Chapter 9

Graph Properties of HPSG Feature Structures

STEPHAN KEPSER AND UWE MÖNNICH

Dept. of Linguistics, University of Tübingen, Germany
E-mail: {kepser,um}@sfs.uni-tuebingen.de

ABSTRACT. A general principle for grammars for languages is that they must use *finite* means to produce the infinitely many structures of the languages. When the term *production* is understood algebraically, there are at least two notions of finiteness involved. Firstly in each set of structures each structure is generated, i.e., the result of *finitely* many applications of operators. Secondly, the set of operators as such is also *finite*. For algebraic theories, this is eo ipso the case. But there are linguistic frameworks that are not generating but rather licensing linguistic structures such as Head-Driven Phrase Structure Grammar or GB-Theory. Within these, structures are not generated. Rather only general properties of admissible structures are stated. But these structures must be provided somehow. Thus one may ask if there are algebraic approaches to licensing theories.

We show here that HPSG is non-algebraic in the following sense. There is no finite family of operations such that the members of every finite set of feature structures can be constructed by this finite family of operations. We also show that HPSG feature structures are abstractly recognisable in the sense that for every set of finite HPSG feature structures there exists a (many-sorted) algebra with *finite* sort sets of infinitely many operations such that every feature structure in that set is recognised by the algebra (seen as an abstract automaton). In other words, the first finiteness demand can be fulfilled, the second one cannot.

9.1 Introduction

Formal frameworks for linguistic theories like tree adjoining grammars, minimalist grammars, and context-free tree grammars belong to the generative-enumerative paradigm (Pullum and Scholz, 2001). This paradigm can be considered as a branch of applied recursive functions theory. Languages, under this view, consist of a set of finite concrete structures that are enumerable by an algorithmic device. The Chomsky hierarchy and its refinements is one of the success stories of this approach towards a general syntactic meta-theory.

Recent years have seen the emergence of a competing paradigm which is not so much inspired by a general theory of algorithms but by methods developed within finite model theory (Ebbinghaus and Flum, 1995). According to this approach languages are to be viewed as sets of (finite) relational structures that satisfy a family of constraints which are stated in a suitable logical specification language.

Proceedings of Formal Grammar 2003
G. Jäger, P. Monachesi, G. Penn & S. Wintner (editors).

Grammatical structures are not the result of an enumerative process, but they comprise those structures that satisfy all the constraints of a language. Prominent incarnations of this paradigm are the Government and Binding Theory (Chomsky, 1981) and Head-driven Phrase Structure Grammar (HPSG (Pollard and Sag, 1987, 1994)).

What has not been noticed before is the fact that the difference between the generative and the model-theoretic licensing paradigm is parallelled by a concomitant distinction of data structures. To put it in a nutshell: whereas the notion of an initial abstract data structure is fundamental for the generative-enumerative framework, the dual notion of a final abstract object structure plays a similar role within the model-theoretic framework. The definition of an initial abstract data structure specifies in which ways elements of that structure may be generated. The central concepts of this type of data are those of an algebra of a signature, of a congruence relation and of induction. Familiar examples are provided by strings and trees. The dual type of data is characterised by the corresponding notions of coalgebra, bisimulation relation and coinduction. The elements of a final object structure are not described in terms of a representation through their recursive buildup, but via the attributes and methods that characterise their behaviour. The most prominent example for such a structure is that of a labeled graph.

The difference between these two types of data structures is the reason for the association of strings and trees with the generative-enumerative paradigm and of labeled graphs with the model-theoretic paradigm of syntactic frameworks. Trees can be uniquely characterised by specifying their hierarchical structure, they can be enumerated by means of a production system and they can be fed into an abstract automaton. For graphs no suitable notion of finite automaton has been proposed. Similarly, graph grammars do not generate their output in a way that is patterned by an independently given hierarchical structure which would be intrinsic to this data type.

The notion of generation often entails the existence of a finite set of operators such that every element of the structure under consideration is the result of finitely many applications of these operations on a finite set of constants (nullary operations). One way of interpreting this technical statement is to say that a grammar constructs a language with *finite* means, where there are two aspects of finiteness to be distinguished. Firstly, each structure can be constructed by finitely many applications of operations (this is the notion of generation in algebra). Secondly, there are only finitely many different operators. These properties may be desirable for licensing theories as well. The fact that the data structures underlying licensing theories are graphs does not as such preclude such a possibility. Bauderon and Courcelle (1987) present an algebraic approach to graphs by defining a (generally infinite) set of graph operations such that every finite graph can be constructed by finitely many applications of these graph operations. Terms consisting of graph operation symbols are graph *expressions*, they describe how a graph is constructed. The value of such an expression is of course the graph the construction of which it describes. Thus there is a constructive, algebraic approach to graphs. The important question is then, whether the set of graph operators needed for a family of grammatically well-formed structures is finite or not.

For the particular licensing theory HPSG we will show here that the classes of finite graphs defined by a finite HPSG signature and grammar are indeed such that they cannot be generated by a finite set of graph operations. The set of graph expressions for finite HPSG graphs can in general not be generated by a regular or even context-free tree grammar. This result is independent of the particular logic chosen for HPSG. It just relies on standard assumptions on HPSG signatures and grammars, as they are spelled out, e.g., by Pollard and Sag (1994).

We also show that HPSG is abstractly recognisable. This means that for every set of HPSG feature structures defined by a grammar there exists a (many-sorted) algebra over the possibly infinite set of graph operations where all sort sets are *finite*. This algebra can be seen as an abstract automaton. The automaton is abstract, because there are infinitely many graph operations and hence infinitely many different sorts. In other words, the states of the automaton are sorted. For each sort, there are only finitely many states, but the set of all states of all sorts is infinite. This is why the automaton cannot really be used for recognition or generation. The notion *recognisable* remains abstract. But at least this finiteness property can be established for HPSG.[1]

9.2 HPSG-Style Feature Structures

The data structures underlying HPSG are so-called feature structures. The probably simplest approach to them, and the one we will follow here, is to regard them as relational structures. On this view, sorts are unary predicates and features are binary functional predicates. HPSG feature structures are required to be *totally well-typed* and *sort resolved*. Sort-resolvedness, demanding that the sorts partition the universe, is of no particular relevance here. Well-typedness restricts the admissible correlations between sorts and features. Each sort is correlated with a set of admissible features. And for each such feature there is an indication listing the admissible set of sorts on the target node. *Total* well-typedness additionally requires each admissible feature to be present. Thus a signature for HPSG is a triple $\Delta = (\mathcal{S}, \mathcal{F}, A)$ where $\mathcal{S}$ is a set of sorts (unary predicate symbols), $\mathcal{F}$ is a set of features (binary predicate symbols), and $A : \mathcal{S} \times \mathcal{F} \to \wp(\mathcal{S})$ is an *appropriateness function*. We will restrict our attention to *finite* signatures, i.e., signatures with $\mathcal{S}$ and $\mathcal{F}$ being finite sets. The restrictions imposed by a finite HPSG signature can be expressed as a first-order sentence $\mathcal{A}_\Delta$, as was shown by Aldag (1997) in his

[1] Note that the fact that each single finite graph can be constructed by finitely many applications of graph operations does in no way imply that every (possibly infinite) *set* of feature structures is abstractly recognisable.

master's thesis:

$$\begin{aligned}
&\forall x \bigvee_{S\in\mathcal{S}} S(x) \\
&\wedge \bigwedge_{S_1,S_2\in\mathcal{S},S_1\neq S_2} \forall x \neg(S_1(x)\wedge S_2(x)) \\
&\wedge \bigwedge_{F\in\mathcal{F}} \forall x,y,z((F(x,y)\wedge F(x,z)) \rightarrow y=z) \\
&\wedge \bigwedge_{S\in\mathcal{S},F\in\mathcal{F},A(S,F)\neq\emptyset} \forall x(S(x)\rightarrow \exists y\, F(x,y)\wedge \bigvee_{S'\in A(S,F)} S'(y)) \\
&\wedge \bigwedge_{S\in\mathcal{S},F\in\mathcal{F},A(S,F)=\emptyset} \forall x(S(x)\rightarrow \neg\exists y\, F(x,y))
\end{aligned}$$

One would now expect that a grammar is a first-order sentence over the signature. But a typical HPSG principle is somewhat simpler. It is a boolean combination of path equality expressions and sort statements. A path equality expression is a formula of the following kind:

$$\begin{aligned}
\exists y_1,\dots y_n \exists z_1,\dots z_k\ \ & F_1(x,y_1)\wedge\cdots\wedge F_n(y_{n-1},y_n)\wedge \\
& G_1(x,z_1)\wedge\cdots\wedge G_k(z_{k-1},z_k)\wedge y_n=z_k
\end{aligned}$$

where $F_1,\dots,F_n,G_1,\dots,G_k\in\mathcal{F}$. Such a formula stipulates that the two feature paths $F_1\dots F_n$ and $G_1\dots G_k$ lead to the same node. A sort statement is a formula of the following kind:

$$\exists x_1,\dots x_n\ F(x,x_1)\wedge\cdots\wedge F(x_{n-1},x_n)\wedge S(x_n)$$

where $F_1,\dots,F_n\in\mathcal{F}$ and $S\in\mathcal{S}$. It states that the node at the end of the path $F_1\dots F_n$ is of sort S. An HPSG principle is a boolean combination of these kinds of formulae with the free variable x being universally quantified since the principle is to be true at every node. A grammar is simply a conjunction of principles. We note that for the purpose of the present paper it is immaterial if a grammar is of the limited formula class just described or if it is any first-order sentence.

A relational structure $\mathfrak{A}$ over a signature Δ is a model for a grammar Γ, iff $\mathfrak{A}\models\mathcal{A}_\Delta\wedge\Gamma$. We will only consider *finite* structures. This is linguistically justified if one keeps in mind that such a structure is the linguistic analysis of an utterance. It seems unclear what an infinite analysis is supposed to mean.

The two most prominent alternatives for a formalisation of HPSG are feature logics (see, e.g., Rounds (1997) for a survey) or certain modal logics (see, e.g., (Blackburn and Spaan, 1993; Kracht, 1995)). Since proponents of both alternatives typically propose a weak logic, there often is a translation into first-order logic available. For the particular example of Speciate Reeantrant Logic by King (1989) this was shown by Aldag (1997). For modal logics, several such translation methods are provided by Ohlbach, Nonnengart, de Rijke, and Gabbay (2001).

9.3 Graphs as Logical Structures

Almost from the beginning, feature structures were considered as particular types of graphs. Graph-theoretically they are rooted directed hypergraphs where the sorts are unary edges. We will now define hypergraphs following the proposal by Courcelle (1990a,b) and quoting it freely where appropriate. Since all the graphs we are dealing with are hypergraphs, we will omit the prefix *hyper-*. A signature or ranked alphabet Σ is a finite set L of labels together with a function $\rho : L \to \mathbb{N}$ equipping each label with an arity. In the case of HPSG, the signature consists of the unary sort symbols $\mathcal{S}$ and the binary feature symbols $\mathcal{F}$.

Definition 1. Let Σ be a signature. A *concrete graph* is a quintuple $G = \langle V, E, lab, inc, prt\rangle$ where

- V is a set whose elements are the vertices of the graph;
- E is a set whose elements are the edges;
- $lab : E \to L$ is an edge labelling function;
- $inc : E \to V^*$ associates with each edge e the sequence of its vertices, a sequence of length $\rho(lab(e))$;
- prt is a sequence in V^* of vertices, the *ports*. The length of this sequence is the type of the graph G.

Thus a concrete graph consists of a set of vertices and a set of labelled edges between the vertices. Vertices are labelled by unary edges. A sort of HPSG turns out to be a unary labelled edge that is attached to a vertex. The type of an edge e is the arity of its label, i.e, $type(e) = \rho(lab(e))$. The ports are needed for technical purposes only. A *graph* is the equivalence class of all isomorphic concrete graphs. The set of graphs over a signature Σ is denoted by $\mathcal{G}_\Sigma$.

A graph is *finite*, iff both V and E are finite. As explained before, we restrict our attention to finite graphs. This definition of graphs is very general. Edges are for example not restricted to be binary, they may connect more than two vertices. For a model of HPSG-style feature structures, this generality is certainly not needed. Rather the subset of graphs that are HPSG-style feature structures are *directed multi graphs*. Multi graphs are graphs where the range of ρ is the set $\{0, 1, 2\}$, i.e., each edge is at most binary. A graph is directed if the range of inc is generally not symmetric, i.e., edges have a start and an end vertex. HPSG feature structures are also rooted, but this property is of no relevance for us.

Bauderon and Courcelle (1987) define three families of graph operations to turn a set of graphs into a many-sorted algebra of graphs. (Other such complete families of graph operations are presented, e.g., by Engelfriet (1997) or Courcelle (1997).) The first family of operations is *disjoint sum*. Let G and H be two graphs of types n and n'. We can assume their sets of vertices and edges to be disjoint. Then $G \oplus H$ is $\langle V_G \cup V_H, E_G \cup E_H, lab_G \cup lab_H, inc_G \cup inc_H, prt_G\hat{\ } prt_H\rangle$ of type $n + n'$.

The second operation is the *port redefinition*. This operation renames or "forgets" ports. If G is a graph of type n and $\alpha : \{1, \ldots, k\} \to \{1, \ldots, n\}$ then the graph after port redefinition is $\langle V_G, E_G, lab_G, inc_G, prt_G(\alpha)\rangle$ of type k.

The third operation is the *port fusion*. It fuses port vertices, i.e., the operation identifies groups of vertices. For every equivalence relation R on the set $\{1, \ldots, n\}$ there is a mapping fu_R taking a graph G of type n and returning G' that is obtained from G by identifying the ports that are in the same equivalence class of R.

Note that each of the three families of operations contains denumerably many operators. The set of all graphs together with these three families of operations forms a many-sorted algebra, where the sorts are just the types of the graphs. Bauderon and Courcelle (1987) also define a (many-sorted) algebra of so-called *graph expression* $\mathcal{GE}$ in the following way. It is the term algebra of the (sorted) operation symbols of the above described graph operations together with the following constants: 0 (denoting the empty graph), 1 (denoting the graph consisting of a single vertex), a for each label $a \in L$ (denoting the graph consisting of a single edge of type $\rho(a)$ of label a together with its $\rho(a)$ vertices). An element of this term algebra is called a *graph expression*. It can and should be seen as an instruction for constructing a graph. The graph is the value of the expression: Since the term algebra is the free algebra of this signature of graph operations there exists a unique homomorphism from the algebra of graph expressions to the algebra of graphs. The value of a graph expression is exactly the value of this homomorphism. It is now interesting to see that the homomorphism is surjective.

Proposition 2. (Bauderon and Courcelle, 1987) *Every finite graph is the value of a graph expression.*

A graph expression is a term, hence a tree. We can therefore use graph expressions to define particular sets of graphs.

Definition 3. A set of graphs is *context-free* iff the set of graph expressions denoting this set is regular. A set of graphs is *extended context-free* iff the set of graph expressions denoting it is context-free.

As usual, a set of trees is regular iff it is generated by a regular tree grammar or, equivalently, accepted by a tree automaton. A set of trees is context-free iff it is generated by a context-free tree grammar (Rounds, 1970). The notion of *context-freeness* for graphs stems from context-free graph grammars such as hyperedge replacement grammars. It is well known that a set of graphs is the language of a context-free graph grammar just in case it is the value of a regular set of graph expressions (Lautemann, 1988).

We need two notions for the size of a graph. The first one, *width*, is roughly the size of the signature of the expression for that graph. The second one, *treewidth*, is a measure on how tree-like a graph is.

Definition 4. Let g be a graph expression. The *width* $wd(g)$ is the maximal sort of a symbol of the graph expression algebra occurring in g. The *width* of a graph G is $wd(G) := min\{wd(g) \mid val(g) = G\}$.

Definition 5. Let G be a graph. A *tree decomposition* of G is a pair (T, f) where T is an unrooted unoriented tree and $f : V_T \to \wp(V_G)$ is a mapping such that
(1) $V_G = \bigcup\{f(i) \mid i \in V_T\}$;
(2) for every edge e of G there is a set $f(i)$ such that all vertices of e are in $f(i)$;
(3) if $v \in f(i) \cap f(j)$, then $v \in f(k)$ for every k belonging to the unique loop-free path from i to j in T.

The width of a tree decomposition is defined as $max\{|f(i)| \mid i \in V_T\} - 1$, and the *treewidth* of G is the smallest width of a tree decomposition of G.

As was shown by Courcelle, the treewidth of a directed multi graph provides a lower bound for its width.

Proposition 6. (Courcelle, 1992) *Let G be a finite directed multi graph. Then $twd(G) < wd(G)$.*

A natural and indeed very powerful choice of a logical language for graphs is monadic second-order logic of vertices and and edges (MS_2). Monadic second-order logic extends first-order logic by the addition of set variables and quantification over set variables. Thus the language offers existential and universal quantification over vertices, edges, sets of vertices, and sets of edges. Many sets of graphs can be defined in MS_2 such as, e.g., planar graphs, 3-colourable graphs, or graphs with a Hamiltonian cycle. The translation of the appropriateness function and a grammar of a finite HPSG signature into MS_2 is a simple task, in which no second order quantification is needed.

A set C of graphs of sort n is called *abstractly recognisable*, iff there exists a many-sorted algebra $\mathfrak{A}$ over the possibly infinite signature of graph operations with *finite* universes (sort sets), a homomorphism $h : \mathcal{GE} \to \mathfrak{A}$, and a finite subset FS of A_n such that $C = h^{-1}(FS)$. The triple $(h, \mathfrak{A}, FS)$ is called an automaton, the set FS is the set of final states. The homomorphism h is uniquely given due to the fact that $\mathcal{GE}$ is the free algebra over the signature of graph operations.

Proposition 7. (Courcelle, 1990b) *Every MS_2-definable set of graphs is abstractly recognisable.*

9.4 Graph Properties of HPSG Feature Structures

We can now present a finiteness result for HPSG, namely that for each set of HPSG feature structures defined by a grammar there exists a graph algebra over a possibly infinite signature of operations where each sort set is finite such that each feature structure is generated by this algebra. In other words

Theorem 8. *For every finite HPSG signature and grammar, the set of models of that signature and grammar is abstractly recognisable.*

As shown in the previous sections, every finite HPSG signature and every grammar are MS_2-definable. So, let $\mathcal{A}_\triangle$ be the MS_2-sentence defining the signature and Γ be the MS_2-sentence defining the grammar. By Proposition 7, the set of models of $\mathcal{A}_\triangle \wedge \Gamma$ is abstractly recognisable.

This finiteness result is quite a weak one, because due to the infinite signature there are infinitely many sort sets or universes. That means that the algebra as a whole is not finite. It is only the universe for each sort that is finite.

It would therefore be desirable to obtain a stronger finiteness result, namely one in which the whole algebra is finite. But such a result cannot be established, as we will show now. We demonstrate that there is a set of finite feature structures such that there is no graph algebra with a *finite* graph operator signature which generates this set. To do so, let us consider a special set of graphs, namely grids. Grids are special types of planar graphs. They are the result of gluing squares in lines and columns. forming a rectangular shape. An example of a 6×3 grid is given in Figure 9.1.

$$\begin{tikzcd}
\bullet_u \arrow[r, "R"] & \bullet_u \arrow[r, "R"] & \bullet_u \arrow[r, "R"] & \bullet_u \arrow[r, "R"] & \bullet_u \arrow[r, "R"] & \bullet_u \arrow[r, "R"] & \bullet_c \\
\bullet_s \arrow[r, "R"] \arrow[u, "U"'] & \bullet_s \arrow[r, "R"] \arrow[u, "U"'] & \bullet_s \arrow[r, "R"] \arrow[u, "U"'] & \bullet_s \arrow[r, "R"] \arrow[u, "U"'] & \bullet_s \arrow[r, "R"] \arrow[u, "U"'] & \bullet_s \arrow[r, "R"] \arrow[u, "U"'] & \bullet_r \arrow[u, "U"'] \\
\bullet_s \arrow[r, "R"] \arrow[u, "U"'] & \bullet_s \arrow[r, "R"] \arrow[u, "U"'] & \bullet_s \arrow[r, "R"] \arrow[u, "U"'] & \bullet_s \arrow[r, "R"] \arrow[u, "U"'] & \bullet_s \arrow[r, "R"] \arrow[u, "U"'] & \bullet_s \arrow[r, "R"] \arrow[u, "U"'] & \bullet_r \arrow[u, "U"'] \\
\bullet_s \arrow[r, "R"] \arrow[u, "U"'] & \bullet_s \arrow[r, "R"] \arrow[u, "U"'] & \bullet_s \arrow[r, "R"] \arrow[u, "U"'] & \bullet_s \arrow[r, "R"] \arrow[u, "U"'] & \bullet_s \arrow[r, "R"] \arrow[u, "U"'] & \bullet_s \arrow[r, "R"] \arrow[u, "U"'] & \bullet_r \arrow[u, "U"']
\end{tikzcd}$$

Figure 9.1: A 6×3 grid.

It is not difficult to see that HPSG formalisations allow the construction of grids. Consider the following signature $\langle\{s, u, r, c\}, \{U, R\}\rangle$ with the appropriateness conditions

$$\begin{array}{ccccccc} s & U & s, u & \qquad & u & R & u, c \\ s & R & s, r & \qquad & r & U & r, c \end{array}$$

where $s \quad U \quad s, u$ reads sort s has obligatory feature U and the sort at the end vertex of U may be s or u. We suppose the grammar to be empty. The set $\mathcal{GR}$ of all finite graphs that are models of this signature contains all finite grids such as the one in Figure 9.1. It does, of course, contain many more graphs that are not grids. But this is not of importance for us. If desired, the set of graphs can be restricted by adding principles so that only grid-like graphs are in the set. Indeed, Courcelle (1997) shows how to define finite grids in monadic second-order logic.

Theorem 9. *The set $\mathcal{GR}$ has unbounded width.*

This is shown as follows. $\mathcal{GR}$ contains all finite grids. The treewidth of an $n \times k$ grid is $min(n, k)$ (see (Bodlaender, 1998)). Therefore there is no bound on the treewidth of $\mathcal{GR}$. Since all feature structures are multi graphs, there is by Proposition 6 no bound on the width of $\mathcal{GR}$, either.

This theorem expresses that the signature for the graph expressions generating $\mathcal{GR}$ is infinite.

Proposition 10. *The set $\mathcal{GR}$ of graphs is neither context-free nor extended context-free.*

This is a simple consequence of the fact that every tree grammar has a finite set of productions. Thus the set of operators in terms is finite. But the width of $\mathcal{GR}$ is unbounded.

Corollary 11. *HPSG feature structures are in general neither context-free nor extended context-free.*

Unfortunately this also means that standard techniques from universal algebra or automata theory cannot be applied. It is a philosophical questions whether graphs similar to grids should be regarded as appropriate for linguistic analyses. There are good reasons to believe that HPSG grammarians do not have grid-like feature structures in mind when they think about models of their grammars. From the point of view of a licensing theory this would mean that additional principles have to be added that generally state desirable graph properties of the intended model. But most logics for HPSG, including (R)SRL and modal logics, are not powerful enough to formulate principles that exclude grid-like feature structures (and only these).

There is an interesting analogon in formalisations of GB-theory. Rogers (1998) showed that under the assumption of free indexation, an assumption that is standard in this theory, the construction of grids cannot be excluded either.

One obvious way to remedy the problem for HPSG is to demand of a class of models to be generable by a context-free graph grammar. Such a demand cannot be postulated as a principle but would have to be extrinsic. An interesting consequence of such a demand would be that treebanks for HPSG could be queried in time *linear* in the size of the treebank using MS_2 as a query language.

Bibliography

Aldag, B. (1997). *A Proof Theoretic Investigation of Prediction in HPSG*. Master's thesis, Seminar für Sprachwissenschaft, University of Tübingen.

Bauderon, M. and B. Courcelle (1987). Graph expressions and graph rewritings. *Mathematical Systems Theory*, **20**:83–127.

Blackburn, P. and E. Spaan (1993). A modal perspective on the computational complexity of attribute value grammar. *Journal of Logic, Language, and Information*, **2**:129–169.

Bodlaender, H. L. (1998). A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, **209**:1–45.

Chomsky, N. (1981). *Lectures on Government and Binding*. Foris Publications, Dordrecht, Holland.

Courcelle, B. (1990a). Graph rewriting: An algebraic and logic approach. In J. van Leeuwen, ed., *Handbook of Theoretical Computer Science*, volume B, chapter 5, pp. 193–242. Elsevier.

Courcelle, B. (1990b). The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation*, **85**:12–75.

Courcelle, B. (1992). The monadic second-order logic of graphs III: Tree-decompositions, minors and complexity issues. *Informatique Théoretique et Applications*, **26**:257–286.

Courcelle, B. (1997). The expression of graph properties and graph transformations in monadic second-order logic. In G. Rozenberg, ed., *Handbook of Graph Grammars and Computing by Graph Transformation*, pp. 313–400. World Scientific Publishing.

Ebbinghaus, H.-D. and J. Flum (1995). *Finite Model Theory*. Springer-Verlag.

Engelfriet, J. (1997). Context-free graph grammars. In G. Rozenberg and A. Salomaa, eds., *Handbook of Formal Languages, Vol III: Beyond Words*, pp. 125–213. Springer.

King, P. J. (1989). *A Logical Formalism for Head-Driven Phrase Struture Grammar*. Ph.D. thesis, University of Manchester.

Kracht, M. (1995). Is there a genuine modal perspective on feature structures? *Linguistics and Philosophy*, **18**:401–458.

Lautemann, C. (1988). Decomposition trees: Structured graph representation and efficient algorithms. In *Proceedings CAAP*, volume 299 of *LNCS*, pp. 28–39.

Ohlbach, H. J., A. Nonnengart, M. de Rijke, and D. Gabbay (2001). Encoding two-valued nonclassical logics in classical logic. In A. Robinson, ed., *Handbook of Automated Reasoning*, pp. 1403–1486. North Holland.

Pollard, C. and I. A. Sag (1987). *Information Based Syntax and Semantics, Vol. 1: Fundamentals*. Number 13 in Lecture Notes. CSLI.

Pollard, C. and I. A. Sag (1994). *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

Pullum, G. and B. Scholz (2001). On the distinction between model-theoretic and generative-enumerative syntactic frameworks. In P. de Groote, G. Morrill, and C. Retoré, eds., *LACL 2001*, number 2099 in LNAI, pp. 17–43. Springer-Verlag.

Rogers, J. (1998). *A Descriptive Approach to Language-Theoretic Complexity*. CSLI Publications.

Rounds, W. (1970). Mappings and grammars on trees. *Mathematical Systems Theory*, **4**(3):257–287.

Rounds, W. (1997). Feature logics. In J. van Benthem and A. ter Meulen, eds., *Handbook of Logic and Language*, pp. 475–533. Elsevier.