Logic and Syntax — A Personal Perspective

MARCUS KRACHT

Abstract. Logic, in particular modal logic, can not only be used in semantics but also in syntax. Here we study syntactic formalisms from a logical point of view. This is a reflection in terms of logic of the view that grammar is nothing but a catalogue of abstract descriptive principles rather than a productive device, a view that was current mainly in the 80ies. The main target of this inverstigation is the theory of Government and Binding, although the methods allow for a much wider application.

1. Introduction

When one talks about the connection between logic and language then the first thing that comes to mind is semantics. In fact, many distinguished logicians have concerned themselves quite intensively with the questions of meaning in natural languages, for examples Russell, Carnap, Church, and Montague. Their reflections have led to the development of Type Theory and Categorial Grammar. This is a rather fruitful area of research. Yet, there is another field of application of formal logic to language that has been sparked off by the trend in the 80ies to view grammars not as production devices but as theories about language — where the word 'theory' is used here in the logical sense. The exact meaning of this statement will become clear later on. One should say here that this view has been the result of a confluence of trends. One was the fact that transformational grammar was struggling to get rid of its plethora of transformations by trying to isolate a few abstract principles. The task of these principles is to constrain the transformations in such a way that all and only the 'natural' transformations would be allowed by the principles. This finally led to the elimination of all variety in the transformational kernel (there was only one transformation left, called Move–α). It was then only a small step to a purely descriptive theory of language, which has in that time been advocated by Jan Koster [10]. Another trend that brought up the descriptive approach to grammars was the change from imperative programming languages to so–called logical programming languages, in particular Prolog. Many linguists got persuaded that the best method of developing computer software for natural language manipulation (and here I mean for example analysis, generation and translation) was to use logical programming languages. The hope was that they would liberate the linguist from questions of program execution and allow to just develop a database that would contain simply the linguistic facts. So the linguist would be able to concentrate on the development of this database rather than worrying about lousy details of implementation. This led in turn to the creation of grammar formalisms that were highly reminiscent of the target implementation language. The peak of this development are hpsg (Head Driven Phrase Structure Grammar) and lfg (Lexical Functional Grammar), both based on feature–value formalisms. It is interesting to note here that the pendulum has already returned to the other side. Transformational grammar has radically abandoned any use of descriptive principles in favour of a purely procedural approach. In computer science other programming languages and other paradigms (for example functional programming) have eclipsed Prolog. In linguistics it is not exactly clear what the future will bring but as a matter of fact the grammar formalisms of the eighties, rather than shedding new light on old problems have often enough simply created many internal problems, mainly caused by an improper understading of the nature of the symbolism. The hazy propaganda behind Prolog has done its part in mystifying the new tools rather than clarifying their potential and their limits.

In this paper I will speak about some applications of logic, in particular modal logic, to the field of syntax. Several people have contributed to this enterprise, and they will all be named in due course. However, as this title suggests, the outline will be quite a personal one.

2. The Logical View on Grammar

For a logician it is perhaps quite easy to understand the following description of a grammar. A *theory* about L is a set T such that if $T \vdash \varphi$ then φ holds of all structures of L. A *grammar* of a language L is a theory T such that $T \vdash \varphi$ iff φ is true in all structures of L. So, a grammar is such that its consequences are all and only the L–valid sentences. Structures are either strings, syntactic trees, or pairs $\langle \vec{x}, \sigma \rangle$, where $\vec{x}$ is a string and σ is the meaning of $\vec{x}$. A run of the mill grammar of English, for example, can be seen as a theory about English. Hardly can they claim to be grammars in the technical sense above. To give some examples: a theory of English may contain statements such as *the order of the main constituents in a declarative sentence is subject–verb–object*, *the verb agrees in number with the subject* and so on. Of course, they have to be suitably coded into some logical language, and we have to define many notions such as subject and object, but these are problems of execution rather than principle.

The problem with the logical definition of grammar is that it does not tell us how the sentences of a language can be generated and how they can be analyzed. Hence a large part of our language activity is left unaccounted for. Moreover, still today it is not clear whether what we have in our mind is some knowledge of the syntax in descriptive terms such as the statements above or whether language as we see it is simply an artefact of a rather complex language generator–analyser whose structure we yet have to find. Chomsky has over and over iterated the point that in his view collecting facts about how the sentences of the language look like is not to do lingustics at all. It is like collecting butterflies rather than doing biological research. He has introduced the distinction between E–language — this is what we see at the surface — and I–language — this is the manifestation of the language in the brain. While theories about E–language are easy to develop, studying I–language is still today hopelessly difficult, and all we have at the moment are hypotheses.

Chomsky has also always been against descriptions and in favour of procedures. He has defined a *generative grammar* of a language L to be an algorithm that generates all and only the correct structures of L. This definition has been quite popular in the 70ies. It defines languages by means of a generating device. The generative grammar already tells us how the structures are generated. In order to account for our analytic facility we also need to have a recognizing device for the language. Fortunately, many language classes not only have a characterization in terms of a generating device but also in terms of an automaton. For example, recall that a language is called regular if it can be generated by a *right–regular grammar*, that is, a grammar with rules of the form $\mathtt{X} \to \mathtt{aY}$, or $\mathtt{X} \to \mathtt{a}$, $\mathtt{a}$ a terminal symbol and $\mathtt{X}$ and $\mathtt{Y}$ nonterminal symbols. On the other hand, a language is regular iff it can be recognized by a finite state automaton. Hence, given any regular language, there exists a generating device (a right–regular grammar) and an analysing device (the finite state automaton). Moreover, one can construct one from the other.

Formal language theory has pretty much remained faithful to this approach. Many language classes have been defined by means of generating devices (= generative grammars) and analysing devices (= automata) have been found. There is, however, a problem: we hardly ever have a complete description of a natural language. Our knowledge even of the best studied languages — such as English — is still incomplete. However, we would not like to wait until our knowledge is perfect before we write a grammar of it. And, furthermore, even though it may be enough to describe English in terms of some generative grammar and an automaton, the working linguist as much as the individuum trying to learn English nevertheless have to work with descriptions of it, that is to say, theories. What we are therefore ideally like to have is some way of mediating between these three ways of defining a language: the descriptive, the generative and the analytic definition. There is an additional reason for trying to develop such tools. Grammar formalisms and theories are very often substantially different in the way they describe language. Categorial grammar, HPSG and transformational grammar represent different extremes and still at present it is not really understood how to translate between a grammar presented in categorial form and one that is presented in HPSG or transformational grammar. But if one would know how to do that then we would be able to see what in the different formalisms is mere change of notation and what is a substantial difference, and we would be able to compare different theories even if they were presented in highly different form.

Matters are however not that simple. We will see shortly that many problems turn out to be undecidable. Therefore, only part of the project can at all be carried out to satisfaction. However, to see that matters are this way requires some logical analysis to which we now turn.

We will discuss mainly two approaches, one based on monadic second order logic (MSO) and the other on Propositional Dynamic Logic (PDL). We recall here only the necessary basic facts. MSO is an extension of predicate logic by means of variables over unary predicates and quantifiers over such variables. Clearly, MSO can have any number of constants for predicates, functions and individuals. The sets of variables are $\{x_i : i \in \omega\}$ and $\{P_i : i \in \omega\}$.

Now, PDL or rather PDL with converse, has two sorts of expressions: programs and formulae. Programs are formed by means of program composition (;), nondeterministic union ($\cup$), iteration (*) and converse ($^\smile$). The set of propositional variables is $\{p_i : i \in \omega\}$. Formulae are composed by means of the usual boolean connectives. If α is a program and φ a formula, then $\langle\alpha\rangle\varphi$ and $[\alpha]\varphi$ is a formula. If φ is a formula then $\varphi?$ is a program. The set of basic programs is denoted by Π_0. There are the following equations for the converse

$$\begin{array}{lcl} R^{\smile\smile} & = & R \\ (R;S)^\smile & = & S^\smile ; R^\smile \\ (R \cup S)^\smile & = & R^\smile \cup S^\smile \\ (R^*)^\smile & = & (R^\smile)^* \end{array}$$

Therefore, PDL with converse is expressively equivalent to PDL over a set Π_0' of basic operators which contains for every $\zeta \in \Pi_0$ both ζ and its converse. There is a canonical translation of PDL–formulae into MSO:

$$\begin{array}{lcl} p_i^\ddagger & := & P_i(x) \\ (\varphi \wedge \psi)^\ddagger & := & \varphi^\ddagger \wedge \psi^\ddagger \\ (\neg\varphi)^\ddagger & := & \neg\varphi^\ddagger \\ ([\alpha]\varphi)^\ddagger & := & \forall y.(\alpha(x,y)^\dagger \rightarrow \varphi^\ddagger[y/x]) \end{array}$$

In the last clause y is a variable not occurring in $\varphi^\ddagger$. Now $(\alpha(x,y))^\dagger$ is defined as follows. Pick for each $\zeta \in \Pi_0$ a constant ζ° denoting a binary relation. Now put

$$\begin{array}{lcl} \alpha(x,y)^\dagger & := & \alpha^\circ(x,y) \qquad \text{if } \alpha \in \Pi_0 \\ (\alpha;\beta)(x,y)^\dagger & := & (\exists z)(\alpha(x,z)^\dagger \wedge \beta(z,y)^\dagger) \\ (\alpha \cup \beta)(x,y)^\dagger & := & \alpha(x,y)^\dagger \vee \beta(x,y)^\dagger \\ (\varphi?)(x,y)^\dagger & := & x \doteq y \wedge \varphi^\ddagger \\ (\alpha^*)(x,y) & := & (\forall S)((\forall zz')(S(z) \wedge \alpha(z,z')^\dagger \rightarrow S(z')) \wedge S(x) \rightarrow S(y)) \end{array}$$

Here again some precautions concerning the choice of variables must be made. We will freely switch between viewing a structure as a structure for PDL (with converse) and the corresponding MSO–structure. Furthermore, there will appear a number of linguistic structures (strings, trees, and so on). We will usually only sketch how they can be turned into models for the languages under discussion. A reader with only little experience in logic will however already be able to fill in the details.

3. Regular Languages

Let $A = \{\mathsf{a}_i : i < n\}$ be some finite set, called the *alphabet.* A finite sequence over A is called a *string.* ε is called the *empty string.* A^* is the set of all strings over A. A *(string) language* over A is a subset of A^*. We denote by $\vec{x}\vec{y}$ the result of concatenating $\vec{x}$ and $\vec{y}$. Furthermore, for sets $R, S \subseteq A^*$ we put $R \cdot S := \{\vec{x}\vec{y} : \vec{x} \in R, \vec{y} \in S\}$ and call it the *complex product* of R and S. We define $R^0 := \varepsilon$ and $R^{n+1} := R \cdot R^n$. Finally, we put $R^* := \bigcup_{n\in\omega} R^n$ and call it the *iteration* of R. Instead of $\{\vec{x}\}$ we simply write $\vec{x}$. A language over A^* is *regular* if it is empty or can be produced from $\mathtt{a}_i$, $i < n$, and ε by means of complex products, unions and iteration.

A *context–free grammar* over A is a triple $G = \langle N, \mathtt{S}, R\rangle$ where N is a finite set disjoint from A, $\mathtt{S} \in N$, and $R \subset N \times (N \cup A)^\star$ a finite set. (One writes $\mathtt{X} \to \vec{x}$ rather than $\langle \mathtt{X}, \vec{x}\rangle$ for a rule. Moreover, lower case letters denote symbols from A while upper case letters are reserved for symbols from N. For general reference on context–free grammars and languages see [6].) N is called the set of *nonterminal symbols*, $\mathtt{S}$ the *start symbol* and R the set of *rules*. Given some strings $\vec{x}, \vec{y} \in (N \cup A)^\star$, we say that $\vec{y}$ is *one–step derivable* from $\vec{x}$ if there are $\vec{u}$, $\vec{v}$, $\vec{w}$ and $\mathtt{X}$ such that (1) $\vec{x} = \vec{u}\mathtt{X}\vec{v}$, (2) $\vec{y} = \vec{u}\vec{w}\vec{v}$ and (3) $\mathtt{X} \to \vec{w} \in R$. $\vec{y}$ is *derivable* from $\vec{x}$ if $\vec{y}$ can be derived from $\vec{x}$ by a series of one–step derivations. Finally, G *generates* $\vec{y}$ if $\vec{y}$ is derivable from $\mathtt{S}$. We write $L(G)$ for the set of all strings generated by G. A language L is *context–free* if there is some context–free grammar G such that $L = L(G)$. A context–free grammar is *right–regular* if all rules have the form $\mathtt{X} \to \mathtt{aY}$ or $\mathtt{X} \to \mathtt{a}$.

A finite–state automaton over A is a quadruple $\mathfrak{A} = \langle Q, q_0, F, \delta\rangle$, where Q is a finite set, the set of *states*, $q_0 \in Q$ a distinguished state, the *initial state*, $F \subseteq Q$ a subset of Q of so–called *final* or *accepting states*, and finally $\delta : Q \times A \to \wp(Q)$ the so–called *transition function*. We write $q \xrightarrow{\mathtt{a}} q'$ if $q' \in \delta(q, \mathtt{a})$. Furthermore, we write $q \xrightarrow{\vec{x}\vec{y}} q'$ iff there is a $q'' \in Q$ such that $q \xrightarrow{\vec{x}} q'' \xrightarrow{\vec{y}} q'$. We say that $\mathfrak{A}$ *accepts* $\vec{x}$ if there exists a $q \in F$ such that $q_0 \xrightarrow{\vec{x}} q$. We write $L(\mathfrak{A})$ for the set of all strings accepted by $\mathfrak{A}$. The following is a well–known theorem.

Theorem 3.1 (Kleene)**.** *The following are equivalent for any finite set A and any language L over A.*

1. *L is regular.*
2. *There is a right–regular grammar G such that $L = L(G)$.*
3. *There is a finite–state automaton $\mathfrak{A}$ such that $L = L(\mathfrak{A})$.*

We note here that there are algorithms that construct G given $\mathfrak{A}$ and vice versa. This theorem fully describes the connection between the generative and the analytic aspect of regular languages. Moreover, there is a description of regular languages by means of regular terms. A regular term is a term built from the symbols of A using complex product, union and iteration. For example, $(\mathtt{a}^2 \cdot \mathtt{b})^* \cup \mathtt{a}$ is a regular term. It is also possible to construct such terms given G, and to construct G given the regular term.

For many purposes, however, the regular terms are not strong enough. They are rather clumsy and we wish to replace them by some higher order language. For example, take monadic second order logic using the following non–logical symbols: a binary relation symbol $<$ and a unary predicate P_i for each $i < n$. A string is a model for this language in the following sense. Take the string $\vec{x} = x_0 x_1 \ldots x_{n-1}$. Now let $n := \{0, 1, \ldots, n-1\}$ and $m(\vec{x}) := \langle n, <, \langle P_i^{\vec{x}} : i < n\rangle\rangle$, where $P_i^{\vec{x}}(j)$ iff $\vec{x}_j = \mathtt{a}_i$. In this way we obtain for each language $L \subseteq A^*$ a monadic second order theory $MSO(L)$, namely the common logic of all structures $m(\vec{x})$ such that $\vec{x} \in L$. In particular, let $T_0 := MSO(A^*)$ be the logic of all string models. It is finitely axiomatizable and contains eg sentences expressing that $<$ is a discrete total ordering with endpoints and that for each element x, exactly one P_i is true at x. Further, given a theory T, we let $L(T)$ be the set of all strings $\vec{x}$ such that $m(\vec{x}) \models T$. L is *MSO–definable* if $MSO(L)$ is finitely axiomatizable and $L = L(MSO(L))$. The following now holds.

Theorem 3.2 (Büchi)**.** *The following are equivalent for any finite alphabet A and any language L over A.*

1. *L is regular.*
2. *L is MSO–definable.*
3. *L is definable in MSO by means of a Σ^1_1–sentence.*

This state of affairs is quite satisfactory. It is again possible to pass from a generating grammar to a MSO–sentence and back.

An alternative to monadic second order is modal logic. The string models may also be viewed as models of some polymodal logic. There are several choices with respect to the modal operators. The simplest choice is the following. We will introduce boolean constants, a_i, $i < n$, and two modal operators, $\diamondsuit$ and $\diamondsuit$. Both are tense duals of each other and satisfy alt_1. A string $\vec{x} = x_0 x_1 \dots x_{n-1}$ is made into a model for this language as follows. We put $\mathfrak{F}(\vec{x}) := \langle n, \lhd, \rhd, \langle \mathsf{a}_i^{\vec{x}} : i < n \rangle \rangle$, where $i \lhd j$ iff $j = i+1$, $i \rhd j$ iff $j \lhd i$ and $\mathsf{a}_i^{\vec{x}} := \{j : x_j = \mathsf{a}_i\}$. It turns out however that this logic admits unintended models, namely infinite ones. We therefore add the transitive closure $\diamondsuit^*$ and $\diamondsuit^*$. Both shall satisfy G, by which we successfully eliminate all finite models. Given a language $L \subseteq A^*$ we define $ML_4(L)$ to be the common logic of all $\mathfrak{F}(\vec{x})$, where $\vec{x} \in L$. Given a modal logic Θ containing $ML_4(A^*)$ we put $L(\Theta) := \{\vec{x} : \mathfrak{F}(\vec{x}) \models \Theta\}$. Using the above theorem we deduce that $L(\Theta)$ is always regular. However, not all regular languages give rise to different logics, and so in contrast to the case of monadic second order logic, this modal logic is actually not faithful. Therefore we strengthen our logic once more. This time we take the full power of Propositional Dynamic Logic with converse, $\mathbf{PDL}^{\smile}$. In our case, one program is sufficient, so we put $\Pi_0 := \{\zeta\}$. The interpretation of ζ coincides with that of $\diamondsuit$, and so $\diamondsuit$ corresponds to $\zeta^{\smile}$, $\diamondsuit^*$ with ζ^* and $\diamondsuit^*$ with $(\zeta^{\smile})^*$. Notice that the interpretation of the composite programs can be defined by means of MSO–sentences, so that $\mathbf{PDL}^{\smile}$ is a fragment of monadic second order logic. Therefore, any logic containing the logic of strings over A defines a regular set of strings. The converse also holds, as we will show. The idea is the following. Start with a right–regular grammar $G = \langle N, \mathtt{S}, R\rangle$ and introduce for every nonterminal symbol $\mathtt{X}$ a constant $c(\mathtt{X})$.

$$\begin{array}{rcl} \mathtt{X}^\dagger & := & c(\mathtt{X}) \leftrightarrow \bigvee \langle \mathtt{a} \wedge \langle \zeta \rangle c(\mathtt{Y}) : \mathtt{X} \to \mathtt{aY} \in R \rangle \\ G^\dagger & := & \bigvee \langle \mathtt{X}^\dagger : \mathtt{X} \in N \rangle \end{array}$$

Here, the idea is that $c(\mathtt{X})$ is true at a point if that point is the left periphery of a constituent of type $\mathtt{X}$. The logic defined by this axiom together with those defining all strings is denoted by Θ_G. The frames for Θ_G are not really strings over A since we have the additional constants as well. The aim is therefore to eliminate these constants. This can be done rather easily as follows. First, observe that $G^\dagger$ is a system of equivalences of the form $p \leftrightarrow \psi(p)$ where $\psi(p)$ contains p only embedded in $\langle \zeta \rangle$. Now the following holds.

Lemma 3.3. *In the logic of finite strings, the formula $\varphi(p) := p \leftrightarrow \alpha \vee \beta \wedge \langle \zeta \rangle p$, where α and β do not contain p, is a global implicit definition of p. That is to say, it holds that if in some model $\varphi(p)$ and $\varphi(q)$ holds everywhere then $p \leftrightarrow q$ holds everywhere as well.*

The proof is a simple induction on the depth of a point in the frame. Now, an explicit definition on p is a formula ψ in which p does not occur such that if in some model $\varphi(p)$ holds globally, then $p \leftrightarrow \psi$ holds globally as well. In the present circumstances such a formula exists. It is

$$\langle (\beta?; \zeta)^* \rangle \alpha$$

Using this, we can step by step eliminate the constants $c(\mathtt{X})$ and replace $G^\dagger$ by a constant formula that does not make use of them any more. In that way, the regular language is shown to be axiomatically definable in $\mathbf{PDL}^{\smile}$. Notice that we have actually produced an algorithm to convert a regular grammar into an axiom. Furthermore, the axiom is easily translated into monadic second logic, so that all transitions here are constructive.

Theorem 3.4. *The following are equivalent for any finite alphabet A and any language L over A.*

1. *L is regular.*
2. *L is $\mathbf{PDL}^{\smile}$–definable.*

We remark here that instead of **PDL** with converse, we could have taken standard **PDL** based on two operators which are tense duals of each other. However, these two are expressively equivalent, as remarked above.

4. Context–Free Languages

The results of the previous section showed how we can translate between various ways of defining the same set of objects, in this case finite strings over A. The only problem is that regular languages are not powerful enough. Even programming languages are not regular languages. They are — with some small exceptions — context–free. Fortunately, context–free languages turn out to have a theory that is almost as favourable as that of the regular languages. The following is the correspondence between the generative and the analytic view. (We will not define here the notion of a pushdown automaton.)

Theorem 4.1 (Chomsky). *Let A be a finite alphabet and L a language over A. Then following are equivalent.*

1. *There is a context–free grammar G such that $L = L(G)$.*
2. *There is a pushdown automaton $\mathfrak{A}$ such that $L = L(\mathfrak{A})$.*

Next we will aim to extend our results on definability. For that we must abandon our model structures. They are too poor that we can define anything non–regular even by using monadic second order logic. (However, second order logic would be enough if we use binary relations, for example. But that is far too powerful.) Therefore, rather than talking about strings we shall now talk about *trees*. Recall that each derivation of a string $\vec{x}$ in a context–free grammar also defines a so–called parse or analysis tree of $\vec{x}$ (which usually depends on the derivation). This tree is a pair $\langle T, \ell\rangle$, where $T \subset \omega^*$ is a tree domain and $\ell : T \to A \cup N$ a labelling function. Here, a *tree domain* is a set T of finite sequences of natural numbers such that (i) if $\vec{x} \in T$ and $\vec{y}$ is a prefix of $\vec{x}$ then also $\vec{y} \in T$ and (ii) if $\vec{x}i \in T$ and $j < i$ then also $\vec{x}j \in T$. The tree ordering is taken to be the converse of the prefix ordering. (So the empty sequence is the maximal element.) For future reference a *constituent* of T is a subset of the form $\downarrow x := \{y : y \leq x\}$. If the tree is labelled, the constituent will also be a labelled tree, with the appropriately restricted labelling function.

We give an example. The following is a context–free set of rules.

$$\begin{array}{lll@{\qquad}lll}
\mathtt{S} & \to & \mathtt{S\,S} & \mathtt{S} & \to & \mathtt{A\,B} \\
\mathtt{S} & \to & \mathtt{B\,A} & \mathtt{A} & \to & \mathtt{S\,A} \\
\mathtt{A} & \to & \mathtt{A\,S} & \mathtt{A} & \to & \mathtt{a} \\
\mathtt{B} & \to & \mathtt{S\,B} & \mathtt{B} & \to & \mathtt{B\,S} \\
\mathtt{B} & \to & \mathtt{b} & & &
\end{array}$$

The start symbol is $\mathtt{S}$ and the terminal alphabet is $\{\mathtt{a}, \mathtt{b}\}$. The following is a derivation of $\mathtt{ababab}$.

$$\mathtt{S}, \mathtt{AB}, \mathtt{ASB}, \mathtt{ASSB}, \mathtt{aSSB}, \mathtt{aSSb}, \mathtt{aBASb}, \mathtt{abASb}, \mathtt{abaSb}, \mathtt{abaBAb}, \mathtt{abaBab}, \mathtt{ababab}$$

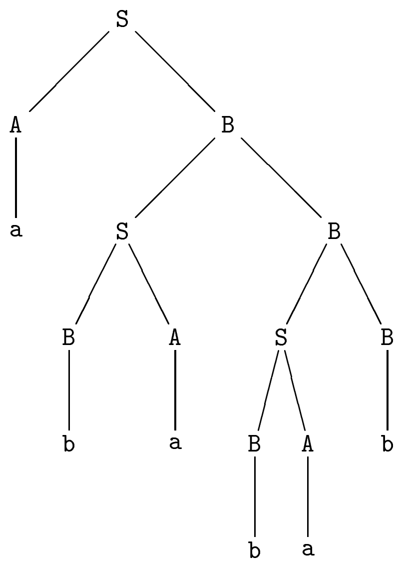
This grammar generates the set of all strings over $\mathtt{a}$ and $\mathtt{b}$ which contain the same number of $\mathtt{a}$'s and $\mathtt{b}$'s. A derivation defines a so–called *parse–tree* or *analysis–tree* (see [6] if details are needed). In the case of the previous derivation we get the tree shown in Figure 1.

Now take monadic second order logic again, with two binary relations $<$ (daughter of) and $\sqsubset$ (left of) and unary predicates $P_{\mathtt{a}}$ for each $\mathtt{a} \in A$ and $P_{\mathtt{X}}$ for each $\mathtt{X} \in N$. We can axiomatize the theory of all labelled trees over A and N. This can be done for any context–free set of trees. In fact, for this we hardly need the power of MSO. But now notice the following. We may remove from the signature any number of predicates $P_{\mathtt{X}}$. (The resulting structures are trees in which the labelling function is only partially defined.) If S is a set of structures that is obtained in this way we say that S is the *projection* of a context–free set of trees. Any projection of a context–free set of trees is also MSO–definable. For the predicates for the nonterminals can be eliminated by existential quantification. The converse also holds, and this is the first result we note. Here, a set S is *boundedly branching* if there is a number n such that each tree of S is at most n–branching.

Theorem 4.2 (Doner, Thatcher, Rogers). *Let A be a finite alphabet and S some set of finite trees over A. Then the following are equivalent.*

1. *S is the projection of a context–free, boundedly branching set.*

FIGURE 1. An analysis tree



2. *S is MSO–definable.*

The modal analysis can likewise be boosted up to the context–free case. We take two basic operators, ζ (interpreted as *immediate left sister of*) and η (interpreted as *immediately below of*). For each symbol of A and N we add constants. The logic of n–branching trees where each constant is true at exactly one node is denoted by $\mathbf{Olt}_n$. Consider now an extension of $\mathbf{Olt}_n$ by a constant axiom χ. Then the set of finite trees satisfying $\mathbf{Olt}_n \oplus \chi$ is a context–free set. We say that a set of finite n–branching trees is $\mathbf{PDL}^{\smile}$–definable if it is the model set of a logic $\mathbf{Olt}_n \oplus \chi$, where χ is a constant formula.

Theorem 4.3. *Let A be a finite alphabet and S some set of finite trees over A. Then if S is $\boldsymbol{PDL}^{\smile}$–definable, S is the projection of a context–free, n–branching set.*

The use of these theorems is the following. Suppose that we have a rather high order principle on syntactic structures, such as *every trace is 1–subjacent to its antecedent* — you do not need to know here what that exactly means — then we have established that if this principle can be codified using either monadic second order logic or PDL with converse, it actually defines a set of context–free structures (modulo projection). In this way both James Rogers [22] and [11] have shown that various theories of grammar (locality theory by Manzini [15] and relativized minimality by Rizzi [21]) predict that English is context–free since it does not participate in unbounded head–movement, which has been isolated as the only source of non–context–freeness.

5. Monadic Second Order Logic or Dynamic Logic?

The preceding discussion seems to imply that there is actually no difference between the monadic second order logic and PDL (with or without converse). However, in the context of trees one can show that MSO is actually expressively stronger. (This is why in the last theorem we have an implication and not an equivalence.) The example is the following. Take all homogeneously ternary branching trees of finite depth. Let S be the set of these trees decorated with boolean labels P and Q such that (a) P is true along a binary branching subtree and (b) Q is true at all leaves at which P is true. The following context–free grammar generates these trees.

$$\begin{array}{lcccc} \mathtt{P} \wedge \neg\mathtt{Q} & \to & \mathtt{P} & \mathtt{P} & \neg\mathtt{P} \\ \mathtt{P} \wedge \neg\mathtt{Q} & \to & \mathtt{P} & \neg\mathtt{P} & \mathtt{P} \\ \mathtt{P} \wedge \neg\mathtt{Q} & \to & \neg\mathtt{P} & \mathtt{P} & \mathtt{P} \end{array}$$

Now consider the projection of this set onto the language containing only Q. Then this set is not PDL–definable, but it is MSO–definable. The proof is given in [12].

The choice between the two language is therefore one of expressivity. It seems therefore to be just a matter of convenience which one we chooses. However, there is more to that. Let F be a finite set. An F–tree is a pair $\langle T, \ell\rangle$, where T is a tree domain and $\ell : T \to \wp(F)$ a function. Define an F–grammar to be a pair $\langle L, F, \Phi\rangle$, where F is a finite set, the set of *features*, L is an F–lexicon and Φ a (finite) set of theorems in some language for trees containing constants for the features from F. L consists of a finite set of pairs $\lambda = \langle a, E\rangle$, where a is a member of the alphabet and $E \subseteq F$ a set of features. We call $\chi_\lambda := \bigwedge_{f\in E} f \wedge \bigwedge_{f \in F-E} \neg f$ the *characteristic formula* of λ. The disjunction of all χ_λ where the first component of λ is a is called the *characteristic formula* of a. We say that *a has* the feature $f \in E$ but that it *lacks* the feature $g \in F - E$. Φ *accepts* a string $\vec{x} = a_0 \dots a_{n-1}$ iff there exists an F–tree T with yield $x_0 \dots x_{n-1}$ such that $T \models \Phi$ and for each $i < n$ the label of x_i satisfies the characteristic formula of a_i.

Let $\mathfrak{T} = \langle T, \ell\rangle$ be an F–tree and $G \subseteq F$. Then $\mathfrak{T}_G := \langle T, \ell_G\rangle$, where $\ell_G(x) := \ell(x) \cap G$ is a G–tree and called the *projection* of $\mathfrak{T}$ onto G. Now call a feature $f \in F$ *inessential* in a set S of F–trees if any tree in S_{F-f} is the projection of exactly one tree from S. This means that if we already know how the features distinct from f are distributed, we know how f itself is distributed. Call f *eliminable* if there is a formula φ using only the constants for features distinct from f such that

$$\mathsf{Th}\, S \models f \leftrightarrow \varphi$$

It follows that if f is eliminable it is also inessential, but the converse need not hold, as the above example shows. In MSO, however, every inessential feature is also eliminable. Therefore, whether or not an inessential feature is actually eliminable depends on the logical language. We may however also turn this around and use the strength of the logical language to define the complexity of languages. A language S (a set of F–trees) has complexity at most Θ, where Θ is some logical language, if (a) S is Θ–definable and (b) every inessential feature of S is Θ–eliminable. We have seen above that there are languages whose complexity is MSO but not PDL. It has been shown in [13] that the logic with four modal operators introduced earlier is strictly weaker than PDL and still stronger than boolean logic. (There is some maneuvering involved to adjust the definitions to the last two cases for we also have to satisfy (a). The interested reader is referred to the paper.)

6. Beyond Context–Free Languages

As we have seen, all logics presented here define some restricted class of languages, namely context–free languages, although they define them not via their strings but via their structure trees. In fact, for some years it has been the consensus among many linguists that natural languages are in fact context–free. Gpsg (Generalized Phrase Structure Grammar) was built on this assumption. However, in the mid eighties rather firm evidence appeared that there exist non–context–free languages. The evidence was based on data from Züritüütsch (see Shieber [25]). To appreciate these results, let us give some examples from three languages, English, German and Dutch. (The examples are translations of each other.)

(1E) that the children swim.
(2E) that Mary teaches the children to swim.
(3E) that Peter lets Mary teach the children to swim.

(1G) daß die Kinder schwimmen.
(2G) daß Maria die Kinder schwimmen lehrt.
(3G) daß Peter Maria die Kinder schwimmen lehren läßt.

(1D) dat de kinderen zwemmen.
(2D) dat Maria de kinderen laat zwemmen.
(3D) dat Peter Maria de kinderen laat leren zwemmen.

These examples can be continued ad libitum. The word order schemata of these languages are remarkably different. To see this more clearly, let us abbreviate the word order as follows. We write V_i for the ith verb and S_i for the subject of the ith verb. Then we have

English	:	$S_1\ V_1\ S_2\ V_2\ S_3\ V_3 \ldots$
German	:	$S_1\ S_2\ S_3 \ldots V_3\ V_2\ V_1$
Dutch	:	$S_1\ S_2\ S_3 \ldots V_1\ V_2\ V_3 \ldots$

We say that in German the dependencies between verbs and their subjects are nested while they are crossed in Dutch. Now, this suffices to show that Dutch cannot be strongly context–free since one cannot get the right constituent analysis using a context–free grammar. Yet, the data above do *not* show that no context–free grammar can generate the Dutch language as a string language. Namely, we can simply generate the crossed dependencies as nested ones. Yet, a proof that Dutch is not weakly context–free has been found by Huybregts, see [7]. He showed that there are verbs which require certain types of subjects, and so generating the dependencies nested rather than crossed would mean that one looses control over the restrictions on subjects. Shiebers proof on Züritüütsch uses a similar argument. Züritüütsch allows for the Dutch type word order and moreover there are raising verbs assigning dative case and those assigning accusative case. Since case marking is distinct, this proves the claim. [1]

It has of course always been felt that even if some languages are context–free as string languages, the corresponding set of analysis trees in many cases is not. An example for which this has been argued is German. (The standard analysis within transformational grammar is that all lower verbs raise to the highest verb to form a single complex verb.) Also, recently it has been shown that German is most likely not weakly context–free either (see [5]). These intuitions however were rather difficult to condense into real proofs, so one should say that linguists hoped rather than feared that one would be able to find non context–free–string languages. After their discovery matters became rather complex. On the one hand, transformational grammar has always been able to handle such languages, and no change was called for; similarly for LFG, which was built on non–cf structures anyway. However, surface based approaches had to adapt to the new situation. Several recipes have been tried. One was the introduction of string vectors to enhance the power of manipulation, another was the introduction of stacks of indices in the nonterminals, a third was the use of adjunction. These approaches are by no means identical; typically they lead to different structural analyses. For a survey of different approaches see [8] and [5].

We have earlier quoted results to that Government and Binding (GB) predicts that languages are strongly context–free if they have no unbounded head movement. A case in point is English. In that case it was possible to eliminate movement by a mechanism that is known as *gap–threading*, which has first been proposed in GPSG (see [4]). It allows to dispense with movement in favour of passing on some finite amount of information on the displaced element to its corresponding gap. If we want to analyse the whole theory however we cannot hope to eliminate head movement in a

[1] Notice that the verbs teach, let and so on on the one hand take an object (which in Züritüütsch can be either direct or indirect object) and on the other hand also an infinitive, whose subject is the object of the higher verb. This is why these noun phrases behave both as objects and as subjects.

similar way. Züritüütsch is known to be not context–free. Yet there is a straightforward analysis in GB using head movement. It follows that head movement adds some complexity to the theory. It is therefore indispensable to analyse in detail the structures created by GB.

7. An Outline of Government and Binding Theory

Let us now take a look at the standard GB theory. GB is part of the tradition called *transformational grammar*, which takes its name from the fact that it uses *transformations*. A transformation is an operation on trees, typically a unary operation. The basic idea has been that the structures of a language are created in two stages. In the first stage a so–called deep–structure is generated by a rather simple grammar (a context–free grammar of some sort). In the next stage, transformations successively operate and transform the deep–structure into the surface–structure. This surface structure is a tree and may contain phonetically empty elements. However, if all these things are stripped off we obtain the sentence in its official form. Here is an example from German. The verb *anvertrauen* has three arguments, a subject (S), a direct object (DO) and an indirect object (IO). At deep–structure these come in the order S–IO–DO, with the verb being last. From this basic order the surface sentence is derived by a series of transformations, which consist in simply moving around certain constituents. To be able to track the derivation the notation of traces with indices is used. A t is put at the place where the string originates, and give it an index. The displaced constituent also gets this index. Notice that only constituents may move and therefore they are the bearers of indices. Some constituents are highlighted by brackets.

der König seinem Minister diese Geheimnis anvertraute.	(deep–structure)
[dieses Geheimnis]$_1$ der König seinem Minister t_1 anvertraute.	
vertraute$_2$ [dieses Geheimnis]$_1$ der König seinem Minister t_1 an t_2.	
[der König]t_3 vertraute$_2$ [dieses Geheimnis]$_1$ t_3 seinem Minister t_1 an t_2.	(surface–structure)
der König vertraute dieses Geheimnis seinem Minister an.	(phonetic form)
The king confided this secret to his minister.	

This model has been reformed a number of times. In GB, it is assumed that there are four so–called levels of representation, D–structure (the former deep–structure), S–structure (the former surface–structure), LF (logical form) and PF (phonetic form). A derivation is a pair $\partial = \langle \vec{\mathfrak{S}}, j\rangle$ where $\vec{\mathfrak{S}} = \langle \mathfrak{S}_i : i < p\rangle$ is a sequence of structures and $j < p$. We say that $\mathfrak{S}_0$ is the *D–structure* of ∂, $\mathfrak{S}_j$ its *S–structure* or surface form and $\mathfrak{S}_{p-1}$ its logical form. The phonetic form is derived from the surface form by means of some processes that were rarely studied in GB. For our purposes we may think of the PF simply as the string associated with the surface form (together will all empty elements deleted). Each $\mathfrak{S}_{i+1}$, $i < p-1$, is derived from $\mathfrak{S}_i$ by means of applying the rule Move–α. We will specify its action later. It is assumed that for each level there is a set of principles that defines well–formedness at this level. In fact, a level of representation is one at which such conditions are operative beyond the mere fact that the sequence is a derivation. To give an example, at S–structure each noun phrase must have case. This need not hold at D–structure. So the situation is like this. We start with a structure $\mathfrak{S}_0$. For it to be a legitimate start, we must satisfy the conditions of D–structure. Next we successively apply Move–α. If the conditions for S–structure are met for $\mathfrak{S}_i$, we may call the structure the S–structure and derive the phonetic form from it. (Also PF has a set of well–formedness conditions associated for it.) After that we continue to apply Move–α until the conditions of LF are met. If we succeed, we have a well–formed derivation.

$$\begin{tikzcd}
 & & & & \mathrm{SS} \arrow[d] & & & \\
\mathrm{DS} \arrow[r] & \mathfrak{S}_1 \arrow[r, squiggly] & \mathfrak{S}_2 \arrow[r, squiggly] & \mathfrak{S}_3 \cdots \arrow[r, squiggly] & \mathfrak{S}_i \arrow[r, squiggly] \arrow[dr] & \mathfrak{S}_{i+1} \cdots \arrow[r, squiggly] & \mathfrak{S}_j & \mathrm{LF} \arrow[l] \\
 & & & & \mathrm{PF} \arrow[r] & \vec{x} & &
\end{tikzcd}$$

Now, what is this rule of Move–α? It consists in taking some constituent of type X in the tree, and moving it to some other place, and coindexing the two elements. This is illustrated in Figure 2. [2] By a conjuration of the various principles at the levels of representation it turns out that the set of places to which the constituent X can be moved as well as the set of constituents that are eligible for movement in the first place are quite restricted. We will note here that the constituent $\mathtt{X}_i$ must in particular c–command t_i. [3] This is defined as follows.

Definition 7.1. *Let $\langle T, <\rangle$ be a tree and $x, y \in T$. x **c–commands** y iff x is the root or there is a node z immediately above x and $z \geq y$.*

Another condition, explained later, says that t_i must be *subjacent to* $\mathtt{X}_i$. The use of the term *movement* is actually inaccurate. What happens is rather complex. At the place where X has been we insert a so–called *trace*, which gets an index that is identical to the index of X. (So, we need to assume that constituents get equipped with an index at D–structure. But there are other solutions.) In the received terminology we say that there is a *chain–link* between the new X and the trace just inserted, and we call $\mathtt{X}_i$ the *antecedent* of the trace t_i, and the trace is in some circumstances also referred to as the *gap* with *filler* $\mathtt{X}_i$. Of course, $\mathtt{X}_i$ may be displaced from its new position as well, so that we get a series of connected chain links for a single D–structure constituent, called a *chain*. Notice that when $\mathtt{X}_i$ is moved to a new position, it leaves a trace t'_i of identical index, which now c–commands the original trace t_i. [4]

8. Multidominance Structures

For many linguists of different background it has always been an alternative to view movement simply as adding another daughter link. In the special jargon one speaks of *structure sharing*. Consider the following configuration, where Y is immediately above X, which has been displaced and received some index i. Let Z immediately dominate the trace of X, t_i.

$$\ldots [\ldots \mathtt{X}_i \ldots [\ldots t_i \ldots]_{\mathtt{Z}} \ \ldots]_{\mathtt{Y}} \ \ldots$$

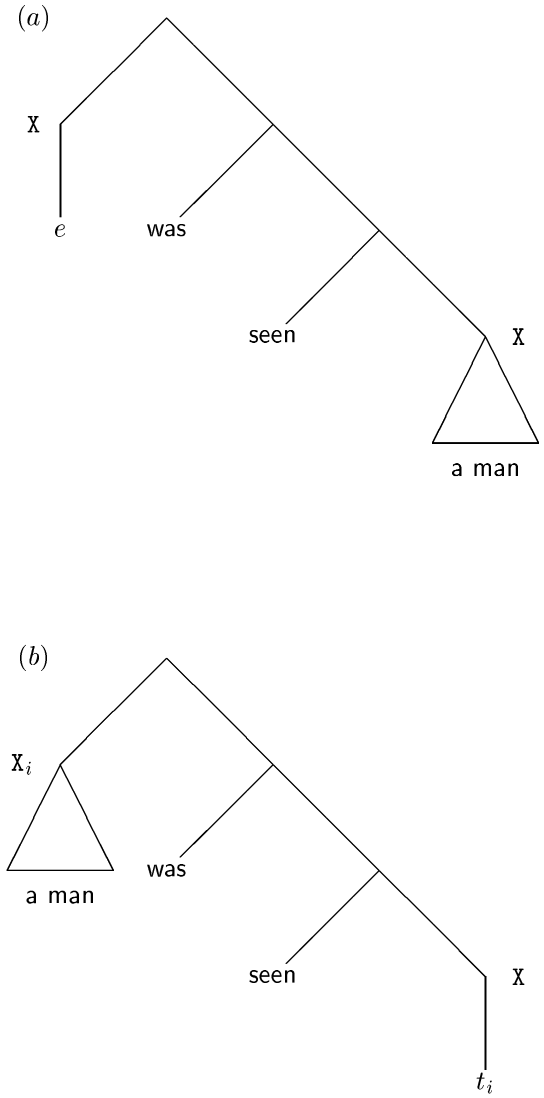
Then we assume that $\mathtt{X}_i$ and t_i are represented by one and the same structure (isomorphic to X), which is immediately below Y and Z. We say that Y and Z *share* the structure X. For example, rather than proceeding from (a) to (b) as in Figure 2 we now proceed from (a) to (c), as shown in Figure 3. In HPSG this idea has been made the leading philosophy. Within transformational grammar this move has always been resisted, for reasons that are actually not so clear. For some brief period in the 90ies Chomsky has revived an older idea that what we analyse as movement is actually a combination of copying and deletion, though the deletion is done at some level of

[2]Actually, in the linguistic literature one talks of *the* constituent X when one actually means a particular constituent of category X. We will also use this kind of talk here hoping that it causes no confusion.

[3]Talk of $\mathtt{X}_i$ as if it uniquely identifies a constituent is justified in this context, since there is always only one full constituent with a given index.

[4]In the standard literature, one uses notations such as t'_i, t''_i etc to distinguish the various traces of the same constituent $\mathtt{X}_i$. However, the actual representation just contains occurrences of t_i at different places.

FIGURE 2. A Movement Step



representation rather than immediately after the copying. Hence, we rather generate the following structure

$$\ldots[\ldots\mathtt{X}\ldots[\ldots\mathtt{X}\ldots]_{\mathsf{Z}}\ \ldots]_{\mathsf{Y}}\ \ldots$$

This is exemplified in Figure 4. The absence of indices is actually fully intentional. The corresponding structure is written out in a string as follows.

[[a man] [was [seen [$\underline{\text{a man}}$]]]].

A empty sequence is marked by underlining it. It was believed — wrongly, I think — that copying makes the formation of chains superfluous.

FIGURE 3. A Linking Step

(*c*)

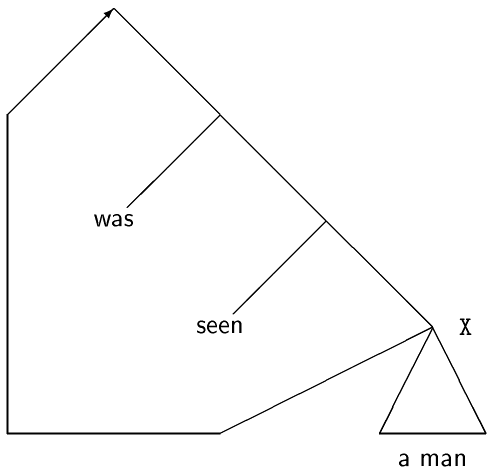
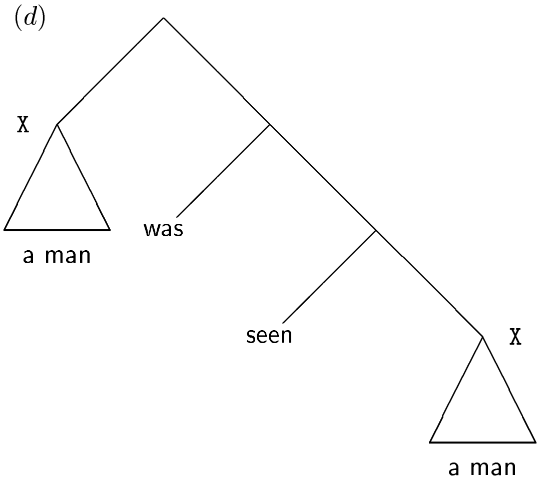


FIGURE 4. A Copying Step

(*d*)



Even if transformational grammarians have never proposed the use of reentrancy and would probably even reject them as legitimate syntactic objects, it is shown in [14] that the reentrant syntactic structures have certain advantages over the corresponding trace structures from a linguistic point of view. Here we will point out a different advantage of these structures. [22] has argued that the addition of indices or the ability to talk about isomorphic constituents require logical tools that easily yield undecidability. If one is careless enough, one ends up with a variant of third order logic over trees, a rather strong logical system indeed. However, there is no need to do things like that. We may simply use reentrant structures instead of the standard ones. In what is to follow in this paper we will actually show that with the help of reentrant structures we can give a full logical analysis of GB–theories without assuming too much power. Namely, a fragment of the monadic second order theory corresponding to **PDL** is with high probability already enough. This means in particular that we make no use of derivations at all (and this seems to be the reason why reentrant structures are so unpopular with transformational syntacticians). We will outline this approach here. Let $\langle T, \prec\rangle$ be a tree. A *pre–chain* is a set of constituents of the tree which is linearly ordered by c–command. A pre–chain is a *pre–trace chain* if all but one constituents is a trace. If we are given a syntactic structure in a derivation (see the previous section), then the chains are the sets of all constituents with the same index. We define a *pre–trace chain structure* (*preTCS*) to be a pair $\langle T, C\rangle$ where T is a tree and C a set of pre–trace chains subject to certain conditions that are of no immediate concern here.

A pair $\mathfrak{G} = \langle G, \prec\rangle$, where $\prec\subseteq G^2$ is called a *rooted acyclic graph* if the transitive closure of $\prec$ is acyclic and has a unique maximal element, called the *root*.

Definition 8.1. *A* ***pre–multidominance structure (preMDS)*** *is pair* $\mathfrak{M} = \langle M, \prec\rangle$ *such that (1)* $\mathfrak{M}$ *is a rooted acyclic graph, (2) for each* $x \in M$ *the set* $M(x) := \{y : x \prec y\}$ *is linearly ordered by* $\prec^+$*, which is denoted by* $<$*.* $\mathfrak{M}$ *is a* ***tree*** *if* $\sharp M(x) \leq 1$ *for all* $x \in M$*.*

Here, $\prec$ is the relation *is immediately dominated by* and $<$ the relation *is dominated by*. A pair $\langle x, y\rangle$ where $x \prec y$ is called a *link*. If y is the maximal element of $M(x)$ with respect to $<$, $\langle x, y\rangle$ is called the *surface link* of x. From a preMDS we can construct a movement structure as follows. Call a sequence $I \in M^+$ an *identifier* if its first element is the root and if $I = J; x; y; K$ for some sequences J, K then $y \prec x$ (ie $\langle y, x\rangle$ is a link). We write $J \prec I$ if $J = I; x$ for some x. It is easy to see that the set of identifiers forms a tree with $\prec$ the relation of immediate domination. (This is the standard unravelling technique.) Now call a *link* of I a pair $\langle x, y\rangle$ such that $I = J; y; x; K$ for some sequences J and K. I is a *surface identifier* if for all links of I are surface links. I is a *trace identifier* if it has the form $J; x; y$ where (a) $J; x$ is a surface identifier and (b) $\langle y, x\rangle$ is a non–surface link. Let $T(\mathfrak{M})$ be the set of surface identifiers and $\mathfrak{T}(\mathfrak{M}) := \langle T(\mathfrak{M}), \prec\rangle$. Then $\mathfrak{T}(\mathfrak{M})$ is a tree. Now say that I *identifies* x if x is the last element of I. Pick an element $x \in M$. $\mathfrak{C}(x)$ will consist of all constituents of $\mathfrak{T}(\mathfrak{M})$ that are of the form $\downarrow I$, where I identifies x. The pair $\langle \mathfrak{T}(\mathfrak{M}), \{\mathfrak{C}(x) : x \in M\}\rangle$ is a preTCS. One can also pass easily from a preTCS $\mathfrak{T} = \langle T, C\rangle$ to a preMDS. Let $x \approx y$ if there is a chain $\mathfrak{C}$ such that $\downarrow x, \downarrow y \in \mathfrak{C}$. Put $[x] := \{y : y \approx x\}$ and $[x] \prec [y]$ if there exists $x' \in [x]$ and $y' \in [y]$ such that $x' \prec y'$. Finally, $[T] := \{[x] : x \in T\}$ and $\mathfrak{M}(\mathfrak{T}) := \langle [T], \prec\rangle$. $\mathfrak{M}(\mathfrak{T})$ is a preMDS.

These constructions show that it is possible to pass back and forth between the original structures of GB (in the guise of preTCSs) and the reentrant structures, called preMDS. A derivation is a sequence of preTCSs whose initial member is a tree and in which each noninitial member is derived from the previous member by means of movement. The analogous operation on preMDSs is called a *link extension*.

Definition 8.2. *Let* $\mathfrak{M} = \langle M, \prec\rangle$ *be a preMDS. An element* x *is an* ***unmoved element*** *if* x *is the root or* $M(x) = \{z\}$*, where* z *is an unmoved element.*

Definition 8.3. *Let* $\mathfrak{M} = \langle M, \prec_M\rangle$ *and* $\mathfrak{N} = \langle N, \prec_N\rangle$ *be preTCSs.* $\mathfrak{N}$ *is called a* ***1–step link extension of*** $\mathfrak{M}$ *if* $N = M$ *and* $\prec_N = \prec_M \cup \{\langle x, y\rangle\}$*, where* $\langle x, y\rangle \notin \prec_M$ *and* y *is an unmoved element. A* ***derivation*** *of preMDSs is a sequence* $\langle \mathfrak{M}_i : i < p\rangle$ *such that* $\mathfrak{M}_0$ *is a tree and* $\mathfrak{M}_{i+1}$ *(*$i < p - 1$*) is a 1–step link extension of* $\mathfrak{M}_i$*.*

Lemma 8.4. *Let* $\mathfrak{M} = \langle M, \prec\rangle$ *be a preMDS. Then there is a derivation* $\langle \mathfrak{M}_i : i < p\rangle$ *such that* $\mathfrak{M} = \mathfrak{M}_{p-1}$ *and* $\langle M_{i+1}, <_{i+1}\rangle = \langle M_i, <_i\rangle$ *for all* $i < p - 1$*. In particular,* $\langle M, <\rangle$ *is a tree.*

It turns out that the operations defined earlier not only translate preTCSs into preMDSs and back but they translate derivations into derivations:

Theorem 8.5. *For each derivation on preTCS the analogous sequence of preMDSs is a derivation. Conversely, for every derivation of preMDSs the corresponding sequence of preTCSs is a derivation. This correspondence is bijective.*

This theorem allows to switch freely between these two representations. Let us say that we have simplified things at least at three points: (A) we have omitted the level of logical form, (B) the structures are unordered and (C) there is no adjunction, only substitution. All this can be remedied. However, the details are lengthy and unrevealing. The interested reader is referred to the abovementioned paper. We will continue with this simplified model.

9. Axiomatizing Government and Binding Theories

Now that we have introduced the new structures let us attack the question whether it is possible to write for some given GB theory an equivalent logical theory over preMDSs. The main obstacle is the notion of a derivation. We wish that the logical theory says directly, given a structure $\mathfrak{M}$, whether or not it is a legitimate structure without taking recourse to the derivational history. That this is at all possible can be motivated from the fact that the logical form (which in our simplified model is already the S–structure) contains a record of the derivational history in form of the traces (alias chains). Although the derivational history cannot be uniquely recovered, the chains alone give as much information as we need. For example, we can easily see how long the derivation was. (Simply count the number of traces.) Yet, the details are still quite involved. We will begin by noting that the class of preMDS is $\mathbf{PDL}^{\smile}$–definable in a language using one primitive program, *up*. We write *down* for its dual, $up^{\smile}$. Furthere, write $\Diamond_{\uparrow}$ for $\langle up\rangle$, $\Diamond_{\downarrow}$ for $\langle down\rangle$, $\Diamond_{\uparrow}^{+}$ for $\langle up^{+}\rangle$ and $\Diamond_{\downarrow}^{+}$ for $\langle down^{+}\rangle$.

$$
\begin{array}{llll}
(a) & \Box_{\uparrow}^{+}(\Box_{\uparrow}^{+}p \rightarrow p) \rightarrow \Box_{\uparrow}^{+}p & (b) & \Box_{\downarrow}^{+}(\Box_{\downarrow}^{+}p \rightarrow p) \rightarrow \Box_{\downarrow}^{+}p \\
(c) & \Box_{\uparrow}^{+}\Diamond_{\uparrow}^{+}p \rightarrow \Diamond_{\uparrow}^{+}\Box_{\uparrow}^{+}p & (d) & \Diamond_{\uparrow}p \wedge \Diamond_{\uparrow}q \rightarrow \Diamond_{\uparrow}(p \wedge (q \vee \Box_{\downarrow}^{+}q \vee \Box_{\uparrow}^{+}q))
\end{array}
$$

(a) and (b) together force that the structures are finite and that *up* is irreflexive and cycle–free. The existence of a root is a consequence of (c). That $M(x)$ is linearly ordered by $<$ is the content of (d).

Next we wish to implement the various principles operative at the levels. The difficulty that we face here is that if φ is a condition for D–structures, we cannot check φ at S–structure, since the two structures are in general not isomorphic. The D–structure can be reconstructed as follows. Call a link $\langle x, y\rangle$ a *base link* if y is the lowest element of $M(x)$. The D–structure is that part of the S–structure that is constituted by the root and all base links. The idea is therefore to substitute the formula φ expressing a condition on D–structure by a formula $\varphi^{\bullet}$ such that the S–structure satisfies $\varphi^{\bullet}$ iff the D–structure satisfies φ. So, rather than using *up* and *down* by which we may follow any link we must use the relation $\prec^{\bullet} := \{\langle x, y\rangle : \langle x, y\rangle \text{ is a base link}\}$ and its converse. It is not possible to do that using some program construct, we have to introduce a new relation $up^{\bullet}$ into Π_0 together will the following axioms.

$$
\begin{array}{llll}
(e) & \langle up^{\bullet}\rangle p \rightarrow [up^{\bullet}]p & (f) & \langle up^{\bullet}\rangle p \rightarrow \Diamond_{\uparrow}p \\
(g) & \Diamond_{\uparrow}q \rightarrow \langle up^{\bullet}\rangle \Diamond_{\uparrow}^{*}q & &
\end{array}
$$

These axioms correspond to the following conditions. $up^{\bullet}$ is a partial function (by (e)), whose graph is contained in $\prec$ (by (f)) and which picks out the least element from $M(x)$ for given x (by (g)). We now enrich the preMDS by a second accessibility relation, $\prec^{\bullet}$, which encodes the D–structure. $\mathfrak{M}$ is a tree iff $\prec^{\bullet} = \prec$. Therefore, $\mathfrak{M} = \langle M, \prec_M, \prec^{\bullet}_M\rangle$ is derived from $\mathfrak{N} = \langle N, \prec_M, \prec^{\bullet}_M\rangle$ iff $M = N$, $\prec^{\bullet}_M = \prec^{\bullet}_N$ and $\prec_M \supseteq \prec_N$ (cf Lemma 8.4). Now let φ be a formula. Then let $\varphi^{\bullet}$ be the result of replacing *up* by $up^{\bullet}$.

Lemma 9.1. *Let $\mathfrak{M} = \langle M, \prec, \prec^{\bullet}\rangle$ be a preMDS and $\mathfrak{N}$ its D–structure. Then $\mathfrak{M} \models \varphi^{\bullet}$ iff $\mathfrak{N} \models \varphi$.*

The proof straightforward. $\varphi^{\bullet}$ uses only relations defined over $up^{\bullet}$ and $down^{\bullet}$. So $\langle M, \prec, \prec^{\bullet}\rangle \models \varphi^{\bullet}$ iff $\langle M, \prec^{\bullet}\rangle \models \varphi^{\bullet}$ iff $\mathfrak{N} \models \varphi^{\bullet}$. Notice that it is immaterial whether or not φ contains occurrences of *up*$^\bullet$ and *down*$^\bullet$, though by construction it does not. Hence, on condition that the D–structure conditions are $\mathbf{PDL}^{\smile}$–definable, we can define them also as S–structure conditions in an extended language.

Now for the last part, the well–formedness of the derivation. The main condition is the following. If $\mathtt{X}_i$ is the antecedent of t_i then

1. $\mathtt{X}_i$ c–commands t_i but does not dominate t_i and
2. t_i is subjacent to $\mathtt{X}_i$.

Our structures are such that condition (1) is automatically fulfilled. Condition (2) must be enforced by some axiom, however. It is defined as follows. There is a notion of a *barrier*, which can be defined by means of some constant formula b. We will not go into the details of that here. Let u be the node immediately dominating $\mathtt{X}_i$. Then t_i is *subjacent* to $\mathtt{X}_i$ if the open interval $]t_i, u[$ in the tree contains at most one barrier. This is the definition for trees. To be able to translate this into a condition on preMDSs, we must analyze the relationship between preMDSs and its derivations. The problem is that a given preMDS (or given a preTCS) there are alternative analyses. We may illustrate this with split–DPs in German. (We annotate each step with the type of movement, for those in the know.)

Diese Bücher hatte ich damals alle meinem Chef gegeben.	(phonetic form)
I had then given all these books to my boss.	
damals ich meinem Chef alle diese Bücher gegeben hatte.	(d–structure)
First Derivation:	
[alle diese Bücher]$_1$ damals ich meinem Chef t_1 gegeben hatte.	(Scrambling)
ich$_2$ [alle diese Bücher]$_1$ damals t_2 meinem Chef t_1 gegeben.	(Scrambling)
hatte$_3$ ich$_2$ [alle diese Bücher]$_1$ damals t_2 meinem Chef t_1 gegeben t_3.	(V2)
[diese Bücher]$_4$ hatte$_3$ ich$_2$ [alle t_4]$_1$ damals t_2 meinem Chef t_1 gegeben t_3.	(Topicalization)
Second Derivation:	
[diese Bücher]$_1$ damals ich meinem Chef [alle t_1] gegeben hatte.	(Topicalization)
[diese Bücher]$_1$ [alle t_1]$_2$ damals ich meinem Chef t_2 gegeben hatte.	(Scrambling)
[diese Bücher]$_1$ ich$_3$ [alle t_1]$_2$ damals t_3 meinem Chef t_2 gegeben hatte.	(Scrambling)
[diese Bücher]$_1$ hatte$_4$ ich$_3$ [alle t_1]$_2$ damals t_3 meinem Chef t_2 gegeben t_4.	(V2)

We see that up to renumbering the structures are the same. However, in the first derivation the phrase die Bücher first takes a free ride up inside the quantified phrase alle diese Bücher and then moves further up to the sentence initial position. In the second derivation, it moves in one step into the final position. One can see that the distance of the movement (in terms of nodes) is larger.

How is the difference between these derivations accounted for? The answer is not clear, in fact. There are two competing principles for derivations that select different derivations. The first is called *Freeze*. A derivation satisfies *Freeze* if no constituent is moved out of a derived constituent (ie one that has been moved before). The first derivation does not satisfy *Freeze*, but the second does. Another principle is that of *Shortest Steps*. It says that a derivation must the shortest possible movement steps. The first derivation satisfies *Shortest Steps*, but the second does not. There are structures which have derivations that satisfy neither principle. It can be shown that *Freeze*–derivations always choose the longest possible path. It turns out that these paths have rather unique properties that make them suitable for our purposes.

Let $x, y \in M$ and $x < y$. A *path* from x to y is a subset Π of M that has x as lowest element, y as maximal element and which is connected via $\prec$ and linearly ordered by $<$. The *length* of Π is its $\sharp\Pi - 1$. The *barrier count* is the number of barriers in $\Pi - \{x, y\}$.

Lemma 9.2. *Let $\mathfrak{M}$ be a preMDS, $x, y \in M$ and $x < y$. There is a unique longest path Λ from x to y. Moreover, every path from x to y is a subset of Λ.*

Proof. For a proof, notice that by Lemma 8.4, $\mathfrak{M}$ is s derived from a preMDS $\langle M, \prec_0\rangle$ that is also a tree. Hence, $\langle M, <\rangle$ is a tree and $< = \prec_0^+$. So, take $x < y$. Now set $\Lambda := [x, y]$. This set is linearly ordered by $<$. It is also maximal. We have to show that it is connected via $\prec$. But $< = \prec_0^+$ by Lemma 8.4. So, Λ is $\prec_0$–connected and therefore also $\prec$–connected, since $\prec_0 \subseteq \prec$. □

Now consider an element x. Let $y \in M(x)$ and let z be the minimal element of $M(x)$ such that $z > y$. Then a path Π from y to z is called *movement link of x at y*. Π is called *maximal* if it is of maximal length, and *minimal* if it is of minimal length. Now, informally stated, the situation is as follows. Given a preMDS $\mathfrak{M}$, a *Freeze* derivation satisfies *Subjacency* iff all maximal movement links have barrier count ≤ 1. In this reformulation we can modally define those preMDSs which are generated from trees by means of *Freeze* derivations respecting *Subjacency*.

$$U(p) := \Box_{\downarrow}^{*}\Box_{\uparrow}^{*}(p \to \Box_{\downarrow}^{+}\neg p \wedge \Box_{\uparrow}^{+}\neg p)$$

$U(p)$ holds at x iff the set of points satisfying p is an antichain.

$$K(p, q, r) := p \wedge U(p) \wedge U(q) \wedge U(r) \wedge \Diamond_{\uparrow} q \wedge \Diamond_{\downarrow}(r \wedge \Diamond_{\uparrow} q) \wedge \Box_{\uparrow}^{*}(\Diamond_{\uparrow}^{+} q \to \neg\Diamond_{\downarrow} r)$$

$K(p, q, r)$ holds at x if there are y and z such that $y > x$, $x \in M(z)$, q holds at y, r at z, the set of points for p, q and r are antichains, and y is the smallest element $> x$ which is also in $M(z)$.

$$\begin{array}{rcl} \nabla & := & up; (\neg \mathsf{b}?; up)^{*}; \mathsf{b}? \\ \sigma_F & := & K(p, q, r) \to [\nabla^2]\Diamond_{\uparrow}^{+} q \end{array}$$

Theorem 9.3. *$\mathfrak{M} \models \sigma_F$ iff it has a Freeze–derivation satisfying Subjacency.*

10. Shortest Steps

Now, we wish to extend this result to *Shortest Steps* derivations. Our candidate is σ_S. It says that for each element we can design a sequence of minimal movement steps satisfying subjacency.

$$\sigma_S := K(p, q, r) \to \langle\nabla^2\rangle\Diamond_{\uparrow}^{+} q$$

Obviously, if there exists a unique minimal path then this formulation would suffice. For then a *Shortest Steps* derivation satisfies *Subjacency* iff all minimal movement links have barrier count ≤ 1 iff there always exists *some* path from x to y which has barrier count ≤ 1. However, matters are more difficult here. For there need *not* exist a unique smallest path. A conterexample is as follows. Let M be the set natural numbers < 5 and $\prec = \{\langle 0, 1\rangle, \langle 1, 2\rangle, \langle 2, 3\rangle, \langle 3, 4\rangle, \langle 0, 3\rangle, \langle 1, 4\rangle\}$. Then $\prec^+ = <$. Let $x := 0$ and $y := 4$. The longest path from x to y is the whole interval. However, there exist two minimal paths: $\{0, 3, 4\}$ and $\{0, 1, 4\}$. Moreover, there exist structures in which the minimal path is actually not eligible, that is, it does not correspond to an actual derivation. This is exemplified in Figure 5. In this derivation it is the constituent marked `A` that moves first. After that `B` moves and in the third step `C` jumps out of the landing site of `A`. However, the shortest path would definitely be the one from `D`. However, `D` is empty, since its content has been moved out of `B` at the very first step. So, `D` is not eligible. The problem that we see here is that one cannot have all elements optimize their journey independently.

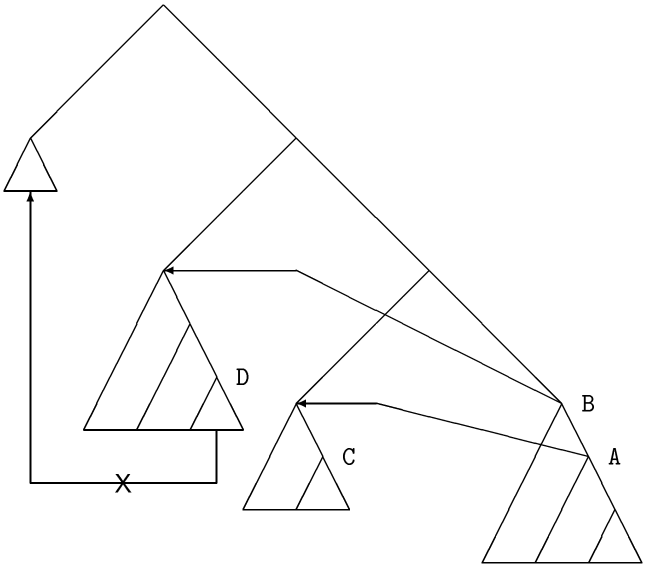
What can be done? In [14] it is proposed that a path is a shortest path if its links are the highest possible.

Definition 10.1. *Let $\langle u, v\rangle$ be a link. $\langle u, v\rangle$ is maximal for y if there is no $v' > v$ such that $\langle u, v'\rangle$ is a link and $v' \leq y$. Let x and y be points such that $x < y$ and Π a path from x to y. Π is called* ***contracted*** *if all links are maximal for y.*

Lemma 10.2. *For each x and y such that $x < y$ there exists exactly one contracted path from x to y.*

The contracted paths are not necessarily the shortest paths and this is the reason that we cannot employ a definition of subjacency using σ_S. Notice that in order to define the notion of maximality we need to compare the link $\langle u, v\rangle$ with an alternative link $\langle u, v'\rangle$. Such definitions

FIGURE 5. The Shortest Steps Paradox



are generally beyond the capacity of **PDL**. Hence, in general we cannot define *Shortest Steps* derivations. However, structures in which contracted paths are not minimal are rather contrived. It would therefore seem that if there are additional principles that conjure to exclude such pathological structures, we can actually define *Subjacency* for derivations satisfying *Shortest Steps*.

We note that while *Freeze* is something that can be checked on a derivation by checking each step separately, *Shortest Steps* can only be checked by comparing alternative derivations from the same structure. For example, suppose that in the second derivation of the preceding section there is no other place where to move the phrase die Bücher. Then this step is the shortest one available at that moment. This derivation would therefore satisfy *Shortest Steps*. The crux is however that there is an alternative derivation that saves the phrase die Bücher to move that long. *Shortest Steps* is therefore a transderivational economy principle in the sense of the Minimalist Program.

11. Decidability Issues

The use of formal analytic tools can lie among other in nontrivial technical results, such as decidability results. Indeed, there are certain rather satisfying results concerning decidability of systems. However, the balance between expressivity on the one hand and the wish to maintain decidability is rather difficult to keep. A very early result is Peters and Ritchie [19], who have shown that transformational grammars with a regular grammar generating the deep–structures using very elementary transformations can generate any given recursively enumerable languages. This has been seen as a negative fact, since the claim that certain languages are generable by transformational grammars is then nearly vacuous. We will argue below that GB is different in this respect, though probably too weak to generate all himan languages.

The optimal case is provided by regular languages.

Theorem 11.1. *It is generally decidably, given two regular grammars G and H, whether $L(G) = L(H)$.*

The proof is not difficult. It uses certain constructions on automata. This property is lost already when we move to context–free languages.

Theorem 11.2. *It is in general undecidable, given two context–free grammars G and H, whether $L(G) = L(H)$.*

In fact, there is also a rather powerful theorem on context–free languages, which is as follows. A *homomorphism* of languages is a map $h : A^* \to B^*$, where A and B are the respective alphabets of the languages, which commutes with concatenation. Moreover, $h(\varepsilon) = \varepsilon$.

Theorem 11.3. *Any recursively enumerable language is the homomorphic image of an intersection of two context–free languages.*

There are linguistic theories which assume that sentences have two different analysis trees. Examples are Lexical Functional Grammar and Autolexical Grammar; for the latter see [24]. Let us call them *dual analysis grammars.* If these analysis trees are individually generated by context–free grammars G and H, respectively, then call such a grammar *context–free.* If we additionally assume that there may be empty categories, then any recursively enumerable language possesses a context–free dual analysis grammar. [5] This shows how powerful certain theories may become. However, we have from [20] the following result.

Theorem 11.4 (Rabin)**.** *The MSO theory of any projection of a context–free set of trees is decidable.*

It follows that if we have just any two finitely axiomatized theories of projections of context–free sets, we can decide whether or not they define the same set. So, while on the string level matters are hopeless, on the structure level they are again as nice as we can hope for. One can also show that for two context–free grammars it is decidable whether they generated the same parenthesized strings (this is due to McNaughton [16]).

Now let us move into the non context free case. Here matters are still unclear. [22] warns us that unlimited use of higher order devices can yield undecidability. However, the structures that we have defined are rather constrained and we wish the launch the following

Conjecture 11.5. *The MSO theory of the set of preMDSs is decidable. In particular the* ***PDL****–theory of the set of preMDSs is decidable.*

It would follow, we claim, that the theory of the full structures for GB is decidable (though we have not given evidence for that claim let alone defined these structures). It seems that for all intents and purposes the **PDL**–fragment is enough. Now, this is only the minimal logic. We still have to account for theories of sets of preMDS corresponding to certain possible human languages. However, if one follows the philosophy of GB, there is little variation to be expected, which mainly consist in alternative variants of defining barriers, bounding nodes etc and of course the lexicon. It seems therefore that the minimal logic is a generic case from which a general result can be established.

Be matters as they are, there are reasons to believe that GB in this formulation is not strong enough to generate all possible human languages. The evidence is based on the contention that GB theories imply that human languages generate semilinear sets (see [6] for a definition and details). Space does not permit us to give evidence for this contention. Jens Michaelis [17] has recently proved this claim for Stabler's formalization of the Minimalist Program (see [26]). This gives some substance for our claim. Now, [5] as well as [18] give evidence that human languages are not necessarily semilinear. If we are right, then it follows that GB is restrictive enough to be decidable in all its parametric variants, but too weak to generate structures for all human languages.

References

[1] J. R. Büchi. Weak Second–Order Arithmetic and Finite Automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 6:66 – 92, 1960.

[5] This is not to say that Sadock assumes empty categories. If not, then a context free dual analysis grammar defines a language recognizable in cubic time. This is a much more restricted class of languages.

[2] Noam Chomsky. Context–free grammars and pushdown storage. *MIT Research Laboratory of Electronics Quarterly Progress Report*, 65, 1962.
[3] John Doner. Tree Acceptors and Some of Their Applications. *Journal of Computer and System Sciences*, 4:406 – 451, 1970.
[4] Gerald Gazdar, Ewan Klein, Geoffrey Pullum, and Ivan Sag. *Generalized Phrase Structure Grammar*. Blackwell, Oxford, 1985.
[5] Annius Groenink. *Surface without Structure. Word order and tractability issues in natural language processing.* PhD thesis, Utrecht University, 1997.
[6] Michael A. Harrison. *Introduction to Formal Language Theory.* Addison Wesley, Reading (Mass.), 1978.
[7] R. Huybregts. Overlapping Dependencies in Dutch. *Utrecht Working Papers in Linguistics*, 1:3 – 40, 1984.
[8] Aravind Joshi, K. Vijay-Shanker, and David Weir. The Convergence of Mildly Context–Free Grammar Formalisms. In Peter Sells, Stuart M. Shieber, and Thomas Wasow, editors, *Foundational Issues in Natural Language Processing*, pages 31 – 81. MIT–Press, 1991.
[9] Steven C. Kleene. Representation of events in nerve nets. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3 – 40. Princeton University Press, 1956.
[10] Jan Koster. *Domains and Dynasties: the Radical Autonomy of Syntax.* Foris, Dordrecht, 1986.
[11] Marcus Kracht. Syntactic Codes and Grammar Refinement. *Journal of Logic, Language and Information*, 1995.
[12] Marcus Kracht. Tools and Techniques in Modal Logic. Habilitationsschrift, 1996.
[13] Marcus Kracht. Inessential Features. In Christian Retoré, editor, *Logical Aspects of Computational Linguistics*, number 1328 in Lecture Notes in Artificial Intelligence, pages 43 – 62. Springer, 1997.
[14] Marcus Kracht. Syntax in Chains. unpublished, 1998.
[15] Maria R. Manzini. *Locality – A Theory and Some of Its Empirical Consequences.* Number 19 in Linguistic Inquiry Monographs. MIT Press, 1992.
[16] Robert McNaughton. Parenthesis grammars. *Journal of the Association for Computing Machinery*, 14:490 – 500, 1967.
[17] Jens Michaelis. Derivational minimalism is mildly context–sensitive. manuscript to be published, 1998.
[18] Jens Michaelis and Marcus Kracht. Semilinearity as a syntactic invariant. In Christian Retoré, editor, *Logical Aspects of Computational Linguistics (LACL '96)*, number 1328 in Lecture Notes in Computer Science, pages 329 – 345, Heidelberg, 1997. Springer.
[19] Stanley Peters and Graeme Ritchie. On the generative power of transformational grammars. *Information Sciences*, 6:49 – 84, 1973.
[20] Michael O. Rabin. Decidability of Second–Order Theories and Automata on Infinite Trees. *Transactions of American Mathematical Society*, 141:1 – 35, 1969.
[21] Luigi Rizzi. *Relativized Minimality.* MIT Press, Boston (Mass.), 1990.
[22] James Rogers. *Studies in the Logic of Trees with Applications to Grammar Formalisms.* PhD thesis, Department of Computer and Information Sciences, University of Delaware, 1994.
[23] James Rogers. What Does a Grammar Formalism Say About a Language? unpublished manuscript, 1995.
[24] Jerrold M. Sadock. *Autolexical Syntax. A Theory of Parallel Grammatical Representations.* Studies in Contemporary Linguistics. University of Chicago Press, Chicago, 1991.
[25] Stuart Shieber. Evidence against the Context–Freeness of Natural Languages. *Linguistics and Philosophy*, 8:333 – 343, 1985.
[26] Edward P. Stabler. Derivational minimalism. In Christian Retoré, editor, *Proceedings of LACL 96*, number 1328 in Lecture Notes in Artificial Intelligence, Heidelberg, 1997. Springer.
[27] J. W. Thatcher. Characterizing Derivation Trees of Context–Free Grammars through a Generalization of Finite Automata. *Journal of Computer and System Sciences*, 1:317 – 322, 1967.

II. Mathematisches Institut, Freie Universität Berlin, Arnimallee 3, D – 14195 Berlin, kracht@math.fu-berlin.de