Tractability issues in extraposition grammar

A.V. Groenink

Computer Science/Department of Interactive Systems

CS-R9635 1996

Report CS-R9635
ISSN 0169-118X

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Tractability issues in Extraposition Grammar

Annius V. Groenink

CWI

P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

avg@cwi.nl

September 5, 1996

Abstract

Extraposition Grammar (XG) was introduced in [Per81] as a grammar formalism whose increase in recognizing power over context free grammars is limited to mechanisms for adequate description of structural phenomena occurring in natural language.

This paper proposes versions of XG whose fixed recognition problems are in deterministic polynomial and exponential time; furthermore it is shown that the unrestricted XG defined in the original work [Per81] describe any recursively enumerable language.

AMS Subject Classification (1991): 03D10, 68Q15, 68Q45, 68Q50, 68Q52, 68S05

CR Subject Classification (1991): D3.1, F1.2, F1.3, F4.2, F4.3

Keywords & Phrases: Grammar formalisms; complexity of language recognition; weak generative capacity

Note: The author is supported by SION grant 612-317-420 of the Netherlands Organization for Scientific Research (NWO). The paper is a preliminary version of material to appear in a larger document on various applications of simple literal movement grammar (LMG).

1. Introduction

Extraposition Grammar (XG), introduced in [Per81], is a grammar formalism that augments context-free grammar with a dedicated construction modelling *leftward extraposition.* Pereira writes that left extraposition is a widely used model for describing interrogative sentences and relative clauses in most Western European languages, and these constructions are so essential, even in small real-world applications, that we would like to be able "to express them in a clear and concise manner".

I will show here that XG in their original form from [Per81] describe any recursively enumerable language, which implies that even though the extraposition construction in XG may be "clear and concise", it can't be considered 'mimimal' in expressive power (many movement constructions can be expressed formalisms like LMG [Gro96] that only generate languages recognizable in polynomial time). However, I will also propose variants of XG whose fixed recognition problems are in deterministic polynomial and exponential time, and whose descriptive properties are still favourable.

The following definition gives a slightly simplified[1] characterization of the original form of XG as in [Per81].

[1] In the XG described in [Per81], the rules of the second type are of the more general form $A_1 \ldots A_2 \cdots\cdots\cdots A_n \to X_1 X_2 \cdots X_n$. It is not hard to prove directly that the 'bilinear' version we discuss here is weakly equivalent to Pereira's definition; however, we get the result for free in this paper once we've proved that the bilinear version describes all r.e. languages.

$$\begin{array}{lll} (1) & \text{VP} & \rightarrow \text{NC VC} \\ (2) & \text{NC} & \rightarrow \lambda \\ (3) & \text{NC} \ldots \text{Trace} & \rightarrow \text{NP NC} \\ (4) & \text{VC} & \rightarrow \text{VT Trace} \\ (5) & \text{VC} & \rightarrow \text{VR Trace VC} \end{array}$$

Figure 1: XG for Dutch verb phrases.

$$\begin{array}{rlll} \text{VP} & \overset{G}{\Longrightarrow} & \text{NC VC} & (1) \\ & \overset{G}{\Longrightarrow} & \text{NC VR Trace VC} & (5) \\ & \overset{G}{\Longrightarrow} & \text{NC VR Trace VT Trace} & (4) \\ & \overset{G}{\Longrightarrow} & \text{NP NC [VR] VT Trace} & (3) \\ & \overset{G}{\Longrightarrow} & \text{NP NP NC [[VR] VT]} & (3) \\ & \overset{G}{\Longrightarrow} & \text{NP NP [[VR] VT]} & (2) \\ & \overset{G}{\Longrightarrow}{}^{*} & \texttt{Marie koffie [[zag] drinken]} & \end{array}$$

Figure 2: XG Derivation

Definition 1.1 An *extraposition grammar* (XG) is a tuple (N, T, S, P) where N and T are disjoint sets of nonterminal and terminal symbols, $S \in N$ is the start symbol, and P is a finite set of productions of the forms

1. $A \rightarrow X_1 X_2 \cdots X_n$
2. $A \ldots B \rightarrow X_1 X_2 \cdots X_n$

where $A, B \in N$ and $X_i \in (N \cup T)$.

Derivation in an XG is defined over *bracketed* sequences $\overline{\alpha} \in (N \cup T \cup \{[,]\})^*$ of nonterminal and terminal symbols; let $\overline{\alpha}, \overline{\beta}, \overline{\gamma}$ be such sequences, and let a rule of type 1, 2 be in P, then we have, respectively:

1. $\overline{\alpha} A \overline{\beta} \Rightarrow \overline{\alpha} X_1 X_2 \cdots X_n \overline{\beta}$
2. $\overline{\alpha} A \overline{\gamma} B \overline{\beta} \Rightarrow \overline{\alpha} X_1 X_2 \cdots X_n [\overline{\gamma}] \overline{\beta}$

provided that the brackets in $\overline{\gamma}$ are *balanced.* Now G derives a string w if there is a bracketing $\overline{w}$ of w such that $S \overset{*}{\Rightarrow} \overline{w}$.

We will use the following notation: A, B, C are nonterminals, $\mathsf{a}, \mathsf{b}, \mathsf{c}$ are terminals, u, v, w stand for terminal words, and $\overline{u}, \overline{v}, \overline{w}$ are (not necessarily balanced) bracketings of u, v and w, and $\alpha, \beta, \gamma, \ldots$ are sequences of terminal and nonterminal symbols, and $\overline{\alpha}, \overline{\beta}, \overline{\gamma}$ are bracketed sequences.

Figure 1 shows an example grammar that gives a description of Dutch cross-serial verb phrases, as in sentence (1.1). It divides the verb phrase VP into a nominal cluster NC and a verb cluster VC. A

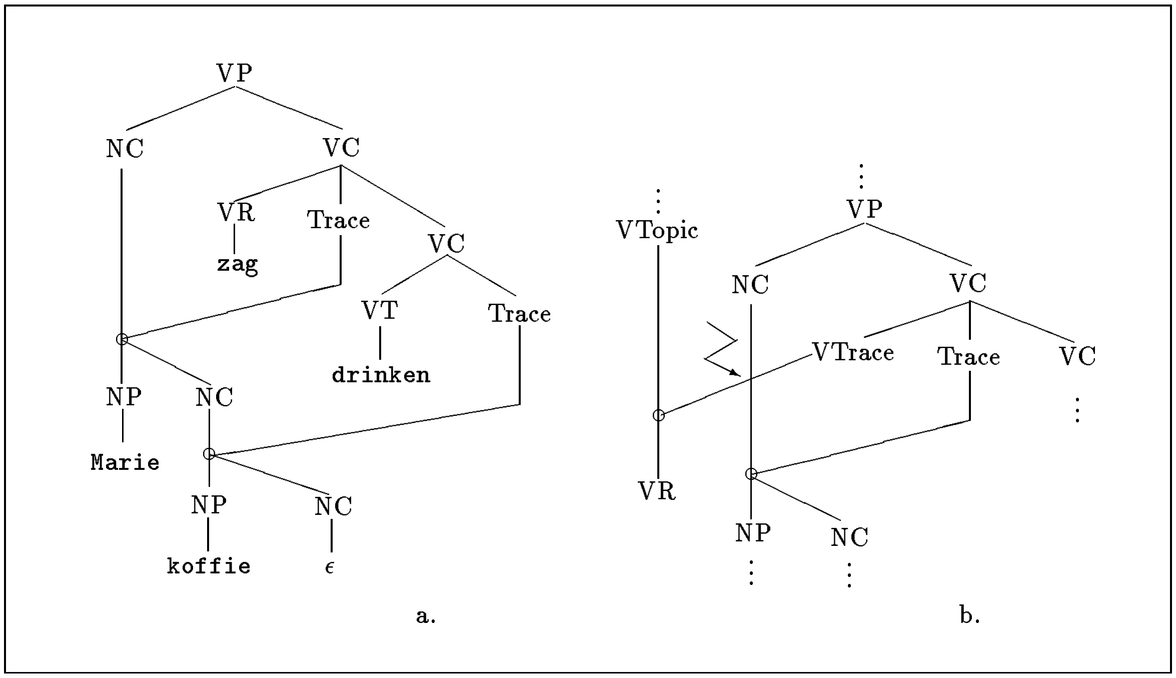

Figure 3: XG derivation graphs

nominal cluster splits off a noun phrase whenever there is a trace to be eliminated to its right. The derivation of the verb phrase *Marie koffie zag drinken* is shown in figure 2, and the corresponding derivation graph in figure 3a.

(1.1) ...dat Jan $[_{\mathrm{VP}}$ Marie$_i$ koffie$_j$ zag e_i drinken e_j].
(...that Jan saw Mary drink coffee)

The derivation graph shows more intuitively how the ellipsis rules establish a link between a fronted NP and its trace. A sequence is between brackets in the textual derivation, whenever it is enclosed in a *cycle* in the derivation graph. In terms of derivation graphs, the balancedness constraint in the second XG rewrite rule ensures that the graphs have a top-down, left-to-right ordered, planar representation; in other words: the lines in the graph, the way they are drawn in figure 3, do not cross.

(1.2)
$$\begin{aligned} \mathrm{VP} &\overset{G}{\Longrightarrow}{}^* \ \mathrm{NC\ \ VR\ \ Trace\ \ VT\ \ Trace} \\ &\overset{G}{\Longrightarrow}{}^* \ \mathrm{NP\ \ NC\,[VR\ \ Trace\ \ VT]} \end{aligned}$$

Let's try to swap the dependencies between the NPs and the traces, as in (1.2). We see that the balancedness constraint makes that we are at a dead end in the derivation, because an unbound trace is enclosed in brackets (or in other words: an island). Hence the incorrect analysis (1.3) is ruled out.

(1.3) *Marie$_j$ koffie$_i$ zag e_i drinken e_j*

The corresponding tentative derivation graph would have crossing lines between the two NPs and their traces. The beauty of this approach is that it gives a model of *extraposition* without a hint of *transformation* of structures. There is one clearly defined sentential structure which has elements of both deep structure and surface structure.

(1)	Rel	$\rightarrow$	**dat** NP VP
(2)	S	$\rightarrow$	NPtopic Vtopic NP VP
(3)	S	$\rightarrow$	NP Vtopic NP VP
(4)	S	$\rightarrow$	Vtopic NP VP
(5)	NPtopic . . . TopicTrace	$\rightarrow$	NP
(6)	Vtopic . . . VTtrace	$\rightarrow$	VT
(7)	Vtopic . . . VRtrace	$\rightarrow$	VR
(8)	VP	$\rightarrow$	$\overline{\text{NC}}$ VC
(9)	$\overline{\text{NC}}$	$\rightarrow$	NC
(10)	$\overline{\text{NC}}$ . . . VTtrace	$\rightarrow$	VTtrace NC
(11)	$\overline{\text{NC}}$ . . . VRtrace	$\rightarrow$	VRtrace NC
(12)	NC . . . NPtrace	$\rightarrow$	NP NC
(13)	NC . . . NPtrace	$\rightarrow$	TopicTrace NC
(14)	NC	$\rightarrow$	ϵ
(15)	VC	$\rightarrow$	VT NPtrace
(16)	VC	$\rightarrow$	VTtrace NPtrace
(17)	VC	$\rightarrow$	VR NPtrace VC
(18)	VC	$\rightarrow$	VRtrace NPtrace VC

Figure 4: Pit-stopping XG for examples (2.4) and (2.5).

2. Multiple threads of movement

It makes sense to try and extend the XG account of verb phrases to cover placement of the inflected verb in Dutch, using the same mechanism. Look, for example, at the interrogative sentence (2.4).

(2.4) *Zag Jan Marie koffie drinken?*
(Did Jan see Mary drink coffee?)

Here *zag* can be thought of as moving leftward out of the verb cluster, crossing over the noun cluster, which itself may be linked to traces right of the verb trace. This would violate the bracketing constraint, illustrated by the crossing lines in figure 3b. A more complex example (2.5) introduces topicalization, where one NP moves forward over the extraposed finite verb.

(2.5) *Wie zag jij koffie drinken?*
(Who did you see drink coffee?)

There are two ways of getting around this problem. The first, discussed in this section, is writing a (excessively) complex grammar that simulates the crossing lines without actually having them cross. Another is to modify the semantics of XG so as to allow for lines belonging to different filler–trace pairs to cross. It will turn out that relaxing the constraints on bracketing in XG derivations, allowing for parallel threads of extraposition, is also the key to obtaining a tractable subclass of XG.

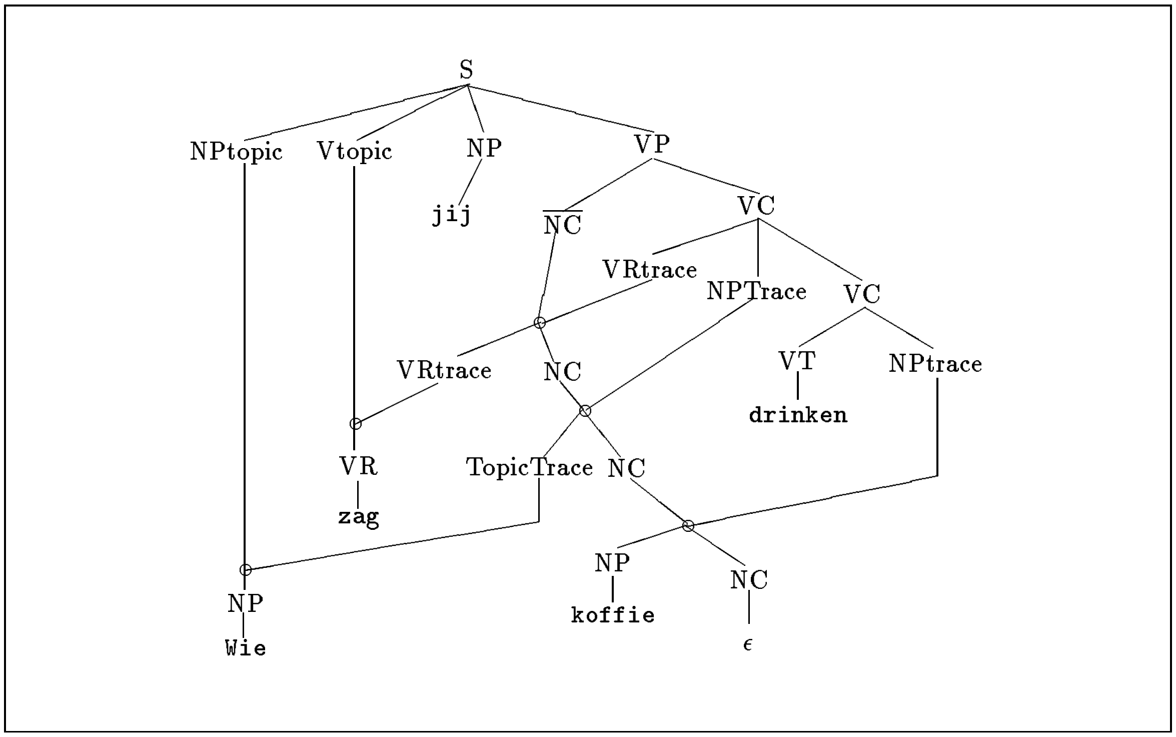

Figure 5: XG derivation of *Wie zag Jan koffie drinken?*

The grammar shown in figure 4 gives a cumbersome analysis of the examples, especially in terms of the grammar which has 'duplicates' of similar rules (4 rules for each subcategorization type). But the resulting graph has a favourable structure and indicates that an analysis that provides a proper mechanism for allowing lines to cross would assign 'nice' structures to the Dutch sentences.

The technique used in the grammar of figure 4 exploits, roughly, the idea of *pit-stopping* known from Government-Binding theory, allowing the optional verb traces to "step over" the brackets (rules (10, 11)) introduced in the derivation to prevent the lines connecting nominal filler–trace pairs to cross. Similarly, an NP-trace can be eliminated in the NC either by generating a real NP (rule 12) or by generating a topic-trace (rule 13). At most one nominal trace, and at most one verb trace are eliminated at S-level (rules 2–4). Because of the distinction between NC and $\overline{\text{NC}}$, only the first verb can be fronted; however any of the NPs is allowed to be topicalized.

3. Alternative forms of XG derivation

Thusfar no results are known about the decidability of extraposition grammar. In section 4 I will show that a decidable subclass can be defined by requiring that the derivations satisfy a boundedness property on the number of brackets. Before I give this result, a few simplifications to the semantics of XG need to be made.

Proposition 3.1 *The additional requirement in the interpretation of the ellipsis rule $A \ldots B \to C$ that $\overline{\gamma}$ contains* no nonterminal symbols, *does not change the semantics of an XG.*

Proof. Let d be the derivation of a terminal string from the start symbol S. Look at a step s in d where an ellipsis rule $A \ldots B \to C$ is used; this step is $\overline{\alpha}A\overline{\gamma}B\overline{\beta} \Rightarrow \overline{\alpha}C[\overline{\gamma}]\overline{\beta}$. Suppose $\overline{\gamma}$ contains nonterminal symbols. Then for each of these nonterminals, there will be a rule, applied *after* step s, to eliminate the nonterminal. Whatever such a rule is, it cannot depend on $\overline{\alpha}$ and $\overline{\beta}$ because $\overline{\gamma}$ is enclosed in brackets. Hence we could also have applied these rules *before* step s. Repeat the observation to obtain a derivation in which $\overline{\gamma}$ contains no nonterminal symbols. □

The modification in the semantics imposes a stronger restriction on the *order* of the XG derivations. Requiring $\overline{\beta}$ in the interpretation of the ellipsis rule to be terminal seems to be the same as requiring the derivation graphs to be *planar* as explained in Pereira's paper. Hence the following simplified definition of XG derivation.

Definition 3.2 (bracket-free interpretation) *Bracket-free derivation* in an XG is defined over unbracketed sequences $\alpha \in (N \cup T)^*$ of nonterminal and terminal symbols as in a context-free grammar. Let α and β be such sequences, let w be a terminal word and let a rule of type 1, 2 be in P, then we have, respectively:

1. $\alpha A \beta \Rightarrow \alpha X_1 X_2 \cdots X_n \beta$
2. $\alpha A w B \beta \Rightarrow \alpha X_1 X_2 \cdots X_n w \beta$

Proposition 3.3 *Bracket-free derivation is equivalent to the original definition in 1.1.*

Proof. One implication is obvious given proposition 3.1. As to the other: suppose there is a bracket-free derivation, then there is a derivation according to the original definition. This is also simple: let d be a bracket-free derivation, then let the bracketed derivation d' apply the same sequence of rules. Then d' is a correct derivation, because whenever it applies an ellipsis rule in a step $\overline{\alpha} A \overline{\beta} B \overline{\gamma} \rightarrow \overline{\alpha} C [\overline{\beta}] \overline{\gamma}$, $\overline{\beta}$ must be a bracketing of a terminal word. So it only puts brackets *around terminal words*; that is no nonterminals appear inside matching brackets. Since $\overline{\beta}$ is immediately next to the two nonterminals A and B, there cannot be any unmatched brackets in $\overline{\beta}$. □

I will use this modified definition to make the proofs further in this paper much easier; for now, we proceed immediately to another equivalent alternative interpretation.

Definition 3.4 (corresponding CFG) Let $G = (N, T, S, P_1 \cup P_2)$ be an extraposition grammar, where P_1 contains context-free rules only, and P_2 contains only ellipsis rules. Then the *corresponding context-free grammar* $\mathcal{CF}(G)$ is the context free grammar $(N, T \cup \{[_B,]_B \mid B \in N\}, S, P_1 \cup P_3)$ where for every rule $A \ldots B \rightarrow \alpha$ in P_1, we add to P_3 the two rules

$$\begin{array}{lccl} \text{(i)} & A & \rightarrow & \alpha\, [_B \\ \text{(ii)} & B & \rightarrow &]_B. \end{array}$$

Proposition 3.5 *Let G be an extraposition grammar. Define the rewriting operation $\rightsquigarrow$ over bracketed terminal strings as follows: for every $B \in N$,*

$$\overline{u}[_B v]_B \overline{w} \rightsquigarrow \overline{u} v \overline{w}$$

(where v contains no brackets).

Then G derives a sentence w iff $\mathcal{CF}(G)$ derives a $\overline{w}$ such that $\overline{w} \overset{*}{\rightsquigarrow} w$.

S	$\rightarrow$	AC
A	$\rightarrow$	λ
A...F	$\rightarrow$	B
B	$\rightarrow$	aA
C	$\rightarrow$	λ
C	$\rightarrow$	bD
D	$\rightarrow$	Ec
E	$\rightarrow$	FC

S	$\rightarrow$	AC
A	$\rightarrow$	λ
A	$\rightarrow$	B[
F	$\rightarrow$	]
B	$\rightarrow$	aA
C	$\rightarrow$	λ
C	$\rightarrow$	bD
D	$\rightarrow$	Ec
E	$\rightarrow$	FC

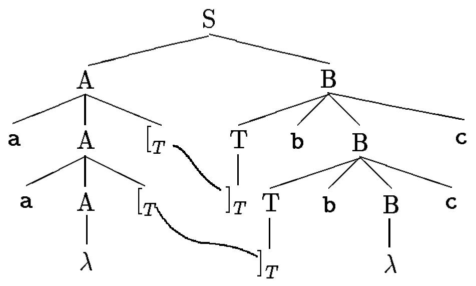

Figure 6: The corresponding context-free grammar for $a^n b^n c^n$ and a derivation graph.

Proof. The definition of $\rightsquigarrow$ is merely a formal characterization of the idea that G derives w if and only if $\mathcal{CF}(G)$ derives a *balanced* bracketing of w.

Clearly, if we have a G-derivation of w, then we also have a $\mathcal{CF}(G)$-derivation of a balanced bracketing $\overline{w}$ (the bracketing is identical to that in the XG derivation according to definition 1.1.

As to the converse, if there is a derivation of a balanced bracketing $\overline{w}$ in $\mathcal{CF}(G)$, then we must show that we can transform the context-free derivation such that for each pair of matching brackets, the rules $R_1 : \quad A \rightarrow \alpha\ [_B$ and $R_2 : \quad B \rightarrow\ ]_B$ applied to produce these brackets, follow eachother immediately. This then corresponds to applying the single XG rule $A \ldots B \rightarrow \alpha$.

It is straightforward to see that this can indeed be done. Order the bracket pairs rightmost–innermost, which is to say the *opening* brackets are ordered from right to left. Repeatedly take the next pair from this list, and write out the least number of rules needed to introduce the brackets. We must show that to do this, no other new brackets need to be introduced. This cannot be because of a rule of type (ii), because it has no nonterminals on its RHS, so we can postpone applying it as long as we want. Rules of type (i) cannot be needed either, because applying one would introduce another opening bracket right of the opening bracket we want to introduce. But in a rightmost–innermost rewriting strategy, we must have already treated that. □

An example grammar and derivation are shown in figure 6.

4. Decidable recognition of subclasses

Given the observation about corresponding CFG's just made, it is considerably easier to give a simple recognition algorithm that is similar to known algorithms for context-free grammars and tuple-based extensions such as LCFRS and LMG [Gro96].

We will augment a left-recursion-proof memoing recursive descent algorithm for context-free grammars with a mechanism that ensures that the derived strings have a balanced bracketing. Note that it should be able to perform this task taking as its input the *original string* w, and not its bracketed version $\overline{w}$.

For the very simple case that there is only one nonterminal B (henceforth "elliptic nonterminal") such that P contains an ellipsis production $A \ldots B \rightarrow C$, we can do this by counting the number of unmatched brackets for each constituent we recognize.

This is formalized in the deductive system in figure 7. Let the input string be $w = a_0 a_1 \cdots a_{n-1}$. The formula $A(i, j, l, r)$ can be derived whenever A will recognize a bracketing $\overline{v}$ of $v = a_i \cdots a_{j-1}$, where l is the number of unmatched closing brackets in $\overline{v}$, and r is the number of unmatched opening brackets. Now the context-free grammar derives a balanced bracketing of a word $w = a_0 a_1 \cdots a_{n-1}$ if and only if the calculus derives $S(0, n, 0, 0)$ from the axioms.

CF rule	Deduction rule
$A \rightarrow a_i$	$A(i, i+1, 0, 0)$
$A \rightarrow \lambda$	$A(i, i, 0, 0)$
$B \rightarrow]$	$B(i, i, 1, 0)$
$A \rightarrow BC$	$\dfrac{B(i, j, l_1, r_1) \quad C(j, k, l_2, r_2)}{A(i, k, l_3, r_3)}$ where $\begin{cases} \text{if } r_1 \leq l_2 \text{ then} & l_3 = l_1 + l_2 - r_1 \\ & r_3 = r_2 \\ \text{if } r_1 > l_2 \text{ then} & l_3 = l_1 \\ & r_3 = r_2 + r_1 - l_2 \end{cases}$
$A \rightarrow C\,[$	$\dfrac{C(i, j, l, r)}{A(i, j, l, r+1)}$

Figure 7: Recognition calculus for XG with one elliptic nonterminal.

For an algorithm based on this calculus to be terminating, we must impose a condition on the grammar; we need to know, given an input w of length n, how many brackets can at most be expected in any derived bracketing $\overline{w}$ of w.

Definition 4.1 Let $G = (N, T, S, P)$ be an arbitrary extraposition grammar; G is *linear bounded* if for any bracketing $\overline{w}$ of a string w, when $S \overset{G}{\Longrightarrow} \overline{w}$ then the number of bracket pairs in $\overline{w}$ is not greater than the length of w.

Instead of checking the linear boundedness property of a grammar, one will usually check some sort of *overt extraposition* property which says that by applying an ellipsis rule $A \ldots B \rightarrow C$ we necessarily immediately introduce one or more terminal symbols; i.e. fillers cannot be empty.

Proposition 4.2 *If an extraposition grammar G has only one ellipsis rule and is linear-bounded, then there is an algorithm that decides G-membership for a given string of length n in $O(n^6)$ time.*

Proof. We use a straightforward memoing recursive descent algorithm, and an argument in the style of [Lee93] and [Gro95]. Construct a memo table which contains a value **true**, **false** or **unknown** for each possible formula $A(i, j, l, r)$ where $0 \leq i, j, l, r \leq n$, and initialize all entries with the value **unknown**. We start computing $S(0, n, 0, 0)$. For each item to be computed, we first look if it has already been computed (memoed value is **true** or **false**); if not, then we recursively try all possible rules in the calculus in upward direction. The most complex case is a rule of type $A \rightarrow BC$. In that case we need to loop over j and l_3/r_3. So altogether we need to compute $O(n^4)$ items, which each take $O(n^2)$ elementary steps, amounting to a time complexity of $O(n^6)$ and a space complexity of $O(n^4)$. □

Proposition 4.3 *If G is an arbitrary linear-bounded XG, then membership for G can be decided in deterministic exponential time in terms of the length of the input.*

Proof (sketch) Instead of keeping track of the *number* of brackets, we now need to loop over arbitrary *sequences* of different types of bracket. The length of the sequences is bounded by the length of the input, hence there is an exponential number of possible sequences. A similar memoing algorithm or an alternating Turing machine construction completes the argument. □

All known examples of extraposition grammars satisfy the linear boundedness constraint. However, we can still improve the situation considerably, if we allow a more loose interpretation of XG derivation, in terms of the corresponding CFG. For if we allow derivations in which the derived bracketings are balanced not w.r.t. all bracket pairs $[_B,]_B$, but only with respect to each of the single pairs, then we have polynomial time recognition.

Definition 4.4 (loose derivation) Let G be an XG. Then G *loosely derives* w if there is a bracketing $\overline{w}$ of w such that $\mathcal{CF}(G)$ derives $\overline{w}$ and $\overline{w} \overset{*}{\leadsto} w$ under the following modified definition: If $\overline{v}$ does not contain any B-brackets $[_B$ and $]_B$, then

$$\overline{u}[_B\overline{v}]_B\overline{w} \leadsto \overline{uvw}$$

The algorithm can now be extended by having a pair l, r for each of the different annotated bracket pairs $[_B$ and $]_B$; it will ensure that the derived bracketed string is balanced only w.r.t. each individual pair of brackets. The complexity of the algorithm becomes $O(n^{3+3m})$ where m is the number of bracket pairs.

Proposition 4.5 *If G is an arbitrary linear-bounded XG, then membership for G under loose derivation can be decided in deterministic polynomial time in terms of the length of the input.*

5. Modelling extraposition islands

One of the virtues of the XG formalism mentioned in [Per81] is lost under the loose definition of derivation: the capacity to elegantly describe *extraposition islands* by introducing extra brackets.

The grammar in figure 8, taken from [Per81], derives simple English sentences with relative clauses. Figure 10 shows how this grammar derives sentences such as (5.7) that violate the *complex-NP constraint*: the first occurrence of *that* is linked to a trace within an extraposition island, which is not captured by the grammar.

(5.6) *The mouse that the cat chased squeaks*

(5.7) *The mouse that the cat that chased likes fish squeaks*

This is solved by replacing the second RelPhrase production by the following two rules:

(5.8)
$$\begin{array}{lcl} \text{RelPhrase} & \to & \text{BeginIsland RelMarker S EndIsland} \\ \text{BeginIsland} \ldots \text{EndIsland} & \to & \lambda \end{array}$$

This makes it impossible to derive the incorrect sentence (5.7), because the brackets introduced by RelMarker—Trace and BeginIsland—EndIsland are not balanced. An example derivation is in figure 11 (In the figure, BeginIsland and EndIsland are compressed to BI and EI respectively).

S	$\rightarrow$	NP VP
NP	$\rightarrow$	Det CN RelPhrase
NP	$\rightarrow$	Trace
VP	$\rightarrow$	VI
VP	$\rightarrow$	VT NP
RelPhrase	$\rightarrow$	λ
RelPhrase	$\rightarrow$	RelMarker S
RelMarker . . . Trace	$\rightarrow$	`that`

Figure 8: Simple grammar for English with relative clauses.

If we allow brackets of different types to be unbalanced, the method will not work. So we can't use the *loose* notion of derivation introduced in the previous section; so even in combination with the constraint of linear boundedness, which is satisfied in all known examples including Pereira's example, we are stuck with the exponential algorithm.

However, there is a dedicated construction we can add to XG with the loose notion of derivation, that marks extraposition islands. This will give us a version that *is* tractable. This seems to indicate that the original notion of XG derivation is inherently responsible for an excessive amount of generative capacity.

Add a new type of XG production; an *island production* $A(B) \rightarrow C$ states that A can be rewritten only if it then produces a string in which the brackets $[_B$ and $]_B$ are balanced. The recognition algorithm could be extended be specifying the deduction rule for the island production in figure 9, which requires that the numbers of unmatched opening and closing brackets in C are both zero.

Island rule	Deduction rule
$A(B) \rightarrow C$	$\dfrac{C(i,j,0,0)}{A(i,j,0,0)}$

Figure 9: The island rule in the recognition calculus

Implementing the bracket constraint in Pereira's grammar then amounts to changing the nonempty production for RelPhrase, to say that all Traces within a relativized sentence must be matched within that sentence.

(5.9) RelPhrase(Trace) $\rightarrow$ RelMarker RelS

which can be treated by the modified memoing recognition procedure running in at most $O(n^6)$ time.

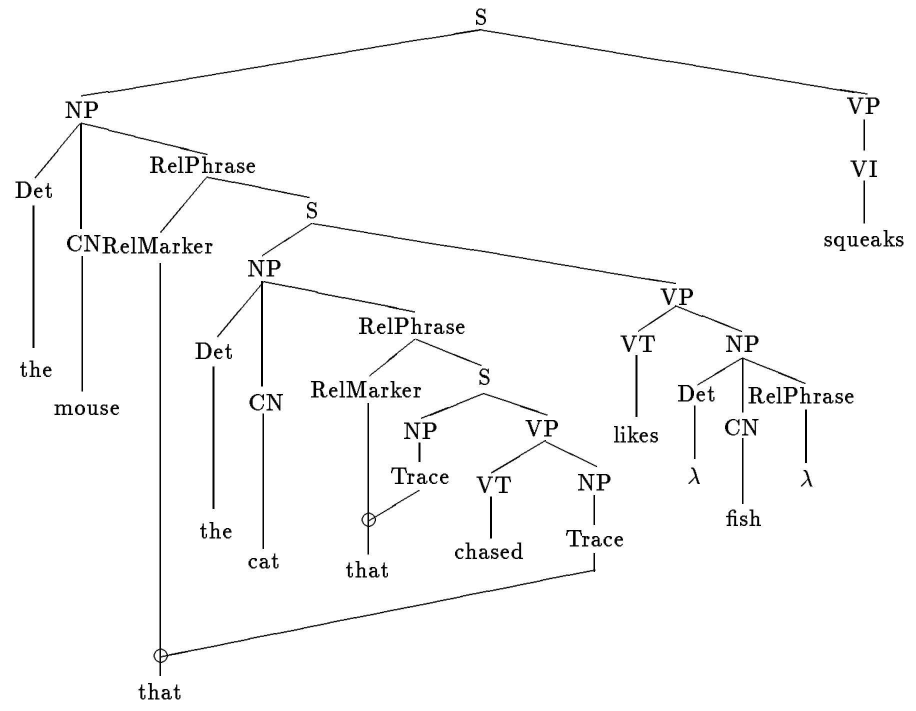

Figure 10: Derivation of a sentence violating the complex-NP constraint.

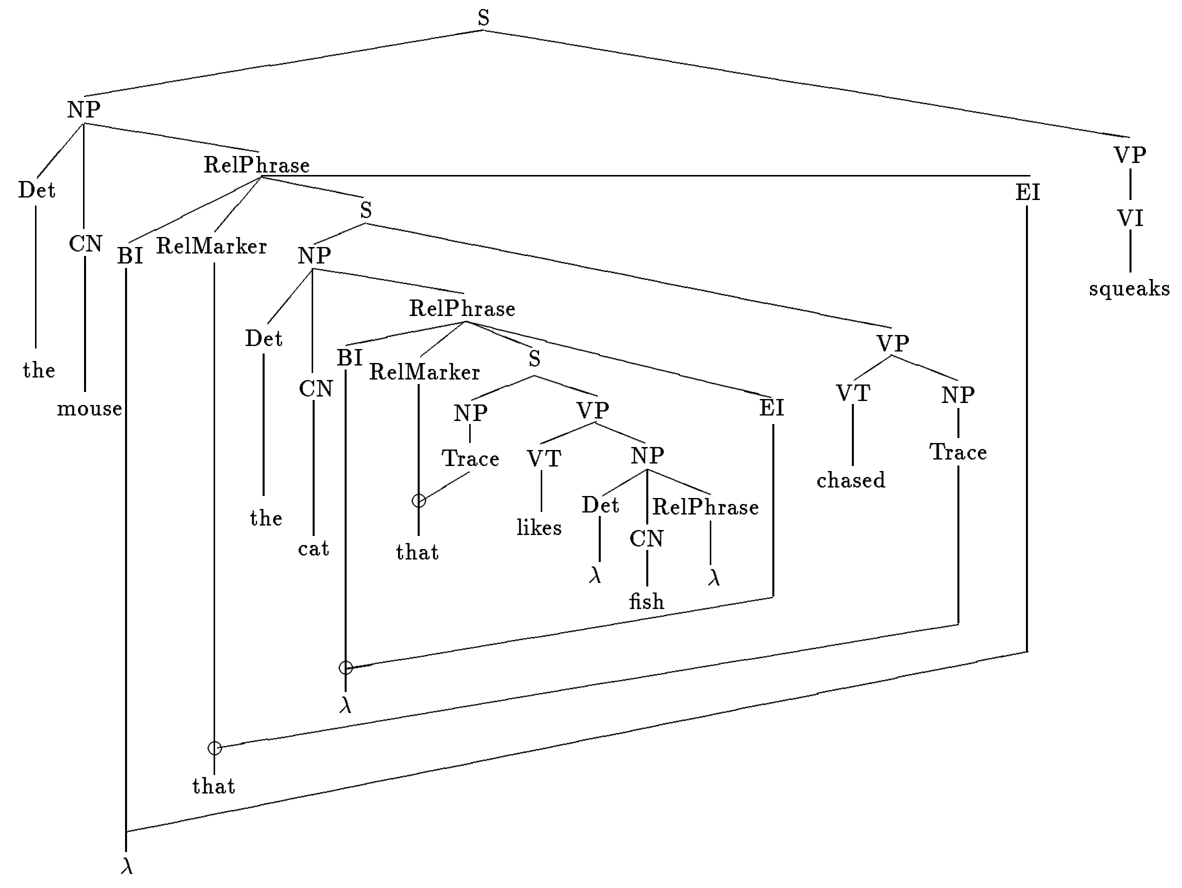

Figure 11: Implementation of the complex NP-constraint using multiple brackets

(1)	Rel	→	**dat** NP VP
(2)	S	→	NPtopic Vtopic NP VP
(3)	S	→	NP Vtopic NP VP
(4)	S	→	Vtopic NP VP
(5)	NPtopic . . . TopicTrace	→	NP
(6)	Vtopic . . . VTtrace	→	VT
(7)	Vtopic . . . VRtrace	→	VR
(8)	VP	→	NC VC
(9)	NC . . . NPtrace	→	NP NC
(10)	NC	→	ϵ
(11)	VC	→	VT NPtrace
(12)	VC	→	VTtrace NPtrace
(13)	VC	→	VR NPtrace VCisland
(14)	VC	→	VRtrace NPtrace VCisland
(15)	VCisland(VTtrace, VRtrace)	→	VC
(16)	NPtrace	→	TopicTrace

Figure 12: Independently branching XG for examples (2.4) and (2.5).

Using the island rule in the generalized form

(5.10) $\quad A(C_1, C_2, \ldots, C_n) \;\rightarrow\; B_1\, B_2\, \cdots\, B_m$

we can formulate the Dutch grammar of figure 4 in independently bracketing XG; the graph structures produced will be similar, but the pit-stopping effect is no longer necessary, and the graph will have less nodes (the nodes that vanish are ones that in the pit-stopping example had only a structural job, and did not connect nodes with true dependencies as introduced by agreement or case marking). A side effect is that the grammar has less reduplications of similar rules. Such a grammar is shown in figure 12 and a parallel derivation is shown in figure 13. Rule (15) makes each embedded VC an island for both types of verb trace, so no verb other than the first (i.e. the finite verb) can be fronted.

6. Non-decidability of generic XG

In three steps, we show that XL includes the r.e. languages. First, we show that XL is closed under homomorphism. We then show how the bracketing mechanism of XG derivations can be used to mimick a context-free derivation. Finally, we construct a bilinear XG that will recognize the intersection of two context-free languages. We will use the bracket-free XG derivation of definition 3.2.

Proposition 6.1 *The languages recognized by XL are closed under homomorphism.*

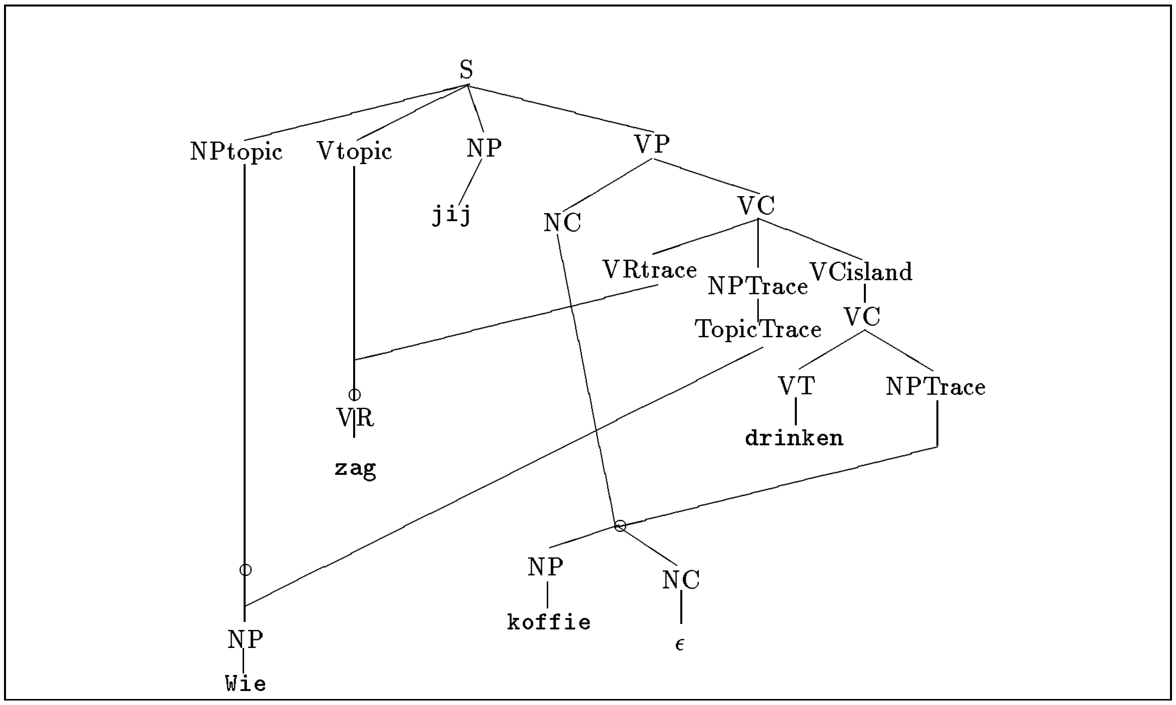

Figure 13: An independently bracketing derivation of sentence (2.5).

Proof. Analogous to the one for context-free grammars; replace terminals a in the XG rules by their images $h(\mathtt{a})$. □

We now begin the construction that mimicks the nonterminal rules of a context-free grammar in Chomsky normal form using ellipsis rules.

Definition 6.2 Let $G = (N, T, S, P)$ be a context-free grammar in CNF, that is, its productions are of the form $S \longrightarrow \lambda$, $A \longrightarrow \mathtt{a}$ or $A \longrightarrow BC$. Then define the set of XG productions P_X as follows: for every rule $A \longrightarrow BC$ in P let P_X contain the rule $B \ldots C \longrightarrow A$.

Definition 6.3 (sound labeling) Let G be a context-free grammar. A *G-sound labeling* is a sequence $A_1 w_1 A_2 w_2 \cdots A_n w_n$ such that for each $1 \leq i \leq n$, $A_i \overset{G}{\Longrightarrow} w_i$.

Lemma 6.4 *Let α be a G-sound labeling, $R \in P_X$ and $\alpha \overset{R}{\Rightarrow} \beta$. Then β is a G-sound labeling.*

Proof. Let R be $B \ldots C \longrightarrow A$; the general case is $\alpha = \gamma B u C v \delta$, and $\beta = \gamma A u v \delta$, where γ and δ are G-sound labelings, so δ is either empty or begins in a nonterminal. Because α is a G-sound labeling, $B \overset{G}{\Longrightarrow} u$ and $C \overset{G}{\Longrightarrow} v$. So $A \overset{G}{\Longrightarrow} uv$, and β is a G-sound labeling. □

Definition 6.5 (terminal labeling) A *G-sound terminal labeling* of $w = a_1 a_2 \cdots a_n$ is a labeling $A_1 a_1 A_2 a_2 \cdots A_n a_n$ such that for each $1 \leq i \leq n$, P contains the production $A_i \longrightarrow a_i$.

Proposition 6.6 *Let G be a context-free grammar and $w = a_1 a_2 \cdots a_n$ a terminal word. Then $A \overset{G}{\Longrightarrow} w$ if and only if there is a G-sound terminal labeling α of w such that $\alpha \overset{P_X}{\Longrightarrow} Aw$.*

Proof. The *if* part follows from the lemma. *Only if* is proved by induction on the depth of the context free derivation tree; let R be the rule applied in its top node;

Terminal case $R = A \rightarrow a$. Obvious: $Aa \overset{P_X}{\Longrightarrow} Aa$.

Nonterminal case $R = A \rightarrow BC$. If $A \overset{G}{\Longrightarrow} w$ then there are u and v such that $w = uv$, $B \overset{G}{\Longrightarrow} u$ and $C \overset{G}{\Longrightarrow} v$. By i.h. there are terminal G-sound labelings β and γ such that $\beta \overset{P_X}{\Longrightarrow} Bu$ and $\gamma \overset{P_X}{\Longrightarrow} Cv$. Then $\beta\gamma \overset{P_X}{\Longrightarrow} BuCv \xrightarrow{R} Auv$. □

Proposition 6.7 *Let L_1 and L_2 be context-free languages. Then there is an XG X such that $L(X) = L_1 \cap L_2$.*

Proof. Let L_1 and L_2 be recognized by the CNF context-free grammars $G_1 = (N_1, T, S_1, P_1)$ and $G_2 = (N_2, T, S_2, P_2)$, respectively, and assume w.l.o.g. that $N_1 \cap N_2 = \emptyset$. Then construct an extraposition grammar $X = (N_1 \cup N_2 \cup \{\Sigma, \Phi\}, T, \Sigma, P)$ as follows:

1. P contains $\Sigma \rightarrow \lambda$ if the empty string is in both L_1 and L_2.
2. P contains the rule $\Sigma \rightarrow \Phi S_1$.
3. P includes the nonterminal rules of G_1.
4. If P_1 contains $A \rightarrow a$ and P_2 contains $B \rightarrow a$, then P contains the rule $A \rightarrow Ba$.
5. P includes $P_X(G_2)$.
6. P contains the rule $\Phi \ldots S_2 \rightarrow \lambda$.

Let d be any X-derivation of w. Then it is easy to see that there is a derivation d' of w that applies the rules of type 1 first, then rules of type 2, and so forth.

Let $w = a_1 a_2 \cdots a_n$ be nonempty. If d is a derivation of w, then d' starts in the derivation $S \rightarrow \Phi S_1 \Longrightarrow^* \Phi B_1 a_1 B_2 a_2 \cdots B_n a_n$ of a terminal G_2-sound labeling using only rules of type 2, 3 and 4. From this initial segment of d' we immediately have a G_1-derivation of w. Because d' will end in an application of rule 6, we must have $B_1 a_1 B_2 a_2 \cdots B_n a_n \overset{P_X}{\Longrightarrow} S_2 w$, which by proposition 6.6 is equivalent to saying that G_2 derives w.

The reverse implication is now obvious. □

Corollary 6.8 *The class XL of languages recognized by XG includes all recursively enumerable languages.*

Proof. We have proved that XL is closed under homomorphism and contains the class $\{L_1 \cap L_2 \mid L_1 \text{ and } L_2 \text{ context-free}\}$; it is a familiar result (see e.g. [Gin75] p. 125) that any such class includes all r.e. languages. □

7. Conclusions

Because we have shown that XG in their original form from [Per81] describe any r.e. language, the original formalism can not be seen as a 'minimal' extension of CFG. However, the examples in section 1 indicate that the 'practical' power of the ellipsis construction is indeed limited: when we want to extend the coverage of an XG analysis of the Dutch VP, we get multiple threads of extraposition which can only be described by constructing cumbersome grammars with "manually copied" rule schemas.

So we introduced parallel bracketing versions of XG, which more easily describe the parallel threads of movement, and turn out to have the additional benefit that they are tractable.

References

[Gin75] S. Ginsburg. *Formal Languages.* North-Holland/Am. Elsevier, Amsterdam, Oxford/New York, 1975.

[Gro95] Annius V. Groenink. A Simple Uniform Semantics for Concatenation-based Grammar. In *Proceedings of the TWLT10/AMiLP meeting, University of Twente*, December 1995.

[Gro96] Annius V. Groenink. Mild Context-Sensitivity and Tuple-based Generalizations of Context-Free Grammar. To appear in the special MOL4 issue of *Linguistics and Philosophy*, 1996.

[Lee93] René Leermakers. *The Functional Treatment of Parsing.* Kluwer, The Netherlands, 1993.

[Per81] Fernando Pereira. Extraposition Grammars. *Computational Linguistics*, 7(4):243–256, 1981.