Algebraic Methods in Language Processing AMiLP-3,
Anton Nijholt and Giuseppe Scollo (eds.), 2003.

(Un-)Decidability Results for Head-Driven Phrase Structure Grammar

Stephan Kepser and Uwe Mönnich*
Dept. of Linguistics, University of Tübingen, Germany
{kepser,um}@sfs.uni-tuebingen.de

Abstract

Head-driven Phrase Structure Grammar (HPSG, Pollard and Sag (1987, 1994)) is currently one of the most prominent linguistic theories. A grammar for HPSG is given by a set of abstract language universal principles, a set of language specific principles, and a lexicon. A sentence is grammatical, if it is compatible with all of the principles. The data structures underlying HPSG are so-called feature structures. These have always been considered as graphs. In this paper, we formalise feature structures as (hyper-) graphs and use monadic second-order logic (MS_2) as a language to talk about these feature graphs. We indicate that HPSG grammar principles can be expressed as MS_2-formulae over feature graphs. And we investigate the decidability properties of this framework. Doing so, we show that the MS_2-theory of HPSG feature graphs is undecidable. Since the general definition of feature graphs we present does not only lead to the undecidability result, but has also other linguistically undesirable properties, we propose to restrict the class of feature graphs to the subclass that can be generated by so-called hyperedge replacement grammars. Under this restriction, the MS_2-theory of feature graphs becomes decidable.

Keywords: HPSG, Monadic Second-Order Logic, Hypergraphs, Decidability

1 Introduction

Head-driven Phrase Structure Grammar (HPSG, Pollard and Sag (1987, 1994)) is currently one of the most prominent linguistic theories. HPSG is a so-called licensing theory. A grammar of such a theory is not a rule-based system that generates the sentences that are grammatical according to the theory. Rather grammaticality is expressed by a set of highly abstract principles some of which are supposed to be language-specific, while others are seen as universally valid for all natural languages. A sentence is grammatical if there exists an analysis of the sentence that does not violate any of the principles. Licensing theories usually do not make any statements on how such an analysis may be gained. They just demand the adherence of the analysis to the principles.

Quite a significant part of the attraction of HPSG to linguists is due to a relatively rigorous formal framework underlying it and the commitment of prominent HPSG grammarians to a strict formalisation of grammar theories. It is therefore only natural that there is a long, ongoing discussion about "the right logic" for HPSG and its intended underlying data structures, namely feature structures. The three most important groups of logics that have been proposed are feature logics, modal logics, and classical first-order predicate logic. An important question in these discussions concerns the decidability of the particular modal or feature logic. This question is not only interesting for the theoretical logician. Pollard and Sag explicitely demand decidability of an HPSG formalisation (see Pollard and Sag (1994), p. 10).

*This research was partly funded by a grant of the German Science Foundation (DFG SFB 441-2).

Feature logics are logics especially designed to describe and model feature structures, thereby also defining what feature structures are or should be. They typically have a restricted expressive power. A good overview of feature logics is given by Rounds (1997). Particularly influential on the ideas about feature structures developed by Pollard and Sag in their later book (Pollard and Sag, 1994) were the proposals by Carpenter (1992) and King (1989), although these proposals are not necessarily compatible.

The design of King's Speciate Reentrant Logic (King, 1989) is very closely driven by the needs of formalisations of HPSG principles. It was shown by Kepser (1994) that consistency of a formula of this logic is decidable. But King et al. (1999) proved that the more important notion of grammaticality, which demands an SRL formula to be true for every object in the denotation domain, is undecidable.

Richter (2000) proposed RSRL, a relational extension of SRL, which significantly extends the expressive power of the logic by introducing sets, lists, and arbitrary relations over them. As RSRL is an extension of SRL, grammaticality is undecidable a fortiori. But furthermore the logic is that expressive that it is in general even undecidable whether a formula is true in a given structure, as was shown by Kepser (2001).

Feature structures have also been interpreted as multi-modal structures (see, in particular, Kracht (1995)). The key idea here is that features are modal operators. For modal logics formalising principles of HPSG Blackburn and Spaan (1993) showed that these logics are undecidable.

It is interesting to see that classical first-order predicate logic has not so often been advocated as a logic suitable for HPSG as compared to the groups of feature logics or modal logics. An important reason behind this observation is the fact that the original ideas behind HPSG as spelled out in the book published in 1987 (Pollard and Sag, 1987) were inspired by concepts of information, information flow, and information growth. These notions are very well captured by intuitionistic logics whereas it takes a considerable amount of forcing and torquing to describe them using classical logic. When in the second book (Pollard and Sag, 1994) the model theoretic ideas were shifted from intuitionistic to the classical paradigm, feature logics and modal logics were already established as "the logics" for HPSG so that no one "in the business" felt a need or inclination to use classical first-order logic.

On the other hand there are methods that translate formalisations in feature logics or modal logics into classical first-order logic. For SRL, such a method was presented by Aldag (1997). For modal logic, there is a whole research branch devoted to the translation into classical logic, an overview of which can be found in the work by Ohlbach et al. (2001).

In this paper, we intend to take the intuition that feature structures are graphs seriously. We describe feature structures as a particular kind of hypergraphs, namely finite rooted multigraphs. We show that monadic second-order logic, i.e., the extention of first-order logic by set variables and quantification over sets of vertices and edges, has an expressive power that makes it easy to state HPSG grammar formalisms. We investigate decidability properties of monadic second-order logic (MS_2) over HPSG feature graphs and find that the MS_2-theory of finite HPSG feature graphs is in general undecidable. In other words, it is in general not possible to decide if a given sentence of MS_2 has an HPSG feature graph as its model.

One way to interpret this result is to state that HPSG feature graphs cannot be generated (see also the article by Kepser and Mönnich (2003)). If one demands graphs to be generated by a so-called context-free hyperedge replacement grammar, then the MS_2-theory of finite HPSG feature graphs becomes decidable. Whether one should postulate this, is a matter of philosophical debate. From the perspective of a purely licensing theory there is no reason to do so. On the other hand, licensing theories do not demand that an analysis must be non-generable. It is unimportant how and from where an analysis is obtained. Thus it could as well be generated somehow. The non-generability conflicts with an old expectation (going at least back to Humboldt) that a grammar should use *finite* means to produce the infinitely many structures of a language. Here, things depend of course on the interpretation of the notion *produce*. For more details, see the article by

Kepser and Mönnich (2003). Of course, decidability of the logic is a virtue of its own that makes it well worth considering if the restriction to generability via graph grammars is not one that we can live with.

After some technical preliminaries we will introduce the relevant concepts from (hyper-) graph theory and graph grammars in Sections 3 and 4. Thereafter the main section follows that contains the description of HPSG as graphs using MS_2 as the logical language, and the main results of this paper.

2 Preliminaries

The definition of many-sorted algebras we give below follows the exposition by Courcelle (1990b). We assume that sets of sorts and sets of operators can be *infinite*. Let $\mathcal{S}$ be a set of *sorts*. An $\mathcal{S}$-signature is a set Σ given with two mappings $\alpha : \Sigma \to \mathcal{S}^*$ (the *arity* mapping) and $\sigma : \Sigma \to \mathcal{S}$ (the *sort* mapping). The length of $\alpha(f)$ is called the *rank* of f, and is denoted by $\rho(f)$. The profile of f in Σ is the pair $(\alpha(f), \sigma(f))$.

A Σ-*algebra* is an pair $\mathfrak{A} = \langle (A_s)_{s\in\mathcal{S}}, (f)_{f\in\Sigma}\rangle$ where A_s is a nonempty set for each $s \in \mathcal{S}$, called the domain or universe of sort s of $\mathfrak{A}$, and $f : A_{\alpha(f)} \to A_{\sigma(f)}$ is a total function for each $f \in \Sigma$. (For a sequence $\mu = (s_1, \ldots, s_n)$ in $\mathcal{S}^+$, we let $A_\mu := A_{s_1} \times A_{s_2} \times \cdots \times A_{s_n}$.)

We will now define regular tree grammars. Let Σ be a single sorted signature and $\mathcal{T}_\Sigma$ the free term algebra over Σ, i.e., $c \in \mathcal{T}_\Sigma$ for all constants c with empty arity; and if $f \in \Sigma$ is of rank n and $t_1, \ldots, t_n \in \mathcal{T}_\Sigma$ then $f(t_1, \ldots, t_n) \in \mathcal{T}_\Sigma$. A *regular tree grammar* is a a quadruple $\Gamma = (\Delta, \Sigma, P, S)$ where Δ is a single sorted signature of operatives, all of rank 0, Σ a single sorted signature of inoperatives (of arbitrary rank), P a finite set of productions, and $S \in \Delta$ the start symbol. Each production is of the form $A \to t$ where $A \in \Delta$ is an operative, and $t \in \mathcal{T}_{\Sigma\cup\Delta}$. Note that since all operatives in Δ are constants, they can only appear as leafs in the tree t. Intuitively, an application of a rule $A \to t$ replaces a leaf node A by the tree t. If s is a tree and A a leaf in s then we write $s \Rightarrow s[A/t]$ for a single step derivation. If there is a sequence $s_0 \Rightarrow \cdots \Rightarrow s_k$ of single step derivations, we write $s_0 \Rightarrow^* s_k$ for the derivation of s_k from s_0. The language of a regular tree grammar Γ is defined as the set $La(\Gamma) := \{t \in T_\Sigma \mid S \Rightarrow^* t\}$. A set M of trees (or terms) is called regular, iff there exists a regular tree grammar Γ such that $M = La(\Gamma)$.

3 Graphs

From its very beginning, HPSG feature structures have intuitively always been regarded as graphs. We propose to take this intuition seriously and formalise them as a special kind of hypergraphs, namely rooted directed multigraphs. Hypergraphs are generalisations of graphs where edges are labelled and have arbitrarily many vertices. 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 a rank.

Definition 1 Let $\Sigma = (L, \rho)$ be a signature. A *concrete graph* over Σ 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 of length n in V^* of pairwise disjoint vertices, the *ports*. The integer n 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 may be labelled by unary edges. 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. 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. We restrict our attention to finite graphs.

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 $\mathcal{G}$, 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 graph operations described above 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 $h : \mathcal{GE} \to \mathcal{G}$ 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.*

Definition 3 Let O be a *finite* set of graph operation symbols. A set M of graphs is called *equational*, iff it is the value of a *regular* set $R \subset \mathcal{GE}$ of graph expressions, i.e., the value of a set R of graph expressions defined by a regular tree language.

To put it the other way around, a regular tree grammar that generates R can be viewed as generating a set M of graphs by taking the values of the graph expressions R. A set M of graphs so given is called equational. The name reflects the fact that the tree grammar can be seen as a system of equations of which $val(R)$ is the least fixed point. This concept was introduced by Mezei and Wright (1967), for details, see Courcelle (1990b) or Engelfriet (1997).

A natural and indeed very powerful choice of a logical language for graphs is monadic second-order logic of vertices and edges (MS_2). Monadic second-order logic extends first-order logic by the addition of set variables and quantification over set variables. More precisely, let $\mathbf{V}$ be the *sort* of vertices and $\mathbf{E}$ be the sort of edges. Let $\{v, v', v_0, v_1, v_2, \ldots, e, e', e_0, e_1, e_2, \ldots\}$ be an infinite denumerable set of *object variables* each having sort $\mathbf{V}$ or $\mathbf{E}$. Let $\sigma(u)$ denote the sort of u. Let $\{V, U, E, E', X, Y, Z, \ldots\}$ be an infinite denumerable set of *set variables* each of sort $\mathbf{V}$ or $\mathbf{E}$. Again, let $\sigma(U)$ denote the sort of U. For each label $l \in L$ with arity k there exists a predicate symbol

edg_l of arity $k+1$ where the first argument is of sort **E** and all others of sort **V**. The atomic formulae of MS_2 are

- $u = u'$ where $\sigma(u) = \sigma(u')$,
- $u \in U$ where $\sigma(u) = \sigma(U)$,
- $edg_l(e, v_1, \ldots, v_k)$ where $\rho(l) = k, \sigma(e) = \mathbf{E}$, and for all $1 \leq i \leq k$: $\sigma(v_i) = \mathbf{V}$.

Complex formulae are recursively defined as follows. Let φ, ψ be formulae, u an object variable (of any sort) and U a set variable (of any sort), then

- $\neg\varphi, (\varphi \wedge \psi), (\varphi \vee \psi), (\varphi \rightarrow \psi)$,
- $\forall u\varphi, \exists u\varphi$,
- $\forall U\varphi$, and $\exists U\varphi$

are formulae.

The semantics of MS_2 is an extension of the semantics of first-order logics. Let $G = \langle V, E, lab, inc, prt\rangle$ be a graph. A *variable assignment* α is a function that assigns each object variable u either an element of V, if $\sigma(u) = \mathbf{V}$, or an an element of E, if $\sigma(u) = \mathbf{E}$, and that assigns each set variable U either a subset of V, if $\sigma(U) = \mathbf{V}$, or a subset of E, if $\sigma(U) = \mathbf{E}$. A modified variable assignment $\alpha[u/v]$ ($\alpha[U/V]$ resp.) is identical to a variable assignment α with the potential difference that it assigns v (resp. V) to variable u (resp. U). The denotation of formulae for a given graph G and variable assignment α is as follows.

- $u = u'$ is true iff $\alpha(u) = \alpha(u')$;
- $u \in U$ is true iff $\alpha(u) \in \alpha(U)$;
- $edg_l(e, v_1, \ldots, v_k)$ is true iff $inc(\alpha(e), \alpha(v_1), \ldots, \alpha(v_k))$ and $lab(\alpha(e)) = l$;
- $\neg\varphi$ is true iff φ is false;
- $(\varphi \wedge \psi)$ is true iff φ and ψ are true;
- $(\varphi \vee \psi)$ is true iff φ is true or ψ is true;
- $(\varphi \rightarrow \psi)$ is true iff φ is false or ψ is true;
- $\forall u\varphi$ is true if $\sigma(u) = \mathbf{V}$ and for all $v \in V : \varphi$ is true under assignment $\alpha[u/v]$ or if $\sigma(u) = \mathbf{E}$ and for all $e \in E : \varphi$ is true under assignment $\alpha[u/e]$;
- $\exists u\varphi$ is true if $\sigma(u) = \mathbf{V}$ and there is a $v \in V$ such that φ is true under assignment $\alpha[u/v]$ or if $\sigma(u) = \mathbf{E}$ and there is a $e \in E$ such that φ is true under assignment $\alpha[u/e]$;
- $\forall U\varphi$ is true if $\sigma(U) = \mathbf{V}$ and for all $V' \subseteq V : \varphi$ is true under assignment $\alpha[U/V']$ or if $\sigma(U) = \mathbf{E}$ and for all $E' \subseteq E : \varphi$ is true under assignment $\alpha[U/E']$;
- $\exists U\varphi$ is true if $\sigma(U) = \mathbf{V}$ and there is a $V' \subseteq V$ such that φ is true under assignment $\alpha[U/V']$ or if $\sigma(U) = \mathbf{E}$ and there is a $E' \subseteq E$ such that φ is true under assignment $\alpha[U/E']$.

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. As an example, the following formula defines 3-colourability of a simple graph. Let v, v', R, G, B be all of sort **V** and e of sort **E**.

$$\begin{aligned}\exists R, G, B\ \ & \forall v\, v \in R \vee v \in G \vee v \in B \wedge \neg(v \in R \wedge v \in G) \wedge \neg(v \in R \wedge v \in B) \wedge \neg(v \in G \wedge v \in B) \wedge \\ & \forall v, v'(\exists e\, edg(e, v, v')) \rightarrow \ \ (v \in R \wedge v' \in G) \vee (v \in G \wedge v' \in R) \vee \\ & \qquad\qquad\qquad\qquad\ \ (v \in G \wedge v' \in B) \vee (v \in B \wedge v' \in G) \vee \\ & \qquad\qquad\qquad\qquad\ \ (v \in R \wedge v' \in B) \vee (v \in B \wedge v' \in R)\end{aligned}$$

Note that since the graph is simple, there is only one *edg* relation. The first line says that the three colour sets R, G, and B partition the domain of vertices. The second part expresses that if two vertices are connected by an edge then they must be in different colour sets.
Courcelle (1990b) showed that if a relation is MS_2-definable then so is its transitive closure. This is another example of the expressive power of MS_2.

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 $h : \mathcal{G} \rightarrow \mathfrak{A}$ from the algebra of graphs 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.

Proposition 4 (Courcelle, 1990b) *The intersection of an abstractly recognisable set of graphs and an equational set of graphs is an equational set.*

In the proof, the intersection is effectively computed.

Proposition 5 (Courcelle, 1990b) *Every MS_2-definable set of graphs is abstractly recognisable.*

There exists a measure on how similar to a tree a graph is. It is called *treewidth*, and was introduced by Robertson and Seymour (1986a).

Definition 6 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.

All proper trees have treewidth 1. Cliques, i.e., graphs where each pair of vertices is connected by an edge, on the other hand, are clearly graphs that differ very much from trees. Their treewidth is the size of the clique minus one. The notion of treewidth will be quite important for us. Roughly speaking, a set of graphs for which there exists a uniform bound on the treewidth is well-behaved, so to speak. Sets of graphs that have an unbounded treewidth on the other hand are a lot more difficult to treat. As an important example, Seese (1991) showed the following

Proposition 7 *The MS_2-theory of a set of finite graphs of unbounded treewidth is undecidable.*

On the other hand, if M is an equational set of graphs, then there exists a uniform bound on the treewidth of the graphs in M (Engelfriet, 1997).

4 Graph Grammars

We will now introduce another way to define sets of graphs via grammars, namely as languages of so-called *context-free hyperedge replacement grammars*. As the name implies, a hyperedge replacement consists of taking out a hyperedge from a given graph and plugging a complete graph, not just an edge, into its original place. Thus a hyperedge replacement grammar mainly consists of pairs of hyperedges and graphs that should take their places. The notion of context-freeness means that the replacement of an edge by a graph takes place independently of the context of that edge in the "hosting" graph. For brevity, we will write *HR grammar* for context-free hyperedge replacement grammar. Recommendable overview articles on HR grammars are the ones by Drewes et al. (1997) and Engelfriet (1997). The technical exposition that follows is a quote from the article by Drewes et al. (1997).

Recall that the type of a concrete graph is the number of its ports. The type of an edge is the rank of its label. Edges are always replaced by graphs of the same type. Let H be a concrete graph and $B \subseteq E_H$ a set of edges to be replaced. Let $repl$ be a mapping from B to a set of concrete graphs such that $type(e) = type(repl(e))$ for each $e \in B$. Then the replacement of B in H by $repl$ yields a concrete graph $H[repl]$ obtained by removing B from E_H, adding the nodes and edges of $repl(e)$ for each $e \in B$ disjointly, and fusing the i-th port of $repl(e)$ with the i-th attachment node of e for each $e \in B$ and $i = 1, \ldots, type(e)$. All hyperedges keep their labels and attachment nodes; the ports of $H[repl]$ are the ports of H. If $B = \{e_1, \ldots, e_n\}$ and $repl(e_i) = R_i$ for $i = 1, \ldots, n$,

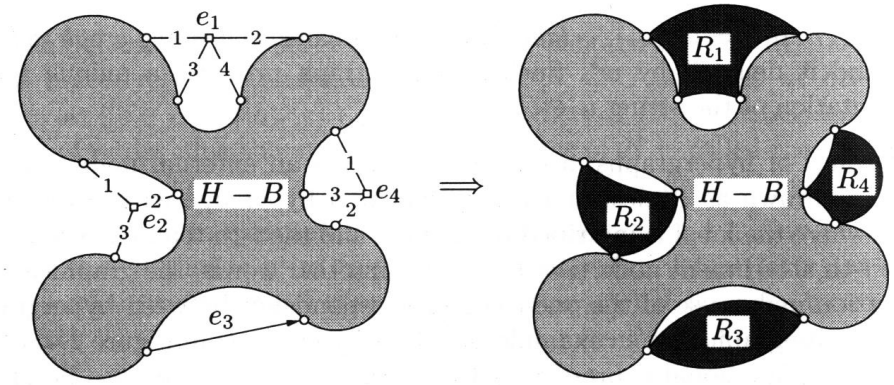

Figure 1: Hyperedge replacement of e_i by R_i in H $(i = 1, \ldots, 4)$. (From (Drewes et al., 1997))

then we write $H[e_1/R_1, \ldots, e_n/R_n]$ instead of $H[repl]$. A pictographical example of a hyperedge replacement is given in Figure 1. Hyperedge replacement is only defined up to isomorphism. Thus we will, as before, consider the isomorphism classes of concrete graphs as the graphs proper.

Hyperedge replacement possesses some nice properties. Firstly, it has the sequentialisation and parallelisation property. It does not matter if one replaces a set of edges simultaneously or one after the other. Therefore, replacement is confluent. The order in which a series of replacement is performed does not effect the result. Secondly hyperedge replacement is associative. The result of replacing e in graph G by F and then e' in graph H by $G[e/F]$ is the the same as replacing first e' in H by G and then e in $H[e'/G]$ by F, i.e., $H[e'/G[e/F]] = H[e'/G, e/F]$.

As before, let L be a set of labels. Let $N \subseteq L$ be a set of *nonterminals*. A *production* over N is an ordered pair $p = (A, R)$ where $A \in N$ and $R \in \mathcal{G}$ and the rank of A is equal to the type of R. Let $H \in \mathcal{G}$ be a graph and P be a set of productions. Let $e \in E_H$ and $(lab(e), R) \in P$. Then H *directly derives* $H' = H[e/R]$ and we write $H \Rightarrow_P H'$. A sequence of direct derivation $H_0 \Rightarrow \cdots \Rightarrow H_k$ is called a *derivation of length* k and denoted by $H_0 \Rightarrow^* H_k$.

Definition 8 A *context-free hyperedge replacement grammar* (or HR-grammar) is a quaduple $HRG = (N, T, P, S)$ where $N \subseteq L$ is a set of nonterminals, $T \subseteq L$ is a set of terminals with $T \cap N = \emptyset$, P is a finite set of productions over N, and $S \in N$ is the start symbol.

The graph language $La(HRG)$ generated by HRG is the set

$$La(HRG) = \{H \in \mathcal{G} \mid S^\bullet \Rightarrow H\}$$

where $S^\bullet$ is the smallest graph of label S, i.e., the graph consisting of rank of S many vertices and a single edge e of type rank of S connecting these vertices, and an empty port sequence.

We close this section with an important result. It shows that the two independent ways of defining a set of graphs, namely by a regular language of graph expressions (as explained in the previous section) and by HR grammars, actually posses the same expressive power.

Proposition 9 (Lautemann, 1988; Engelfriet, 1997) *A set of graphs M is equational iff it is the language of some context-free hyperedge replacement grammar.*

The proof given by Engelfriet (1997) is constructive, i.e., given an HR grammar, the proof provides a construction of a regular tree grammar of graph expressions and vice versa. Since HR grammars are context-free, equational sets of graphs are sometimes also called *context-free* sets of graphs.

5 HPSG Feature Structures as Graphs

Linguists see the feature structures of HPSG as graphs and often use graphs when they want to describe certain properties of a structure. A typical example is given in Figure 2, which is taken out of the book by Pollard and Sag (1994). It describes an entry in a lexicon, namely the entry for the English pronoun *she*. That such graphs can be regarded as special types of hypergraphs is

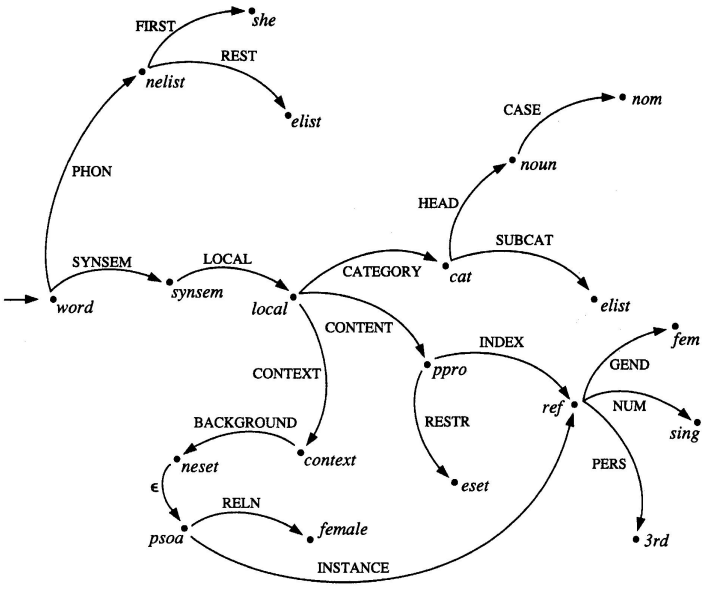

Figure 2: Feature Graph of the English pronoun *she*. (From (Pollard and Sag, 1994))

simple to see. HPSG feature graphs are rooted, and they are multigraphs, i.e., edges are at most binary. The full power of hyperedges is not really necessary. But we need unary edges, they will be used to label vertices. Edge labels are functional. Two edges departing from the same source vertex must have different labels. Edge labels are called *features* (or sometimes *attributes*), vertex labels are called *sorts*.

In interpreting feature structures as graphs we have to have a closer look at signatures for HPSG. In general, these signatures list the features and sorts. But these features and sorts are not unrelated. Rather there is a strict relationship between them: HPSG feature structures are required to be *totally well-typed* and *sort resolved*. Sort-resolvedness, basically 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 vertex. *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 finite set of sorts (unary predicate symbols), $\mathcal{F}$ is a finite set of features (binary predicate symbols), and $A : \mathcal{S} \times \mathcal{F} \to \wp(\mathcal{S})$ is an *appropriateness function* expressing the well-typedness restrictions. For example, part of the HPSG appropriateness function of Figure 2 is $A(word, \textsc{synsem}) = \{synsem\}, A(word, \textsc{phon}) = \{nelist\}$. It expresses that a vertex of sort *word* must have exactly two outgoing binary edges, one labelled PHON and one labelled SYNSEM. And the target vertex of the edge labelled PHON must be of sort *nelist*, while the target vertex of the edge labelled SYNSEM must be of sort *synsem*.

These signature requirements can be expressed in MS_2. HPSG sorts are unary edge labels, features are binary edge labels. Let u, v, w be variables of sort (in the algebraic sense) $\mathbf{V}$, i.e., vertex variables, and let e, e' be variables of sort $\mathbf{E}$ (edge variables). (Note that the HPSG sort and feature disjunctions and conjunctions (like $\bigvee_{s \in \mathcal{S}}$) are just abbreviations for long disjunctions and

conjunctions in a pure graph language, because the sets of sorts and features are both finite.)

$$
\begin{array}{l}
\forall v \exists e : \bigvee_{s \in \mathcal{S}} edg_s(e, v) \\
\wedge \forall v \forall e, e' : (\bigvee_{s \in \mathcal{S}} edg_s(e, v) \wedge \bigvee_{s \in \mathcal{S}} edg_s(e', v)) \rightarrow e = e' \\
\wedge \forall f \in \mathcal{F} : \forall u, v, w, e, e' : (edg_f(e, u, v) \wedge edg_f(e', u, w)) \rightarrow (v = w \wedge e = e') \\
\wedge \bigwedge_{s \in \mathcal{S}, f \in \mathcal{F}, A(s,f) \neq \emptyset} \forall v \ \ (\exists e : edg_s(e, v)) \rightarrow \\
\qquad\qquad\qquad\qquad (\exists e, u, e' : edg_f(e, v, u) \wedge \bigvee_{s' \in A(s,f)} edg_{s'}(e', u)) \\
\wedge \bigwedge_{s \in \mathcal{S}, f \in \mathcal{F}, A(s,f) = \emptyset} \forall v (\exists e : edg_s(v, e))) \rightarrow (\neg \exists e, u : edg_f(e, v, u))
\end{array}
$$

The first line states that each vertex must have an HPSG sort, i.e., a unary edge labeld with a sort. The second line states that a vertex has at most one HPSG sort. The third line states that features are functional, i.e., two binary edges departing from the same vertex and having the same (feature) label must be identical. The next two lines cope with the appropriateness function. The fourth line states that if a sort has admissible features, there must be edges present labelled with these features, and there must be target vertices for these edges that bear the right sort labels. The fifth line states that if a sort has no admissible features, a vertex labelled with this sort must not have any outgoing binary edges.

HPSG principles are formed out of boolean combinations of so-called path expressions and sort statements. Path expressions allow one to state that two feature paths in a graph end in the same vertex. Sort statements define the sort of a vertex at the end of a feature path in a graph. Both path expressions and sort statements can easily be expressed in MS_2. Indeed, when considered as a logic for HPSG, MS_2 is very expressive and should be powerful enough to render any linguistically motivated principle. Here is an example of an HPSG principle. It is the Head-Feature-Principle by Pollard and Sag (1994, p. 34):

> In a headed phrase, the values of SYNSEM|LOCAL|CATEGORY|HEAD and DAUGHTERS|HEAD-DGHT|SYNSEM|LOCAL|CATEGORY|HEAD are token-identical.

Its rendering in MS_2looks as follows. Let $v, v', u_1, u_2, u_3, u_4, w_1, w_2, w_3, w_4, w_5$ be variables of sort **V** and $e, e', f_1, f_2, f_3, f_4, g_1, g_2, g_3, g_4, g_5, g_6$ be variables of sort **E**.

$$
\begin{array}{l}
\forall v \ \ (\exists e, w_1, e', v' : edg_{\textsc{daughters}}(e, v, w_1) \wedge edg_{\textit{head-struc}}(e', v')) \rightarrow \\
\quad (\ \exists u_1, u_2, u_3, u_4, w_1, w_2, w_3, w_4, w_5, f_1, f_2, f_3, f_4, g_1, g_2, g_3, g_4, g_5, g_6 : \\
\quad edg_{\textsc{synsem}}(f_1, v, u_1) \wedge edg_{\textsc{local}}(f_2, u_1, u_2) \wedge edg_{\textsc{category}}(f_3, u_2, u_3) \wedge edg_{\textsc{head}}(f_4, u_3, u_4) \wedge \\
\quad edg_{\textsc{daughters}}(g_1, v, w_1) \wedge edg_{\textsc{head-dght}}(g_2, w_1, w_2) \wedge edg_{\textsc{synsem}}(g_3, w_2, w_3) \wedge \\
\quad edg_{\textsc{local}}(g_4, w_3, w_4) \wedge edg_{\textsc{category}}(g_5, w_4, w_5) \wedge edg_{\textsc{head}}(g_6, w_5, u_4))
\end{array}
$$

The premise of the implication demands the graph to be a headed phrase. Note that the key information of this long path equation is hidden in the reappearance of variable u_4 from $edg_{\textsc{head}}(f_4, u_3, u_4)$ in $edg_{\textsc{head}}(g_6, w_5, u_4)$, i.e., the two paths both end in vertex variable u_4.

Thus an HPSG grammar, consisting of the signature requirements and the principles, can be expressed by an MS_2-sentence. This sentence can be regarded as an axiomatisation for HPSG feature graphs. Compatibility of a graph with the grammar is rendered by the graph being a model of the MS_2-sentence.

We will now turn to showing that the MS_2-theory of finite HPSG feature graphs is undecidable. 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 3.

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 function

$$
\begin{array}{ll}
A(s, U) = \{s, u\} & A(u, R) = \{u, c\} \\
A(s, R) = \{s, r\} & A(r, U) = \{r, c\}
\end{array}
$$

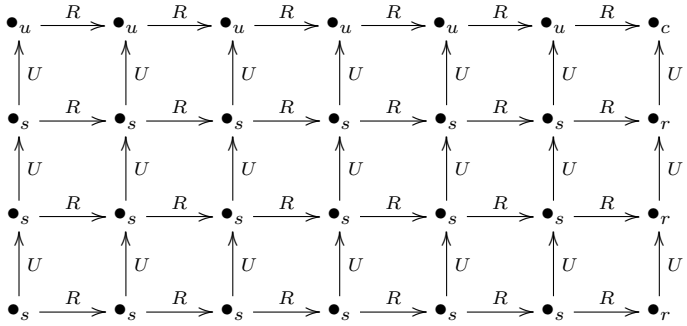

Figure 3: A 6×3 grid.

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 3. 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 MS_2.

Proposition 10 *The set $\mathcal{GR}$ has unbounded width.*

Proof. $\mathcal{GR}$ 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 $\mathcal{GR}$. ■

As a consequence, the main result of this paper can be established.

Theorem 11 *The MS_2-theory of finite HPSG feature graphs is undecidable.*

The proof of this theorem is a simple consequence of the above proposition and Proposition 7. ■

Note that since King et al. (1999) use infinite feature structures the above proposition is not an immediate consequence of the undecidability of grammaticality of SRL.

Since the proof is based on constructing graphs that are not very likely suitable models of linguistic analyses, there may be a way to get around the above undecidability result by excluding grid-like graphs from the set of models considered. One natural way to to this is to demand of the set of graphs to be generable by an HR grammar.

Theorem 12 *The MS_2-theory of finite HPSG feature graphs is decidable, if this set of graphs can be generated by an HR grammar.*

Proof. This proof is due to Courcelle (1990b).
Let ϕ be an MS_2-sentence. By Proposition 5, the set of graphs M_ϕ which are models of ϕ is abstractly recognisable. Courcelle (1990b) gives a construction of the abstract automaton in his proof. Let M be a set of graphs given by an HR grammar. By Proposition 9, this set is equational. By Proposition 4, one can construct a regular tree grammar defining $M \cup M_\phi$. One can test whether $M \cup M_\phi = \emptyset$. ■

6 Conclusion

It would in our opinion be desirable to obtain a decidability result without recourse to a graph grammar formalism that paves the way to decidability. But it seems that the expressive power of a typical modal or feature logic for HPSG is insufficient to forbid grids and their generalisations. The chances to exclude these graphs by means of MS_2 are a lot better. This can indeed be done and it seems worth adding such a formula as an axiom to the HPSG theory, because there are also independent linguistic motivations to exclude these types of graphs. The collection of graphs of a fixed treewidth is MS_2-definable. Suppose that H is a finite planar graph. Then the class $FORB(H)$ is MS_2-definable. This is the class of graphs G such that no member of G has H as minor, where H is a minor of G if it can be obtained from a subgraph of G by means of repeated edge contractions. According to an important result due to Robertson and Seymour (1986b) there is a bound n on the treewidth of all members of G. In other words, for any fixed n the set of finite graphs of treewidth at most n is MS_2-definable.

References

Aldag, B. (1997). A proof theoretic investigation of prediction in HPSG. Master's thesis, Seminar für Sprachwissenschaft, University of Tübingen.

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

Blackburn, P. and Spaan, E. (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.

Carpenter, B. (1992). *The Logic of Typed Feature Structures*. Cambridge University Press.

Courcelle, B. (1990a). Graph rewriting: An algebraic and logic approach. In van Leeuwen, J., editor, *Handbook of Theoretical Computer Science*, volume B, chapter 5, pages 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. (1997). The expression of graph properties and graph transformations in monadic second-order logic. In Rozenberg, G., editor, *Handbook of Graph Grammars and Computing by Graph Transformation*, pages 313–400. World Scientific Publishing.

Drewes, F., Kreowski, H.-J., and Habel, A. (1997). Hyperedge replacement graph grammars. In Rozenberg, G., editor, *Handbook of Graph Grammars and Computing by Graph Transformation*, pages 95–162. World Scientific Publishing.

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

Kepser, S. (1994). A satisfiability algorithm for a typed feature logic. Master's thesis, Seminar für Sprachwissenschaft, Universität Tübingen, Arbeitspapiere des SFB 340, Bericht Nr. 60.

Kepser, S. (2001). On the complexity of rsrl. In Kruiff, G.-J., Moss, L., and Oehrle, R., editors, *Proceedings FG-MOL 2001*, volume 53 of *ENTCS*. Kluwer.

Kepser, S. and Mönnich, U. (2003). Graph properties of hpsg feature structures. In Jäger, G., Monachesi, P., Penn, G., and Wintner, S., editors, *Formal Grammar 2003*.

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

King, P. J., Simov, K. I., and Aldag, B. (1999). The complexity of modellability in finite and computable signatures of a constraint logic for head-driven phrase structure grammar. *Journal of Logic, Language and Information*, 8(1):83–110.

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 Dauchet, M. and Nivat, M., editors, *CAAP '88, 13th Colloquium on Trees in Algebra and Programming*, volume 299 of *LNCS*, pages 28–39.

Mezei, J. and Wright, J. (1967). Algebraic automata and contextfree sets. *Information and Control*, 11:3–29.

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

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

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

Richter, F. (2000). *A Mathematical Formalism for Linguistic Theories with an Application in Head-Driven Phrase Structure Grammar*. PhD thesis, SfS, Universität Tübingen.

Robertson, N. and Seymour, P. (1986a). Graph minors II. Algorithmic aspects of treewidth. *Journal of Algorithms*, 7(309–322).

Robertson, N. and Seymour, P. (1986b). Graph minors V: Excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 52:92–114.

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

Seese, D. (1991). The structure of the models of decidable monadic theories of graphs. *Annals of Pure and Applied Logic*, 53:169–195.