mathrm{VP}}$ **Marie koffie te geven**] (16)
tries coffee to give
b. ...dat Jan **Marie** probeert [$_{\overline{\mathrm{V}}}$ **koffie te geven**]
c. ...dat Jan **Marie koffie** probeert [$_{\overline{\mathrm{V}}}$ **te geven**]

Example a. may be solved by taking the rules for full extraposition (VE) verbs, but b. and c. suggest that the nominal cluster *Marie koffie* of the $\overline{\mathrm{V}}$ can be split up into at least two parts: one preceding and one following the PE verb. However, there is no indication that it cannot be scrambled into an arbitrary number of parts, in which case the LMG formalism would be intrinsically unable to describe partial extraposition. The following "data" suggest that nesting of partial extraposition verbs is very restricted:

(17)

a. dat Jan Marie meent te hebben zien [$_{\overline{\mathrm{V}}}$ proberen **Anne koffie** te **laten drinken**]
believes to have seen try make drink
b. ?dat Jan Marie meent **Anne** te hebben zien [$_{\overline{\mathrm{V}}}$ proberen **koffie** te **laten drinken**]
c. ?dat Jan Marie meent **Anne koffie** te hebben zien [$_{\overline{\mathrm{V}}}$ proberen te **laten drinken**]
d. ?dat Jan Marie **Anne** meent te hebben zien [$_{\overline{\mathrm{V}}}$ proberen **koffie** te **laten drinken**]
e. *dat Jan Marie **Anne** meent **koffie** te hebben zien [$_{\overline{\mathrm{V}}}$ proberen te **laten drinken**]
f. dat Jan Marie **Anne koffie** meent te hebben zien [$_{\overline{\mathrm{V}}}$ proberen te **laten drinken**]
that John believes to have seen Mary try to make Anne drink coffee

Cases (17b) and (17c) correspond to (16b), which is already disliked by many Dutch speakers, but cases (17d) and (17e) are very questionable. In LMG there seems to be no choice but to require the objects that move leftward over *proberen* to appear as an unaltered sequence in the resulting sentence, allowing all forms except (17e).

The second fragment replaces the nominal cluster by two parts, only one of which will be allowed to move leftward over partial extraposition verbs. The grammar will produce all possible divisions of the objects into two parts.

$\overline{\mathrm{V}}(n, m_1, m_2, v)$ produces a level one verbal projection missing the clusters

n : empty or a single noun phrase that strictly selects for topicalized position.

m_1 : a nominal cluster that may cross partial extraposition verbs, or when consisting of a single NP, may move to topicalized position.

m_2 : a nominal cluster that will appear at the rightmost possible position left of the verb that immediately dominates it.

v : the heading verb

The $\overline{\mathrm{V}}$ rules given so far are modified as follows:

$$\begin{array}{lll}
(\mathrm{vi})' & \overline{\mathrm{V}}(\lambda,\lambda,\lambda,v) & \rightarrow (\mathrm{VI}/v) \\
(\mathrm{vt})' & \overline{\mathrm{V}}(\lambda,n,\lambda,v) & \rightarrow (\mathrm{VT}/v)\ \ (\mathrm{NP}/n) \\
(\mathrm{aux})' & \overline{\mathrm{V}}(n,m_1,m_2,v) & \rightarrow (\mathrm{Aux}/v)\ \ w\ \ \overline{\mathrm{V}}(n,m_1,m_2,w) \\
(\mathrm{vr})' & \overline{\mathrm{V}}(p,nm_1,m_2,v) & \rightarrow (\mathrm{VR}/v)\ \ (\mathrm{NP}/n)\ \ w\ \ \overline{\mathrm{V}}(p,m_1,m_2,w) \\
(\mathrm{ve})' & \overline{\mathrm{V}}(p,n,\lambda,v) & \rightarrow (\mathrm{VE}/v)\ \ (\mathrm{NP}/n)\ \ \mathrm{VP}^{+\mathrm{te}}(p,\lambda)
\end{array}$$

Note that the double rules for introducing topics have disappeared. Rules that introduce a new object n add that object to the first nominal cluster. If the first nominal cluster m_1 consists of just one NP, it may move to the second nominal cluster (the `shift` rule) or to topicalized position (`topic` rule).[6]

$$\begin{array}{lll}
(\mathrm{shift}) & \overline{\mathrm{V}}(p,\lambda,m_1m_2,v) & \rightarrow\ \ (\mathrm{NP}/m_1)\ \ \overline{\mathrm{V}}(p,m_1,m_2,v) \\
(\mathrm{topic}) & \overline{\mathrm{V}}(m_1,\lambda,m_2,v) & \rightarrow\ \ (\mathrm{NP}/m_1)\ \ \overline{\mathrm{V}}(\lambda,m_1,m_2,v)
\end{array}$$

Partial extraposition verbs can now be accounted for as follows:

$$(\mathrm{pe})\quad \overline{\mathrm{V}}(n,m_1,\lambda,v)\ \rightarrow\ (\mathrm{PE}/v)\ \ m_2\ \ \mathtt{te}\ \ w\ \ \overline{\mathrm{V}}(n,m_1,m_2,w)$$

The first nominal cluster m_1 of the daughter $\overline{\mathrm{V}}$ is percolated upward as the left nominal cluster of the mother, and is hence allowed to skip PE verbs higher up, but can no more be broken into two parts, as suggested above. The second nominal cluster m_2 is yielded immediately after the PE verb.

The following VP rules conclude the fragment.

$$\begin{array}{ll}
(\mathrm{vp\text{-}v2})' & \mathrm{VP}(n,v)\rightarrow m_1\ m_2\ \overline{\mathrm{V}}(n,m_1,m_2,v) \\
(\mathrm{vp\text{-}fin})' & \mathrm{VP}(n,\lambda)\rightarrow m_1\ m_2\ v\ \overline{\mathrm{V}}(n,m_1,m_2,v) \\
(\mathrm{vp\text{-}v2\text{-}conj})' & \mathrm{VP}(n,v)\rightarrow m_1\ m_2\ \overline{\mathrm{V}}(n,m_1,m_2,v)\ \mathtt{en}\ m_3\ \overline{\mathrm{V}}(n,m_1,m_3,v) \\
(\mathrm{vp\text{-}fin\text{-}conj})' & \mathrm{VP}(n,\lambda)\rightarrow m_1\ m_2\ v_1\ \overline{\mathrm{V}}(n,m_1,m_2,v_1)\ \mathtt{en}\ m_3\ v_2\ \overline{\mathrm{V}}(n,m_1,m_3,v_2)
\end{array}$$

The conjunction rules now have a larger coverage than in the first fragment as they allow an initial part of the nominal cluster to be shared between the conjuncts, producing sentences like (18):

Jan zag Marie koffie drinken en een koekje eten. (18)
saw coffee drink and a biscuit eat
John saw Mary drink coffee and (saw Mary) eat a biscuit.

---

[6]The `topic` and `shift` rules enforce, by using a slash item, that the shifted subcluster n consists of just one NP. Here we assume the non-existence of funny NP sequences such as *Anne Frank*, which can be read either as one NP or as two NPs. This would lead to problematic sentences such as *Anne Frank zag ik gisteren kussen.*

7 Conclusion

This paper has sketched an approach to the description of languages with a highly complex surface structure, together with a grammar for a fragment of Dutch to illustrate its strength and ease of use. Particular to this approach w.r.t. other frameworks in both descriptive computational linguistics and formal language theory is that at the same time (1) a variety of surface order phenomena is actually shown to be feasibly described in a single grammar, and (2) the formalism used has been shown to be tractable. Although simple LMG has been shown (Groenink 1995a) to describe *precisely* the class of tractable languages, this of course does not imply that the formalism allows such languages to be defined in a straightforward way.

The fragment sketched in this paper is still limited (e.g., it only partially describes what is covered in the categorial account of (Bouma and Van Noord 1995), on which the fragments discussed are largely based), and it remains interesting to investigate whether the formalism is able to adequately describe larger fragments in a straightforward way. Some readers may find the second literal movement grammar in section 6, which splits up the nominal cluster into two parts, theoretically undefendable, while the verb order phenomena which remain to be described, such as the inverted verb order (Bouma and Van Noord 1995) in

...dat Jan het boek gelezen moet hebben (19)
the book read must have
that John must have read the book

suggest that the same may have to be done to the verb cluster.

Acknowledgements

The coverage of the fragments in section 6, and the descriptions of Dutch in this paper are almost universally borrowed from (Bouma and Van Noord 1995). The author is supported by SION grant 612-317-420 of the Netherlands Organization for Scientific Research (NWO).

References

Bouma, G., and Van Noord, G. (1995). A lexicalist account of the Dutch verbal complex. In *Papers from the Fourth CLIN Meeting, Rijksuniversiteit Groningen.*

Groenink, A. V. (1995a). An elegant grammatical formalism for the polynomial-time recognisable languages. Paper presented at the fourth Mathematics of Language workshop (MOL4), Univ. of Pennsylvania.

Groenink, A. V. (1995b). Literal movement grammars. In *Proceedings of the 7th EACL Conference, University College, Dublin.*

Groenink, A. V. (1995c). A unifying framework for concatenation-based grammar formalisms. In *Proceedings of Accolade 1995, Amsterdam (to appear).*

Kaji, Y., Nakanishi, R., Seki, H., and Kasami, T. (1992). The universal recognition problems for parallel multiple context-free grammars and for their subclasses. *IEICE*, E75–D(4):499–508.

Keller, B., and Weir, D. (1995). A tractable extension of linear indexed grammars. In *Proceedings of the 7th EACL Conference, University College, Dublin.*

Pereira, F. (1981). Extraposition grammars. *Computational Linguistics*, 7(4):243–256.

Pollard, C. J. (1984). *Generalized Phrase Structure Grammars, Head Grammars, and Natural Language.* PhD thesis, Standford University.

Radzinski, D. (1991). Chinese number-names, tree adjoining languages, and mild context-sensitivity. *Computational Linguistics*, 17(3):277–299.

Rounds, W. C. (1988). Lfp: A logic for linguistic descriptions and an analysis of its complexity. *Computational Linguistics*, 14(4):1–9.

Vijay-Shanker, K., and Weir, D. J. (1994). The equivalence of four extensions of context-free grammar. *Math. Systems Theory*, 27:511–546.

Weir, D. J. (1988). *Characterizing Mildly Context-Sensitive Grammar Formalisms.* PhD thesis, University of Pennsylvania.