*URL:* [*http://www.elsevier.nl/locate/entcs/volume53.html*](http://www.elsevier.nl/locate/entcs/volume53.html) *18 pages*

# *Observations on*

*Strict Derivational Minimalism*

*Jens Michaelis*

*Universität Potsdam, Institut für Linguistik, PF 601553, 14415 Potsdam, Germany*

*Abstract*

*Deviating from the de nition originally presented in [12], Stabler [13] introduced inspired by some recent proposals in terms of a minimalist approach to transforma- tional syntax a (revised) type of a minimalist grammar (MG) as well as a certain type of a strict minimalist grammar (SMG). These two types can be shown to de- termine the same class of derivable string languages.*

*1 Introduction*

*The type of a minimalist grammar (MG) as introduced in [12] provides an at-* tempt of a rigorous formalization of the perspectives adopted nowadays within the linguistic framework of transformational grammar. As shown in [4], this type of an MG constitutes a weakly equivalent subclass of linear context free rewriting systems (LCFRSs) [14,15]. Recently, independent work of Harkema

*[2] and Michaelis [7] has proven the reverse to be true as well. Hence, MGs*

*as de ned in [12], beside LCFRSs, join to a series of mildly context sensitive* formalism classes among which there is e.g. the class of multicomponent tree adjoining grammars (MCTAGs) in their set local variant of admitted adjunc- tion (cf. [15]) all generating the same class of string languages, which is known to be a substitution closed full AFL. 1 Mainly inspired by the linguis- tic work presented in [3], in [13] a revised type of an MG has been proposed whose departure from the version in [12] can be seen as twofold: the revised type of an MG neither employs any kind of head movement nor covert phrasal movement, and an additional restriction is imposed on the move operator as to which maximal projection may move overtly. Deviating from the operation move as originally de ned in [12], a constituent has necessarily to belong to the transitive closure of the complement relation or to be a speci er of such a constituent in order to be movable. Closely in keeping with some further

*? This work has been funded by DFG grant STA 519/1-1.*

*1 For a list of some of such classes of generating devices, beside MCTAGs, see e.g. [9].*

*c 2001 Published by Elsevier Science B. V. Open access under* [*CC BY-NC-ND license.*](http://creativecommons.org/licenses/by-nc-nd/3.0/)

*suggestions in [3], a certain type of a strict minimalist grammar (SMG) has* been introduced in [13] as well. This MG type allows only movement of con- stituents belonging to the transitive closure of the complement relation. But di erent from the rst type, the triggering licensee feature may head the head label of any constituent within the re exive transitive closure of the speci er relation of a moving constituent. Furthermore, due to the general de nition of a lexical item of an SMG, an SMG does not permit the creation of multiple speci ers in the course of a derivation. This paper answers to some important questions explicitly left open in [13]: the respective types of an MG and an SMG are shown to determine the same class of derivable string languages. This is done by proving both formalism types to be weakly equivalent to the same subclass of LCFRSs. The respective class of generated string languages is also shown to constitute a substitution closed full AFL. Whether it coin- cides with the class of all LCFRS de nable string languages remains an open problem here.

## *2 Multiple Context Free Grammars*

*LCFRSs form a proper subclass of multiple context free grammars (MCFGs)* [11], which in their turn are a subtype of generalized context free grammars [8]. But LCFRSs de ne the same class of derivable string languages as MCFGs.

*De nition 2.1 [8] A generalized context free grammar (GCFG) is a ve*

*tuple G = hN; O; F; R; Si, where N is a nite non empty set of nonterminals,* and where O is a set of (linguistic) objects. F is a nite subset of Sn2IN Fn nf;g,

*F the set of partial functions from hOin into O. 2 R is a nite set of (rewriting)* rules, i.e. a subset of S (F \ F ) hN in+1. S is a distinguished symbol

*n*

*n2IN n*

*from N , the start symbol.*

*An r = hf; hA ;A ;::: ;A ii 2 (F \F ) hN in+1 for some n 2 IN is written*

*0*

*1*

*n*

*n*

*A0 ! f (A1;::: ; An), and also A0 ! f (;) if n = 0. In case n = 0, i.e. if f is a constant in O, r is terminating, otherwise r is nonterminating.*

*For each A 2 N and k 2 IN, Lk (A) O is given recursively by means of*

*G*

*2 L0 (A) for each terminating A ! 2 R, and 2 Lk+1(A) if 2 Lk (A), or*

*G G G*

*if there are A ! f (A ;::: ;A ) 2 R and 2 Lk (A ) for 1 i n such that*

*1 n i G i*

*h ;::: ; i 2 Dom(f ) and f ( ;::: ; ) = . 3 The set L*

*(A) = S Lk (A)*

*1 n 1 n*

*G k2IN G*

*is the language derivable from A (by G). LG(S), also denoted by L(G), is the*

*language derivable by G.*

*De nition 2.2 [11] A multiple context free grammar (MCFG) is a GCFG* G = hN; O; F; R; Si with O = S h in+1, and satisfying (M1) and (M2),

*n2IN*

*2 IN is the set of all non negative integers. For n 2 IN and any sets M ;::: ;M , Q M is*

*n*

*1 n i=1 i n*

*the set of all n tuples hm1;:::; mni with i th component mi 2 Mi, where Qi=1 Mi := f;g*

*n*

*for n = 0. We write hM in instead of Q*

*i=1 Mi*

*if for some set M , Mi*

*= M for 1 i n.*

*3 For each partial function g from a set M into a set M 0 , Dom(g) M is the domain of g.*

*where is a nite set of terminals with \ N = ;. 4*

*(M1) For each f 2 F , some n(f ) 2 IN, '(f ) 2 IN n f0g and di(f ) 2 IN n f0g for*

*1 i n(f ) exist such that f is a (total) function from Qn(f )h idi(f )*

*i=1*

*into h i'(f) for which (f1) and (f2) hold.*

*(f1) Let Xf = fxij j 1 i n(f ) ; 1 j di(f )g be a set of pairwise distinct variables, for 1 i n(f ) let xi = hxi1;::: ; xidi (f ) i, and for 1 h '(f ) let fh be the h th component of f , i.e. the function from Dom(f ) into*

*such that f ( ) = hf ( );::: ;f ( )i for all 2 Dom(f ). Then, for*

*1 '(f )*

*each 1 h '(f ) there are an l (f ) 2 IN, a (f ) 2 for 0 l l (f ),*

*h hl h*

*and a z(fhl) 2 Xf for 1 l lh(f ) such that fh is represented by (cfh ). (cfh ) fh(x1;::: ; xn(f)) = (fh0) z(fh1) (fh1) z(fhlh (f )) (fhlh(f ))*

*(f2) Each x 2 Xf occurs at most once in all righthand sides of (cf1 ) (cf'(f ) ),*

*i.e. for the set IDom(f) = fhi; jij 1 i n(f ) ; 1 j di(f )g and for the*

*set IRange(f) = fh h; l ij 1 h '(f ) ; 1 l lh(f )g, the binary relation gf IDom(f) IRange(f) such that hhi; ji; hh; lii 2 gf i xij = z(fhl) is an injective partial function onto IRange(f).*

*(M2) There is a function dG from N into IN with dG(S) = 1 such that, if A0 ! f (A1;::: ; An) 2 R for some n 2 IN then '(f ) = dG(A0) and di(f ) = dG(Ai) for 1 i n.*

*The rank of G, denoted by rank (G), is the number maxfn(f ) j f 2 F g.* The language derivable by G, the set L(G), is called a multiple context free language (MCFL). Note that L(G) , because d (S) = 1.

*G*

*De nition 2.3 [14,15] An MCFG G in the sense of De nition 2.2 such that* for each f 2 F condition (f3) holds in addition to (f1) and (f2) is a called linear context free rewriting system (LCFRS). In this case L(G) is a linear context free rewriting language (LCFRL).

*(f3) Each xij 2 Xf has to appear in one of the righthand sides of (cf1 ) (cf'(f ) ), i.e. the function gf from (f2) is total, and therefore, a bijection.*

*The class of all MCFLs and the class of all LCFRLs are known to be identical* (cf. [11, Lemma 2.2]). Theorem 11 in [9], therefore, leads to

*Corollary 2.4 For each MCFG G there is a weakly equivalent LCFRS G0* with rank (G0 ) 2.

*De nition 2.5 An MCFG1;2 (LCFRS1;2 ) is an MCFG (LCFRS) G in the* sense of De nition 2.2 (De nition 2.3) such that rank (G) 2, and such that d1(f ) = 1 for each f 2 F with n(f ) = 2. In this case the language derivable by G is an MCFL1;2 (LCFRL1;2 ).

*4 For each set M , M is the Kleene closure of M , including , the empty string. M denotes the set M [ f g.*

*3 MCFGs in Monotone Function Form*

*We now introduce a special type of an MCFG, the type of an MCFG in mono-* tone function form (MFF), which will be of considerable interest in Section 6. Roughly, the idea leading to the corresponding de nition is the fact that (at least in terms of weak equivalence) synchronized parallelism in an MCFG is in a certain sense independent of the order of the constituents (each of which represented by a terminal string) that are derivable as a tuple from a given nonterminal. More technically, for a given rule r = A ! f (A1;::: ; An(f )), it is not the order of the components of a di(f ) tuple i = h i1 ;::: ; idi (f ) i derivable from the nonterminal Ai that really matters, but rather the (partial) order of these components induced by their left to right appearance within the components of the '(f ) tuple f ( 1 ;::: ; n(f )) derivable from A by means of r. Using this insight, we will focus on the possibility of an a priori re ordering of the components of a corresponding di(f ) tuple i in a particular way: it is a consequence of (f1) and (f2) that for each 1 i n(f ) there is a permuta-

*tion Æ (f ) on f1;::: ;d (f )g such that for 1 j; j0 d (f ) with j < j0 , if the*

*i i i*

*variables xij and xij0 appear at all within some component fh(x1 ;::: ; xn(f))*

*for some 1 h '(f ), these two variables are monotonically ordered by* Æi(f ) w.r.t. the function gf from (f2) in the sense that

*Æ (f )(j) < Æ (f )(j0 ) i g (i; Æ (f )(j)) < g (i; Æ (f )(j0 ))*

*i i f i IN IN f i*

*for each hi; ji, hi; j0 i 2 Dom(g ). 5 What we will rely on is that each MCFG G* can be transformed into a weakly equivalent MCFG G0 such that, in particular, for each function f in G0 , the corresponding monotonic order w.r.t. g for

*f*

*f*

*1 i n(f ) holds with Æi(f ) being the identity function on f1;::: ; di(f )g.*

*De nition 3.1 An MCFG G = hN; O; F; R; Si is in monotone function form (MFF) if for each f 2 F and i; j; k 2 IN with hi; ji; hi; ki2 Dom(gf ) it holds that j< k i gf (i; j) <IN IN gf (i; k), where gf is de ned as in (f2).*

*Proposition 3.2 For each MCFG G is a weakly equivalent LCFRS G0 which is in MFF.*

*1;2 1;2*

*Proof. By Corollary 2.2.10 and 2.4.4 of [5].*

*4 Minimalist Grammars*

*Throughout we let :Syn and Syn be a nite set of non syntactic features and* a nite set of syntactic features, respectively, in accordance with (F1) and (F2) below. We take Feat to be the set :Syn [ Syn.

*(F1) :Syn is disjoint from Syn and partitioned into a set Phon of phonetic* features and a set Sem of semantic features.

*5 0 0 0 0*

*<IN IN denotes the lexical order on IN IN, i.e. for all p; q; p ;q 2 IN, hp; qi <IN IN hp ;q i*

*0*

*0 0*

*i (a)*

*p < p or (b) p =*

*p and q < q .*

*(F2) Syn is partitioned into a set Base of (basic) categories, a set Select of* selectors, a set Licensees of licensees and a set Licensors of licensors. For each x 2 Base, usually typeset as x, the existence of a matching x0 2 Select, denoted by = x, is possible. For each x 2 Licensees, usually depicted as -x, the existence of a matching x0 2 Licensors, denoted by

*+X, is possible. Base includes at least the category c.*

*De nition 4.1 A ve tuple hN ; / ; ;< ; label i is called an expression*

*(over Feat) if it ful lls (E1) (E4).*

*(E1) hN ; / ; i is a nite, binary (ordered) tree de ned in the usual sense:* N is the nite, non empty set of nodes, and / and are the respec- tive binary relations of dominance and precedence on N . 6

*(E2) < N N is the asymmetric relation of (immediate) projection that* holds for any two siblings in hN ; / ; i.

*(E3) label is the leaf labeling function from the set of leaves of hN ; / ; i* into Syn Phon Sem .

*(E4) hN*

*; / ;*

*i is a subtree of the natural interpretation of a tree domain. 7*

*We take Exp(Feat) to denote the set of all expressions over Feat.* Let = hN ; / ; ; < ; label i 2 Exp(Feat).

*For each x 2 N , the head of x (in ), denoted by head (x), is the (unique)* leaf of with x / head (x) such that each y 2 N on the path from x to head (x) with y 6= x projects over its sibling, i.e. y < sibling (y). 8 The head of is the head of 's root. is said to be a head (or simple) if N consists of exactly one node, otherwise is said to be a non head (or complex ).

*A ve tuple = hN ; / ; ; < ; label i is a subexpression of in case*

*hN ; / ; i is a subtree of hN ; / ; i, < = < N N and label = label N .* Thus, 2 Exp(Feat). Such an is a maximal projection (in ) if 's root is a node x 2 N such that x is the root of , or such that sibling (x) < x. MaxProj ( ) is the set of all maximal projections in .

*comp MaxProj ( ) MaxProj ( ) is the binary relation de ned such* that for all ; 2 MaxProj ( ) it holds that comp i head (r) < r, where r and r are the roots of and , respectively. If comp holds for some ; 2 MaxProj ( ) then is a complement of (in ). comp is the

*+*

*+*

*transitive closure of comp . Comp+( ) is the set f j comp g.*

*6 Thus, /*

*is the re exive transitive closure of /*

*N N*

*, the relation of immediate*

*dominance on N*

*7 A tree domain is a non empty set N IN such that for all 2 IN and i 2 IN it holds that 2 N if 0 2 N for some 0 2 IN , and i 2 N if j 2 N for some j 2 IN with i < j. hN ; / ; i is the natural (tree) interpretation of N in the case that for all*

*; 2 N*

*it holds that /*

*i = i for some i 2 IN, and*

*i = !i 0 and*

*= !j 0 for some !; 0 ; 0 2 IN and i; j 2 IN with i < j.*

*8 sibling (x) denotes the (unique) sibling of any given x 2 N di erent from 's root.*

*spec MaxProj ( ) MaxProj ( ) is the binary relation de ned such that* such that for all ; 2 MaxProj ( ) it holds that spec i r = sibling (x)

*+ +*

*for some x 2 N with r / x / head (r), where r and r are the roots*

*of and , respectively. If spec for some ; 2 MaxProj ( ) then is a* speci er of (in ). spec is the re exive transitive closure of spec . Spec( )

*and Spec ( ) are the sets f j spec g and f j spec g, respectively.*

*An 2 MaxProj ( ) is said to have feature f if the label assigned to 's* head by label is non empty and starts with an instance of f 2 Feat.

*is complete if its head label is in f gPhon Sem and each other of its* leaf labels in Phon Sem . Hence, a complete expression over Feat is an ex- pression that has category c, and this instance of c is the only instance of a syntactic feature within all leaf labels.

*c*

*The phonetic yield of , denoted by YPhon ( ), is the string which results* from concatenating in left to right manner the labels assigned to the leaves of hN ; / ; i via label , and replacing all instances of non phonetic features with the empty string, afterwards.

*An = hN ; / ; ; < ; label i 2 Feat(Exp) is (label preserving) isomor- phic to if there is a bijective function i from N onto N with x / y i i(x) / i(y), x y i i(x) i(y), x < y i i(x) < i(y), and with label (x) = label (i(x)) for x; y 2 N . i is an isomorphism (from to ).*

*De nition 4.2 For = hN ; / ; ; < ; label i 2 Exp(Feat) with N = tN for some t 2 IN and some tree domain N , and for r 2 IN , ( ) denotes the expression shifting to r, i.e. the expression hN(r) ; /(r) ;(r) ; <(r) ; label(r) i over Feat with N(r) = rN such that the function i(r) from N onto N(r) with i (tx) = rx for all x 2 N is an isomorphism from to ( )r. 9*

*r*

*(r)*

*For ; 2 Exp(Feat) let = hN; /;; <; labeli be a complex expression* over Feat with root such that ( )0 and ( )1 are the two subexpressions of whose roots are immediately dominated by . Then is of one of two forms: in order to refer to we write [< ; ] if 0 < 1, and [> ; ] if 1 < 0.

*De nition 4.3 [13] A minimalist grammar (MG) is a ve tuple of the form* h:Syn; Syn; Lex ; ; ci, where Lex is a lexicon (over Feat), i.e. a nite set of simple expressions over Feat each of the form hN ; / ; ; < ; label i with N = f g and label ( ) 2 (Select [ Licensors) Base Licensees Phon Sem , and where is the operator set consisting of the structure building functions merge and move de ned w.r.t. Feat as in (me) and (mo) below, respectively.

*(me) merge is a partial mapping from Exp(Feat) Exp(Feat) into Exp(Feat).* A pair h ; i with ; 2 Exp(Feat) belongs to Dom(merge) if for some x 2 Base conditions (i) and (ii) are ful lled:

*(i) has selector = , and*

*x*

*9 Note that, by (E4), for each = hN*

*; / ;*

*; < ; label*

*i 2 Exp(Feat ) a t 2 IN and a*

*tree domain N with N = tN do exist.*

*(ii) has category x.* Then,

*(me.1) merge( ; ) = [ 0 ; 0 ] if is simple, and*

*<*

*(me.2) merge( ; ) = [ 0 ; 0 ] if is complex,*

*>*

*where 0 and 0 result from and , respectively, just by deleting the* instance of the feature that the respective head label starts with.

*(mo) move is a partial mapping from Exp(Feat) into Exp(Feat). An expres-* sion 2 Exp(Feat) is in Dom(move) if for some -x 2 Licensees condi- tions (i) (iii) are true:

*(i) has licensor feature +X,*

*(ii) there is exactly one 2 MaxProj ( ) that has feature -x, and*

*(iii) there exists a 2 Comp+( ) with = or 2 Spec( ).* Then,

*move( )= [> ; ],*

*0 0*

*where 0 2 Exp(Feat) results from by canceling the instance of +X the* head label of starts with, while the subtree is replaced by a single node labeled . 0 2 Exp(Feat) arises from by deleting the instance of

*-x the head label of starts with.*

*De nition 4.4 [13] A strict minimalist grammar (SMG) is a ve tuple of* the form h:Syn; Syn; Lex ; ; ci, where Lex is a nite set of expressions over Feat each of the form hN ; / ; ; < ; label i with N = f g and label ( ) is in Select (Select [ Licensors) Base Licensees Phon Sem , and where is the operator set consisting of the structure building functions merge and moves de ned w.r.t. Feat as in (me) above and (smo) below, respectively.

*(smo) moves is a partial mapping from Exp(Feat) into Exp(Feat). An ex-* pression 2 Exp(Feat) is in Dom(move) if for some -x 2 Licensees conditions (i) (iii) are true:

*(i) has licensor feature +X,*

*(ii) there is exactly one 2 MaxProj ( ) that has feature -x, and*

*(iii) there exists a 2 Comp+( ) with 2 Spec ( ). 10* Then,

*moves ( )=[ 0 ; 0 ],*

*>*

*where 0 2 Exp(Feat) results from by canceling the instance of +X the* head label of starts with, while the subtree is replaced by a single node labeled . 0 2 Exp(Feat) arises from by deleting the instance of -x the head label of starts with.

*10 Note that such a 2 Comp+ ( ) is unique.*

*For each (S)MG G = h:Syn; Syn; Lex ; ; ci the closure of G, CL(G), is the* set S CLk(G), where CL0(G)= Lex , and for k 2 IN, CLk+1(G) Exp(Feat)

*k2IN*

*is recursively de ned as the set*

*CLk(G) [ fmerge( ; ) j h ; i2 Dom(merge) \ CLk(G) CLk(G)g* [ fmove0 ( ) j 2 Dom(move0 ) \ CLk(G)g,

*where move0 2 n fmergeg. L(G) denotes the (string) language derivable by* G, i.e. the set fYPhon ( ) j 2 CL(G) and completeg.

*De nition 4.5 A set L is a (strict) minimalist language ((S)ML) if there exists an (S)MG G with L = L(G).*

# *5 (S)MLs as MCFLs*

*A method of transforming an MG as de ned in [12] into an MCFG is presented* in [4]. As demonstrated in [5], this method can be adapted to transform an (S)MG as de ned in [13] into an MCFG1;2. But note that this adaptation is not of trivial kind, since in the original MG de nition move was de ned as in (mo) above, but without condition (iii), i.e. a maximal projection could move completely independently of its position within an expression. Also, the handling of derivable tuples by means of the rewriting rules and functions has to be changed rather signi cantly in order to arrive at an MCFG as desired. 11

# *6 MCFLs as (S)MLs*

*Throughout this section, G = hN; O; F; R; Si denotes an MCFG1;2 in the sense* of De nition 2.5. In order to de ne an MG GMG = h:Syn; Syn; Lex ; ; ci in the sense of De nition 4.3 such that L(GMG ) = L(G), we suppose w.l.o.g. G to be an LCFRS1;2 in MFF (cf. Proposition 3.2).

*Of course, in [2] and [7] respective methods are presented how to construct,* for an arbitrary MCFG, a weakly equivalent MG of the type originally given in [12]. Starting from an MCFG1;2, w.r.t. each of both methods, the lexicon of the resulting MG can even be interpreted as the lexicon of an MG in the sense of De nition 4.3 without leading to a change in the closure of the lexicon under the structure building functions. 12 A di erence in the closure of the lexicon under the structure building functions may arise, however, if the lexicon of the MG resulting from the construction according to [7] is interpreted as the

*11 The respective considerations in [5] are even somewhat more involved than it would be necessary as to our concerns here: there, a corresponding transformation is given w.r.t. a type of an (S)MG which, in contrast to the de nition in [13], still allows (overt) head movement and covert phrasal movement to take place. The plain case of transforming an MG in the proper sense of [13] into an MCFG1 2 is considered in [6].*

*12 As far as the approach presented in [2] is concernced some slight modi cations of the*

*;*

*original construction are actually necessary before.*

*lexicon of an SMG, i.e. if the operator move is replaced by the operator moves* in order to build the corresponding closure. 13 This is not possible w.r.t. the MG GMG which we develop here, since it ful lls (a) and (b) of Proposition 6.5, implying that the language derivable by GMG is also an SML (Corollary 6.4). This result yields the interesting consequence that the class of MLs and that of SMLs are identical, con rming the corresponding conjecture in [13].

*Motivating the construction below, let A ! f (A1;::: ; An(f)) 2 R and pi 2 LG(Ai) for 1 i n(f ), hence, p = f (p1;::: ; pn(f)) 2 LG(A): our aim is to de ne GMG such that there is some 2 CL(GMG ) derivable from some expressions 1;::: ; n(f ) 2 CL(GMG ), thereby successively calculating*

*the '(f ) tuple p in n(f )+ 3'(f )+ P'(f ) 2l (f ) steps. 14 Each expression ,*

*h=1*

*h*

*i*

*for 1 i n(f ), will be related to Ai and pi, and the resulting expression to* A and p in a speci c way (cf. De nition 6.1). Roughly speaking, as for , for each 1 h dG(A) there is a h 2 MaxProj ( ) that has a particular licensee, and up to the phonetic yields of the proper subtrees potentially extractable from h, p's component ph is the phonetic yield of h.

*Let Phon = and Sem = ;.*

*For 1 h m and 0 n 1 let -lhh; ni be a licensee and +Lhh; ni the match- ing licensor such that Licensees and Licensors both have cardinality 2m. 15*

*For each A 2 N introduce new, pairwise distinct basic categories ã and a as well as corresponding selectors = ã and = a for 1 h d (A). For each A ! f (A1;::: ; An(f)) 2 R introduce new, pairwise distinct basic cate- gories a and a as well as corresponding selectors = a*

*h h G*

*hf; '(f )+1; 0i hf; h; li hf; '(f )+1; 0i*

*and = a , where 1 h '(f ) and 0 l l (f ). 14 Finally, assume c 2 Base*

*hf; h; li h*

*to be di erent from all other elements in Base.*

*Next we de ne Lex Exp(:Syn [ Syn). 16 The rst item de ned to belong to Lex is*

*= = s c ,*

*c*

*where s 2 Base is the corresponding category arising from S 2 N . The form*

*of all other items in Lex depends on the rules in R. We distinguish three cases.*

*Case 1 . A ! f (B; C) 2 R for some A, B, C 2 N and f 2 F . In this case '(f ) = dG(A), n(f ) = 2, d1(f ) = dG(B) = 1 and d2(f ) = dG(C). The following elements belong to Lex :*

*=*

*= c a*

*hA; f; B; Ci 1 hf; '(f )+1; 0i*

*13 Note that the use of multiple speci ers is of rather constitutive moment within the ap- proach of Harkema [2], causing that, in particular, the MG lexicon which results from an MCFG1 2 according to his construction does generally not match the SMG de nition.*

*14 Recall that '(f ) = dG(A) by (M3).*

*;*

*15 Here, m = maxfdG(A) j A 2 N g.*

*16 Since it is a head with root , we identify a lexical item with its (unique) label.*

*For each 1 h '(f ) with lh(f ) = 0 we add*

*=*

*= a a -l (f ) .*

*hA; f; h; 0i hf; h+1; 0i hf; h; 0i hh; 0i h0*

*For each 1 h '(f ) with lh(f ) > 0 we add*

*=*

*= a a -l (f )*

*hA; f; h; 0i hf; h; 1i hf; h; 0i hh; 0i h0*

*> ahf; h+1; 0i*

*8*

*=*

*=*

*hf; h; l (f )i hl (f ) hl (f ) 11*

*b a (f ) if z(f ) = x*

*hA; f; h; l (f )i =*

*h*

*>>< = a*

*h*

*+L a*

*hf; h+1; 0i*

*hj; 1i*

*hf; h; lh(f )i*

*h*

*(f*

*h*

*) otherwise,*

*hlh(f )*

*>: where 1 j d2(f ) with z(fhl (f)) = x2j 17*

*h*

*For each 1 h '(f ) and 1 l < lh(f ) we add*

*> ahf; h; l+1i*

*>< a +L a (f ) otherwise,*

*8*

*=*

*= b a (f ) if z(f ) = x*

*hA; f; h; li =*

*hf; h; li hl hl 11*

*> =*

*hf; h; l+1i hj; 1i hf; h; li hl*

*>: where 1 j d2(f ) with z(fhl(f)) = x2j 17*

*Case 2 . A ! f (B) for some A; B 2 N and f 2 F . In this case '(f ) = dG(A), n(f ) = 1 and d1(f ) = dG(B). Then, the following element belongs to Lex :*

*=*

*= b a*

*hA; f; B; i 1 hf; '(f )+1; 0i*

*Case 3 . A ! f () for some A 2 N and f 2 F . Then '(f )= dG(A) and n(f )= 0. lh(f )= 0 for 1 h '(f ), i.e. f ()= h (f10);::: ; (f'(f )0)i, since f is a constant in hi'(f ) . The following entry belong Lex :*

*hA; f; ; i = ahf; '(f )+1; 0i*

*In Case 2 and 3 , also the following items are in Lex : For each 1 h '(f ) with lh(f ) = 0 we just add*

*= a a -l (f ) .*

*=*

*hA; f; h; 0i hf; h+1; 0i hf; h; 0i hh; 0i h0*

*For each 1 h '(f ) with lh(f ) > 0 we add*

*hA; f; h; 0i = = ahf; h; 1i ahf; h; 0i -lhh; 0i (fh0)*

*hA; f; h; lh(f )i = = ahf; h+1; 0i +Lhj; 1i ahf; h; lh(f )i (fhlh(f )) ,*

*where 1 j d1(f ) such that z(fhlh (f )) = x1j.*

*17 Since d1 (f ) = 1, such a j exists and is unique.*

*For each 1 h '(f ) and for 1 l < lh(f ) we add*

*=*

*= a +L a (f ) ,*

*hA; f; h; li hf; h; l+1i hj; 1i hf; h; li hl*

*where 1 j d1(f ) such that z(fhl) = x1j.*

*In Case 1 3 , nally the following items are in Lex :*

*=*

*= a +L a -l ,*

*hA; '(f )i hf; 1; 0i h'(f ); 0i '(f ) h'(f ); 1i*

*for each 1 h < '(f ) the simple expression*

*=*

*= a +L a -l ,*

*hA; hi h+1 hh; 0i h hh; 1i*

*and as last item the expression*

*ehA; 1i = <8*

*:*

*=*

*ahf; 1; 0i*

*+Lh1; 0i*

*=*

*a*

*2*

*+Lh1; 0i*

*ã otherwise*

*ã if '(f )= 1*

*De nition 6.1 For each A 2 N and each p = h 1;::: ; d (A)i with i 2 for 1 i dG(A) an expression 2 CL(GMG ) is said to correspond to the pair hA; pi if (Y1) (Y4) are ful lled, where 1 = .*

*G*

*(Y1) 's head label is of the form a1-lh1; 1i h1; 1i or ã h1; 1i for a h1; 1i 2. (Y2) For each 2 h d (A) there is exactly one 2 Comp+( ) whose*

*G h*

*head label is of the form -lhh; 1i hh; 1i for some hh; 1i 2. (Y3) For each 1 h dG(A) it holds that*

*f 2 MaxProj ( h) n f hgj has some licenseeg = f i j h < i dG(A)g;*

*i.e. for each 1 h < dG(A) the subexpression h+1 is the unique maximal* maximal projection in h that has some licensee feature.

*(Y4) For each 1 h dG(A) the string h is the phonetic yield of h. Here we have dG (A) = dG (A), and for 1 h < dG(A) the expression h results from h by replacing the subtree h+1 with a single node labeled .*

*Proposition 6.2 For each 2 CL(GMG ) that has category feature a1 or ã for some A 2 N , there is some p 2 LG(A) such that corresponds to hA; pi.*

*Proof (sketch). We exclude the trivial case by assuming that there is some*

*2 CL(GMG ) such that has category a1 or ã for some A 2 N . We take* K 2 IN to be the smallest number, thereby existing, for which CLK(G ) includes such a . The de nition of Lex leaves us with the fact that K > 0.

*MG*

*The proof follows from an induction on k 2 IN with k +1 K.*

*For some k 2 IN with k + 1 K consider 2 CLk+1(G ) n CLk(G ) such that has category a1 or ã for some A 2 N . By de nition of Lex the procedure to derive as an expression of GMG is deterministic in the following sense: there are some r = A ! f (A1;::: ; An(f)) 2 R, some k0 2 IN with*

*MG MG*

*k = k + 1 3'(f ) P'(f ) 2l*

*(f ) and some*

*2 CLk0 (G ) such that*

*0 h=1 h*

*0 MG 0*

*serves to derive in GMG . 0 has category feature ahf; '(f )+1; 0i and is of one*

*of three forms depending on r:*

*Case 1 . There is some r = A ! f (B; C) 2 R, there is some 2 CL (GMG ) for some k0 k1 k, and there is some 2 CLk0 (G ) such that and have category feature b and c1, respectively, and*

*k1*

*MG*

*0 = merge( hA; f; B; Ci; ).*

*More explicitly, k1 can be speci ed by*

*1 0 Ph 1*

*( )*

*k = k*

*+ 2l +1+ h +*

*h0 =0 2l' f*

*h0 (f )*

*for 0 h < '(f ) and 0 l < l'(f) h(f ) such that g 1('(f ) h; l'(f) h(f ) l) = h1; 1i.*

*f*

*By induction hypothesis there are some pB 2 LG(B) and pC 2 LG(C) such*

*that and correspond to hB; pBi and hC; pCi, respectively. In this case we de ne p 2 LG(A) by p = f (pB; pC). Note that pB 2 by assumption on G.*

*Case 2 . There are some r = A ! f (B) 2 R and 2 CL (GMG ) such that has category feature b1, and such that*

*k0*

*0 = merge( hA; f; B; i; ).*

*By induction hypothesis there is some pB 2 LG(B) such that corresponds to hB; pB; 1i. Let p = f (pB) 2 LG(A).*

*Case 3 . There is some r = A ! f () 2 R and 0 is a lexical item, namely,*

*0 = hA; f; ; i.*

*In this case we simply let p = f () 2 LG(A).*

*If k +1 = K (the base case of our induction) then 0 is necessarily of the last*

*form by choice of K. We also see that the given 2 CLk+1(G ) n CLk(G )*

*MG*

*MG*

*corresponds to hA; pi in any case. The single derivation steps in order to end* up with starting from 0 are explicitly given by the following procedure:

*Procedure (derive from 0).*

*For 0 h < '(f )*

*hh+1; 0i = h*

*for 0 l < l'(f) h(f )*

*step 2l +1+ h + Ph 1 2l*

*h0 =0*

*'(f ) h0*

*(f )*

*hh+1; 2l+1i = merge ( hA; f; '(f ) h; l'(f ) h (f ) li; hh+1; 2li)*

*step 2l +2+ h + Ph 1 2l*

*h0 =0*

*'(f ) h0*

*(f )*

*if '(f ) = 2, and if g 1('(f ) h; l'(f) h(f ) l) = h1; 1i then*

*f*

*hh+1; 2l+2i = merge ( hh+1; 2l+1i; )*

*else*

*hh+1; 2l+2i = move ( hh+1; 2l+1i)*

*[ checks licensee -lhj; 1i with gf (2; j) = h'(f ) h; l'(f) h(f ) li ]*

*h+1 = merge ( hA; f; '(f ) h; 0i; hh+1; 2l'(f ) h(f )i) For 0 h < '(f ) 1*

*step h +1+ Ph*

*h0 =0*

*2l*

*'(f ) h0*

*(f )*

*step 2h +1+ '(f )+ P'(f ) 2l*

*h0 =1*

*h0 (f )*

*'(f )+2h+1 = merge ( hA; '(f ) hi; '(f )+2h)*

*step 2h +2+ '(f )+ P'(f ) 2l*

*h0 =1*

*h0 (f )*

*'(f )+2h+2 = move ( '(f )+2h+1) [ checks licensee -lh'(f ) h; 0i ]*

*Either*

*step 3'(f ) 1+ P'(f ) 2l*

*h0 =1*

*h0 (f )*

*'(f )+2'(f ) 1 = merge( hA; 1i; 3'(f ) 2)*

*step 3'(f )+ P'(f ) 2l*

*h0 =1*

*h0 (f )*

*3'(f ) = move( 3'(f ) 1) [ checks licensee -lh1; 0i ]*

*or*

*step 3'(f ) 1+ P'(f ) 2l*

*h0 =1*

*h0 (f )*

*3'(f ) 1 = merge( ehA; 1i; 3'(f ) 2)*

*step 3'(f )+ P'(f ) 2l*

*h0 =1*

*h0 (f )*

*3'(f ) = move( 3'(f ) 1) [ checks licensee -lh1; 0i ]*

*= 3'(f )*

*An embedded induction on 0 h < '(f ) and 0 l < l'(f) h(f ) yields that indeed corresponds to hA; pi. We omit further details at this point.*

*Proposition 6.3 Let A 2 N and p 2 hidG (A) with p 2 LG(A). Furthermore let x 2 fa1; ãg. Then there is some 2 CL(GMG ) that has category feature x such that corresponds to hA; pi.*

*Proof (sketch). Let A 2 N and p 2 hidG (A) such that p 2 LG(A). Then,* again w.l.o.g., we are concerned with one of three possible cases.

*Case 1 . There is some r = A ! f (B; C) 2 R, and for some k 2 IN there are some pB 2 Lk (B) and pC 2 Lk (C) such that p = f (pB; pC) 2 Lk+1(A)nLk (A).*

*G G G G*

*By hypothesis on k there are some ; 2 CL(GMG ) such that and have*

*category feature b and c1, respectively, and such that and correspond to*

*hB; pBi and hC; pCi, respectively. We can therefore de ne 0 2 CL(GMG ) by*

*0 = merge( ; merge( hA; f; B; Ci; ).*

*Note that we have dG(B) = 1 by assumption on G.*

*Case 2 . There is some r = A ! f (B; C) 2 R, and for some k 2 IN there is*

*some p*

*2 Lk (B) such that p = f (p ) 2 Lk+1(A) n Lk (A). Here, by induction*

*hypothesis we can choose some 2 CL(GMG ) such that has category feature* b1 and corresponds to hB; pBi. Then we de ne 0 2 CL(GMG ) by

*B G B G G*

*0 = merge( hA; f; B; i; ).*

*Case 3 . There is some r = A ! f () 2 R such that p = f () 2 LG(A). In this*

*0*

*case 0 is taken to be the lexical item de ned by*

*0 = hA; f; ; i.*

*In any case we may refer to the proof of the last proposition, claiming that* there is some derivation in GMG in which 0 serves to derive a 2 CL(GMG ) which has the demanded properties.

*Corollary 6.4 2 L(G) i 2 L(GMG ) for each 2.*

*Proof. To see that the if part holds, consider 2 CL(GMG ) which is com-* plete, and whose phonetic yield is for some 2. By de nition of Lex

*there is some 0 2 CL(GMG ) which has category s such that = merge( c; 0).*

*By Proposition 6.2 there is some p0 2 LG(S) = L(G) such that 0 corresponds* to hS; p0i. Because dG(S) = 1, and because the phonetic yield of c is empty, we have p0 = .

*In order to prove the only if part, assume that 2 L(G) = LG(S) for* some 2. By Proposition 6.3 there is some 0 2 CL(GMG ) with category feature s such that 0 corresponds to hS; i. Then, because it holds that dG(S) = 1, = merge( c; 0) is de ned and complete, and is the phonetic yield of .

*Proposition 6.5 GMG ful lls (a) and (b).*

*(a) For each 2 Lex , the (unique) label of is in particular an element of* Select (Select [ Licensors) BaseLicenseesPhonSem, since it is even of the form s1s2x for some s1 2 Select , s2 2 (Select [ Licensors) , x 2 Base,

*2 Licensees and 2 :Syn.*

*(b) Whenever, for a given 2 CL(GMG ) and -x 2 Licensees, there is some*

*2 MaxProj ( ) that has licensee -x then 2 Comp+( ).*

*Proof (sketch). Property (a) is true due to the de nition of Lex .*

*The validity of (b) arises from the combination of several facts. First, each* expression 2 CL(GMG ) serves to derive a complete expression of GMG . In

*this sense we may say that CL(GMG ) contains no useless expressions. This* in turn implies that each 2 CL(GMG ) is subject to one of the following possibilities:

*(i) 2 Lex .*

*(ii) There are some k 2 IN and r = A ! f (A1;::: ; An(f)) 2 R, and some*

*h*

*2 CLk+1(G*

*MG*

*) and 0*

*2 CLk0 (G*

*) with k0*

*= k 3'(f ) P'(f ) 2l*

*(f )*

*such that 0 serves to derive according to a respective procedure from*

*h=1*

*MG*

*above, and such that is derived within this procedure in order to*

*nally end up with .*

*(iii) is complete, i.e. there is some 0 2 CL(G ) that has category feature*

*MG*

*s such that = merge( c; 0 ).*

*As far as expressions of GMG which result from an application of the merge operator are concerned, property (b) is therefore guaranteed by the fact that* an expression which is merged into a speci er position contains no maximal projection that has any licensee feature, since this expression has category feature b for some B 2 N with dG(B) = 1.

*As it regards expressions that result from an application of the move operator, property (b) essentially results from our assumption that G is in* MFF, and from a further fact concerning the licensees of the form -lhh; 0i for some 1 h m: whenever for some expression 2 CL(GMG ) and some

*1 h m there is some hh; 0i 2 MaxProj ( ) that has licensee -lhh; 0i, each*

*hi; 0i 2 MaxProj ( ) that has some licensee -lhi; 0i with h < i m belongs to Comp+( ).*

*hh; 0i*

*Corollary 6.6 The language L(GMG ) is an ML as well as an SML.*

*Proof. This corollary is an immediate consequence of the preceding propo-* sition. Note, in particular, that from (b) of Proposition 6.5 it follows that the closure of Lex under the structure building operators is the same set of expressions over :Syn [ Syn independently of whether the move operator is de ned as in (mo) or (smo).

## *7 AFL Properties*

*In [13] three further problems concerning the properties of the revised MG type have been explicitly left open. All three fall under a more general ques-* tion: does the class of MLs constitute an abstract family of languages (AFL)? In fact a stronger result holds:

*Proposition 7.1 The class of all MLs is a substitution closed full AFL.*

*Proof. Because the class of all MLs is, as shown above, identical to the class* of all MCFL1;2's, we can likewise prove that the latter is a substitution closed full AFL: it straightforwardly follows from the de nition that the class of all

*MCFL1 2's includes all regular sets and is closed under substitution. Thus, by* Theorem 1.6 [10, p. 129] it remains to con rm that the class of all MCFL1 2's is closed under intersection with regular sets. But exactly this is done implicitly within the proof which Seki et al. [11] give verifying their Theorem 3.9.

*;*

*;*

*References*

*[1] Philippe de Groote, Glyn Morrill, and Christian Retoré, editors. Logical Aspects of Computational Linguistics (LACL '01), Lecture Notes in Arti cial Intelligence Vol. 2099. Springer, Berlin, Heidelberg, 2001.*

*[2] Henk Harkema. A characterization of minimalist languages. In de Groote et al. [1], pages 193 211.*

*[3] Hilda Koopman and Anna Szabolcsi. Verbal Complexes. MIT Press, Cambridge, MA, 2000.*

*[4] Jens Michaelis. Derivational minimalism is mildly context sensitive. In M. Moortgat, editor, Logical Aspects of Computational Linguistics (LACL '98), Lecture Notes in Arti cial Intelligence Vol. 2014, pages 179 198. Springer, Berlin, Heidelberg, 2001. Also available at* [*http://www.ling.uni-potsdam.*](http://www.ling.uni-potsdam/) *de/~michael/papers.html.*

*[5] Jens Michaelis. On Formal Properties of Minimalist Grammars. PhD thesis, Potsdam University, Potsdam, 2001.*

*[6] Jens Michaelis. Some formal implications of a revised perspective on minimalist grammars. Draft, Potsdam University, 2001. Available at http://www.ling. uni-potsdam.de/~michael/papers.html.*

*[7] Jens Michaelis. Transforming linear context free rewriting systems into minimalist grammars. In de Groote et al. [1], pages 228 244. Also available at* [*http://www.ling.uni-potsdam.de/~michael/papers.html.*](http://www.ling.uni-potsdam.de/~michael/papers.html)

*[8] Carl J. Pollard. Generalized Phrase Structure Grammars, Head Grammars, and Natural Language. PhD thesis, Stanford University, Stanford, CA, 1984.*

*[9] Owen Rambow and Giorgio Satta. Independent parallelism in nite copying parallel rewriting systems. Theoretical Computer Science, 223:87 120, 1999.*

*[10] Arto Salomaa. Formal Languages. Academic Press, New York, NY, 1973.*

*[11] Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. On multiple context free grammars. Theoretical Computer Science, 88:191 229, 1991.*

*[12] Edward P. Stabler. Derivational minimalism. In C. Retoré, editor, Logical Aspects of Computational Linguistics (LACL '96), Lecture Notes in Arti cial Intelligence Vol. 1328, pages 68 95. Springer, Berlin, Heidelberg, 1997.*

*[13] Edward P. Stabler. Remnant movement and complexity. In G. Bouma, G.*

*J. M. Kruij , E. Hinrichs, and R. T. Oehrle, editors, Constraints and Resources in Natural Language Syntax and Semantics, pages 299 326. CSLI Publications, Stanford, CA, 1999.*

*[14] K. Vijay Shanker, David J. Weir, and Aravind K. Joshi. Characterizing structural descriptions produced by various grammatical formalisms. In 25th Annual Meeting of the Association for Computational Linguistics (ACL '87), Stanford, CA, pages 104 111. ACL, 1987.*

*[15] David J. Weir. Characterizing Mildly Context Sensitive Grammar Formalisms. PhD thesis, University of Pennsylvania, Philadelphia, PA, 1988.*

*18*