Morphology consuming Syntax' Resources: Generation and Parsing in a Minimalist Version of Distributed Morphology

Jochen Trommer

Institut fuer Linguistik/Allgemeine Sprachwissenschaft
Universitaet Potsdam Postfach 601553 D-14415 Potsdam

Abstract

Distributed Morphology (DM) as presented in Halle & Marantz (1993) shows a bewildering variety of rule types. In this paper I present a formalization of DM in which the main part of its rule inventory is reduced to one single operation: Vocabulary Insertion. Extending proposals by Noyer (1997) and Halle (1997) vocabulary insertion is assumed to be iterable and to consume featural resources, whenever it is applicated. I show that this interpretation gives rise to simple algorithms for the generation and parsing of DM expressions.

1 Introduction

In DM syntactic derivations operate on lexical items without phonological content. At some point in the derivation ("Spell-Out") a copy of the actual syntax tree is made and delivered to the morphological component (Morphological Structure, MS) which modifies it in several respects, supplies the lexical items with phonological content and thus creates the input for phonology. MS has roughly the following structure:

1. Semantically non interpretable nodes like AGR heads are inserted.
2. Terminal nodes are further manipulated. Features are deleted ('Impoverishment'), or split off into separate nodes etc.
3. Phonological specified 'vocabulary items' (*VIs*) are inserted into the terminal nodes.

4. Morpho-phonological readjustment rules modify the inserted material.

Here, I will not have to say much about points 1 and 4, but I will argue that all rules that belong to 2 can be subsumed under a generalized formalization of vocabulary insertion. In section 2 I'll give some data from Classical Arabic that will serve to illustrate the working of DM and the formalization that follows in section 3. This formalization is also intended as a generation algorithm for DM. In sections 4 to 7 it will be shown that the core operations of DM are special cases of the new definition of vocabulary insertion or that they are empirically unnecessary. An algorithm for parsing DM is presented in section 8, making crucial use of the feature-consuming nature of vocabulary insertion. Some theoretical consequences of the formalization are considered in section 9. Finally (section 10) I discuss some limitations and possible ways to overcome them.

2 Classical Arabic

As an illustration for the working of DM I give a short analysis of some classical Arabic data, namely a fragment of the jussive verb paradigm (Halle, 1997)[1]:

(1)

	Singular	Dual	Plural
1	*?-aktub*	*n-aktub*	*n-aktub*
2m	*t-aktub*	*t-aktub-aa*	*t-aktub-uu*
3m	*y-aktub*	*y-aktub-aa*	*y-aktub-uu*

In terms of Halle & Marantz(1993) these data suggest the following analysis: An agreement node is introduced onto which the features of the subject are copied. An impoverishment rule deletes the distinction between plural and dual, i.e. the value +dl in the 1st person. Person and number of the agreement node are fissioned into two separate X^0s. Finally vocabulary items from the following list are inserted:

[1] 1 = 1st person, 2m = 2nd person masculine, 3m = 3nd person masculine. 2nd and 3rd person feminine forms are omitted.

(2)

/ʔ-/	[+1 -3 -pl]
/n-/	[+1 -3 +pl]
/t-/	[-1 -3]
/y-/	[+3]
/-aa/	[+pl +dl]
/-uu/	[+pl]
/aktub/	[+aktub]

Note that the *VIs* are underspecified. Only a *VI* that subsumes the relevant node can be inserted. In the case of multiple matching *VIs* the one that comes first in the list is preferred. More specific *VIs*, i.e. those with more feature specifications always precede less specified ones. Derivations for the dual forms are schematically depicted in (3):

(3)

	1 Dual	2 Dual		3 Dual	
AGR Insertion	[+1 -3 +pl +dl]	[-1 -3 +pl +dl]		[-1 +3 +pl +dl]	
Impove-rishment	[+1 -3 + pl]	[-1 -3 +pl +dl]		[-1 +3 +pl +dl]	
Fission	[+1 -3 + pl]	[-1 -3]	[+pl +dl]	[-1 +3]	[+pl +dl]
Vocabulary Insertion	*n-*	*t-*	*-aa*	*y-*	*-aa*

3 A formalization of Vocabulary Insertion

3.1 Syntactic input

The basic units of syntactic computation are lexical items which are represented as feature structures (*FSs*), i.e. sets of atomic feature value pairs, e.g. {(1 +)(3 -)(pl +)}[2] I assume that MS doesn't spell out whole sentences but rather maximal 'chunks' of X^0s, corresponding roughly to words in lexicalist theories. The linear ordering inside these chunks is an effect of morphological operations while the ordering of chunks with respect to each other presumably follows more general principles. For the sake of simplicity only the most simple type of X^0 chunk is considered namely binary trees

[2]Prefix notations like "+1", where the value precedes the feature are assumed to be simply abbreviations for (feature value) structures like "(1 +)".

where one daughter is an X^0 and the other is an X^0 or a binary tree of the same type. More formally such a tree is implemented as a set:

(4) a. When F is a lexical item the set $\{ F \}$ is an input tree.
b. When F is a lexical item and T is an input tree then the set $\{ F, T \}$ is an input tree.

A simple version of c-command can be defined over such trees:

(5) a. A set L immediately contains a lexical item F if and only if F is a member of L. L contains F iff L immediately contains F or a member of L contains F.
b. A lexical item L_1 c-commands a lexical item $L_2 \neq L_1$ if and only if L_1 is immediately contained by an input tree T that contains L_2.

It is easy to see that this notion of c-command establishes a complete linear order on the lexical items in an input tree. So we can represent such a tree without loss of information by a simple list of lexical items which I will call *input list*. For practical reasons I reverse the ordering of input trees in these lists so that the highest *FS* in the tree will be the last element in the list , and the deepest embedded one the first element.

As an example assume that the input for the 1sg jussive form of Arabic is the input tree {{+ agr +1 -pl -dl}, { { +tense + jus }, { { +v } }}} The corresponding input list is [{ +v }, { +tense + jus }, {+ agr +1 -pl -dl}] .

3.2 The structure of *VIs*

A *VI* is a 4-tuple (*Phon Context Target Deletes*), where *Phon* is a characterization of the morphological/phonological properties of the item.The *Context* component characterizes the context, i.e. the structurally adjacent *FSs* that have to be present for the item to be inserted. *Target* encodes the necessary features of the target *FS* where insertion can take place and *Deletes* enumerates the features which are deleted when the *VI* is inserted. *Phon* values in *VIs* have the structure *(Cat P)* where $Cat \in$ { pref, suff, stem } and P is a string of phonemes (possibly of length 0). *Target* and *Deletes* are feature structures as defined above, where *Deletes* subsumes *Target* and *Deletes* is nonempty. *Context* is an ordered pair of feature structures *(Left_Context Right_Context)*, where *Left_Context* denotes the *FS* that stands immediately before the target *FS* in the input list and *Right_Context* the one that immediately follows it. The *VIs* from section 2 can then be rewritten as follows:

(6)

Phon	*Context*	*Target*	*Deletes*
((pref ?)	({}{})	{(1 +)(3 -)(pl -)}	{ (1 +)(3 -)(pl -))}
((pref n)	({}{})	{(1 -)(3 -)(pl +))}	{(1 -)(3 -))}
.			
.			
.			

3.3 Vocabulary Insertion

The rough structure of derivation is the following: Spell out the first element in the list. Take the result of this and spell out the second element by adding the resulting affixal material to it, and so on. More formally a state in the derivation is given by a string of phonemes (*String*) standing for the cumulative result of the spell-out process and a pointer that indicates the actual lexical item in the input list to spell out (*L_Pointer*). Two further pointers record the actual left and right context (*LC_Pointer, RC_Pointer*):

(7) **SPELL_OUT(***Input_List***)**

set *String* **to** ϵ (empty string)
set *L_Pointer* **to** the first element of *Input_List*
set *LC_Pointer* **to** [ ] (empty feature structure)

while *L_Pointer* $\neq$ END [3]
  if the next element from *L_Pointer* (*Next*) $\neq$ END
    set *RC_Pointer* **to** *Next*
  else
    set *RC_Pointer* **to** [ ]
  ITEM_SPELL_OUT(*String,LC_Pointer,L_Pointer,RC_Pointer*)
  set *LC_Pointer* **to** *L_Pointer* **set** *L_Pointer*
  to the next List element

[3] I assume that the last element of each list is the formal element END.

ITEM_SPELL_OUT searches in the vocabulary list for the first *VI* that matches the actual X^0 (and its contexts) and inserts the *VI* (VI_INSERT). Then it searches the rest of the list (i.e. the items after the inserted *VI*) for a second matching *VI* and continues this process until no further matching *VI* is found.

(8) **ITEM_SPELL_OUT(***String,LC,B,RC***)**

set *VI_Pointer* **to** the first element of VI_LISTE
while FIRST_MATCH(*VI_Pointer, LC, B, RC*) $\neq$ END
    VI_INSERT(*B, VI_Pointer, String*)

(9) **FIRST_MATCH(** *VI_Pointer, LC, B, RC* **)**

while *VI_Pointer* $\neq$ END
    if Left_Context(*VI_Pointer*) subsumes *LC*
        if Right_Context(*VI_Pointer*) subsumes *RC*
            if Target(*VI_Pointer*) subsumes *B*
                return(*VI_Pointer*)
    set *VI_Pointer* **to** the next list Element
return(END)

The insertion procedure has two main effects. First, it puts the phonological pieces supplied by the *VIs* in the correct place. Secondly, the features indicated the *VIs* 'Deletes' section are deleted in the actual X^0.

(10) **VI_INSERT(** *B, VI_Pointer, String* **)**

delete all features in B that are specified in delete(*VI_Pointer*)

if (*String* $= \epsilon$) **and** (Status(*VI_Pointer*) = Stem)
  set *String* **to** P

else if (*String* $\neq \epsilon$)**and** (Status(*VI_Pointer*) = Pref)
  set *String* **to** $P\hat{}\ String$

else if (*String* $\neq \epsilon$) **and** (Status(*VI_Pointer*) = Suff)
  set *String* **to** $String\hat{}\ P$

where P = Phon(*VI_Pointer*)

Here is an example derivation for the 1st plural form *n-aktub*, that shows how feature deletion blocks multiple insertion in certain cases:

(11) [+aktub] [+ agr +1 +pl -dl]

First [+ aktub] is spelled out. This means that the last *VI* is found, *String* will be set to aktub and [+ aktub] to [ ]. Since this *FS* isn't subsumed by any *VI* in the remaining list spell-out of this item is finished and the second element is spelled out. the first matching *VI* for [+ agr +1 +pl -dl] is number2 in the list (2). *n-* is prefixed. Again all features are deleted and no more *VI* is to be found. Since this is the last element in the input list SPELL_OUT terminates.

4 Fission is Vocabulary Insertion

As argued for in Noyer(1992) and Halle(1997) fission can be interpreted as multiple insertion of *VIs* in terminal nodes. The possibility of such an analysis is already contained in our formalism. E.g. the generation of Arabic *t-aktub-aa* proceeds as follows.

(12) [+aktub] [+ agr +2 +pl +dl]

The string *aktub* is derived from the first *FS* like in (11). Then [+ agr +2 +pl +dl] is spelled out. The first *VI* found (3 in (2)) leads to prefixation of *t-* (*t-aktub*) and deletion of the person feature.We get [+agr +pl +dl]. In the second cycle *-aa* can be inserted (*VI* 5) which by further feature deletion leads to [+agr]. No further *VI*_Insertion is possible.

5 Impoverishment is Vocabulary Insertion

Under the advocated analysis vocabulary insertion and impoverishment rules are both feature deleting and apply to *FSs* that are identified by the features of themselves and of their context (*FSs*). Thus Occams razor (nowadays known as 'minimalist spirit') demands that there be no separate mechanism of impoverishment. Impoverishment is simply the effect of zero-*VIs* that consume features. For the impoverishment rule that deletes the dual feature in 1st person forms of Arabic jussives we thus assume the following *VI* at the beginning of the list:

(13)	*Phon*	*Context*	*Target*	*Deletes*
	((stem ϵ)	({}{})	{(1 +)(3 -)(pl +)(dl +)}	{(dl +)})}

While the reduction of impoverishment to vocabulary insertion is an innocent move in as far zero-*VIs* are deliberately assumed in the DM literature, the empirical question remains, if the *VIs*, which have to be stipulated under this analysis conform to the specificity hierarchy assumed for *VIs*[4]. If this holds true however, it is a further argument for our analysis since supposing that impoverishment obeys the same specificity requirements as *VIs* implies a more restrictive theory of impoverishment.

6 Theme Insertion is Vocabulary Insertion

The standard example for insertion rules (or conditions requiring) insertion are so called thematic vowels. A typical property of these vowels is that they are sensitive to idiosyncratic class membership of stems as in the following example from Ancient Greek(AG):

(14)				
	a.	*hoi log-O-i* 'the words'(mas)	b.	*hai nos-O-i* 'the illnesses'(fem)
	c.	*hai chor-A-i* 'the countries'(fem)	d.	*hoi polit-A-i* 'the citizens'(mas)

[4]cf. section 2, p.3.

Assuming that all instances of thematic vowels are triggered by class features, there is no reason to posit first the insertion of a 'theme position' which then has to be filled by *VIs*. Thematic vowels can simply be viewed as *VIs* consuming class features. Naturally the problem remains how to account for semi-regularity in the distribution of themes, e.g. (syntactic) masculine nouns in AG tend to take -*O*-. But even in standard DM analyses of such phenomena (Halle & Marantz, 1994) this is accomplished by different devices, namely redundancy rules, which won't be discussed her.

7 Fusion is obviated by vocabulary insertion

Fusion in some sense is simply the stipulation that some X^0s share a position, i.e. the best matching *VI* is inserted when it matches one of the fused X^0s, and no more than one *VI* can be inserted for the totality of the viewed items. It is assumed by Halle & Marantz (1993) e.g. to explain the single suffix position for English inflectional affixes. AGR and Tense are fused in a single node. Thus -*d* is taken as the default *VI* for past tense forms ((...{ +past })...)) and -*s* as the one for 3rd person ((...{ +3 +sg})...)). because of fusion forms like **prove-s-d* (3rd sg past) are excluded. Since -*d* comes earlier in the vocabulary list the form isn't **prove-s*.

There are a number of conceptual reasons to eliminate fusion from DM: First, it not only introduces a rule type not found elsewhere in grammatical theory, but also a type of representation (different items sharing one position) that is completely particular to one rule type. Note that fusion phenomena in phonology are 'real fusion' in the sense that features of two segments merge together in one *FS*, which in DM fusion isn't the case. (Alec Marantz, p.c.) Secondly, phenomena treated by fusion analyses simply look like impoverishment phenomena: *VIs* that are to be expected under certain contexts are not there. A third point is the target of fusion operations. All other rule types in DM affect only single items, and by this follow a strict version of locality, while fusion by definition manipulates more than one X^0.

Empirically most fusion analyses can be replaced by analyses without it. Thus assume for English a vocabulary list containing roughly the following:

(15)	*Phon*	*Context*	*Target*	*Deletes*
	((suff s)	({tense -past}{})	{+3 +sg}	{+3 +sg})}
	((suff d)	({}{})	{tense +past}	{tense +past})}

The blocking effect under this analysis will simply fall out from the different insertion conditions of the single *VIs*. Another case analyzed by Halle &

Marantz(1993) as fusion can be treated elegantly by impoverishment, which means of course a zero *VI*. In Georgian object prefixes 'block' subject prefixes, as you can see from the data in (16):

(16) a. *v-xatav* 'I see' b. *xatav-s* 'he sees'

c. *g-xatav-s* 'he sees thee' *g-xatav/*g-v-xatav/*v-g-xatav* 'I see thee'

While Halle & Marantz(1993) ascribe the blocking effect mainly to fusion it can equally well be captured by an $\emptyset$-allomorph for 1st person subjects in the context of 2nd person object morphemes. As we expect this *VI* will - following the specificity hierarchy - precede -*v* which is the default 3rd-person *VI*.

8 Parsing

8.1 Possible inputs as automata

Parsing will be understood here as the task of finding a set of input lists $\{ I_1 \ldots I_m\}$ for a (output) sequence of non-null *VIs* $O = V_1 \ldots V_n$, such that the generation algorithm described above given a vocabulary list VLwill generate for each I1-m an output of the form $\{VN\}^*V_1\{VN\}^*V_2 \ldots \{VN\}^*V_n\{VN\}^*$ where $\{VN\}$ is the set of zero-*VIs* in *VL*. Taking input lists as the relevant input which has to be reconstructed by the parsing procedure has the advantage that the lists can be interpreted as strings of *FSs*. Using only features with finite sets of values, fully specified *FSs* can be interpreted as the vocabulary of deterministic finite state automata (*DSA*s) or - equivalently - regular expressions (*RE*s, Kaplan & Kay,1994), which we then use to formulate restrictions on possible inputs. For the Arabic jussive forms the relevant input lists are enumerated by the *RE*:

(17) $\{ Stem_1, \ldots, Stem_n\} \quad \{ AGRS_1, \ldots, AGRS_n\}$

where $Stem_{1,\ldots,n}$ stands for all *FSs* characterizing stem and $AGRS_{1,\ldots,n}$ for all *FSs* resulting from subject agreement. I will call such automata denoting possible inputs *input automata*. Parsing now proceeds in two steps: a bottom-up and a top-down part

8.2 the bottom up part

The minimum that can be infered from a given (output) *VI* sequence $O = V_1 \ldots V_n$ is the following: In each corresponding input list for each $V_{i \in 1 \ldots n}$ there must be at least one *FS* that is subsumed by the target part of V_i. This is a simple consequence of the formalism developed here. Since there is no other operation in the system each *VI* is inserted in a FS_n that is itself the result of n applications of vocabulary insertion to an input *FS* (FS_i). By induction it can be shown that each such FS_n subsumes one FS_i. By the definition of vocabulary insertion a *VI* can be inserted in FS_n only if its target subsumes FS_n. By transitivity of the subsumption relation *VI* will subsume FS_i.

Technically this inference scheme is implemented by the way of *DSA*s. For each vocabulary item *VI* in a Vocabulary list there exists a finite set of possible input *FSs* subsumed by its target part, *Sub_Sume_Set(VI)*. When $\{X\}$ is the set of all possible *FSs* then the *RE* $\{X\}^*$ *Sub_Sume_Set(VI)* $\{X\}^*$ enumerates all input lists that contain at least one *FS* subsumed by *VI*. We call this *RE At_least_1(VI)*. The minimum parse of a (output) *VI* sequence $V_1 \ldots V_n$ is then the intersection of all *RE*s *At_Least_1*(VI_i), which is again a *RE*, since *RE*s are closed under intersection (Kaplan & Kay, 1994). The resulting *RE* will again be intersected with the input automaton, giving a candidate automaton enumerating possible inputs for O.

8.3 The Top-Down-Part

The basic idea here is to traverse the candidate automaton and to check for each path, if it is consistent with the morphemes in the output *O(List)*, starting at the stem of O and moving outwards as parsing proceeds. The two basic procedures of the algorithm are given in pseudo-code:

(18) **SPELL_PARSE(***State,List,Pref_P,Suff_P,Parse,L_Context***)**

if (*Pref_P* is leftmost in *List*) **and** (*Suff_P* is rightmost in *List*)

  if *State* is final
    accept(*Parse*) % Parse Success %
  else if there are no more transitions from *State*
    reject(*Parse*) %Parse failed %
  else
    for all transitions from *State* over some *FS F* to some state *S*
      ITEM_SPELL_PARSE(*F,S,List,Pref_P,Suff_P,Parse,L_Context*)

We start at the stem of O and the state S we reach from the initial state I of the candidate automaton traversing the stem transition. Then all *FSs* are considered that have a transition from S to another state (the 3rd part of SPELL_PARSE). When none of the termination conditions in SPELL_PARSE is fulfilled (parts 1 and 2 of (18)), each such *FS* is tentatively spelled out by ITEM_SPELL_PARSE. When this test spell-out is compatible with the next affixes the *FS* is added to the actual parse list (*Parse*)and SPELL_PARSE is recursively used to parse the rest of the output (*List*).

(19) **ITEM_SPELL_PARSE(***FS,State,List,Pref_P,Suff_P,Parse,LC***)**

```
set VI_Pointer to the first element of VI_LISTE
Copy FS to New_FS
while
(set First to ROUGH_MATCH( VI_Pointer,LC,FS) ≠ END)

    if First has a right context specification
        RIGHT_PARSE( ... )
        return

    else if phon(First) ≠ NULL
        if TEST_NEXT_AFFIX(First,Pref_P,Suff_P,List) = false
            return

        else
            delete all features in FS_New that are
            in Deletes(First)

Insert FS in Parse
set LC to FS
set New_S to Next_State(FS, State)
SPELL_PARSE( New_S,List,Pref_P,Suff_P,Parse,LC)
```

TEST_NEXT_AFFIX checks the compatibility of the *FS* tested in ITEM_-SPELL_PARSE with the affixes at the actual pointer positions. As soon as the test spell-out would result in adding a non-zero *VI*, it will checked whether the *VI* is present next to the stem, on its left in case of a prefix on its right side otherwise. If the *VI* isn't found there this branch of the parse process is discarded. There are two pointers in each function marking the position of the next prefix and suffix (if any) not yet checked for consistency with the parse branch. For each *VI* that is found at the correct place in the output TEST_NEXT_AFFIX moves the corresponding pointer one position further on the *VIs* 'outside', i.e. the suffix pointer on the right of a suffix, the prefix pointer on the left of a prefix.

Note that in ITEM_SPELL_PARSE to test insertion conditions for *VIs* instead of FIRST_MATCH(9) the "weaker" ROUGH_MATCH is used which doesn't check the *VI* against the actual right context. This is computationally cheaper and innocuous as long as no right context specification of an *VI* emerges during test-spell-out. If, however, this case arises ITEM_SPELL_OUT invokes the function RIGHT_PARSE which tests the right context of *VIs* against all right possible right contexts resulting from the syntax automaton using FIRST_MATCH.

(20) **RIGHT_PARSE(***State,List,Pref_P,Suff_P,Parse,L_C***)**

for all transitions from *State* over some *FS RC* to some state *S*

ITEM_RIGHT_PARSE(*F,S,List,Pref_P,Suff_P,Parse,L_C, R_C*)

ITEM_RIGHT_PARSE corresponds to ITEM_SPELL_PARSE, but takes into account right contexts. Since the next *FS* to be spelled out has to be the actual right context, instead of SPELL_PARSE ITEM_SPELL_PARSE is invoked.

(21) **ITEM_RIGHT_PARSE(***FS,State,List,Pref_P,Suff_P,Parse,LC, RC***)**

set *VI_Pointer* to the first element of VI_LISTE
Copy *FS* to *New_FS*
while
(**set** *First* **to** FIRST_MATCH(*VI_Pointer,LC,FS*) $\neq$ END)
else if phon(*First*) $\neq$ NULL

if TEST_NEXT_AFFIX(*First,Pref_P,Suff_P,List*) = false
return

else
delete all features in *FS_New* that are
in Deletes(*First*)

Insert *FS* in *Parse*
set *LC* **to** *FS*
ITEM_SPELL_PARSE(*RC, State,List,Pref_P,Suff_P,Parse,LC*)

9 Some theoretical Consequences

- There is an upper bound on the length of derivations. As can be seen from (7), each spell-out derivation for a input list $I = Fs_1, \ldots, FS_n$ is a sequence of n applications of ITEM_SPELL_OUT. Within each such application maximally m *VIs* are inserted by (10), where m is the number of morpho-syntactic features in the language. This is a consequence of the feature-deleting nature of VI_INSERT and the stipulation that the Deletes-part of *VIs* be non-empty(3.2): Each application deletes at least one feature, thus after m insertions all features must be deleted. Taken together the derivation of I involves maximally $m \cdot n$ insertion steps.

- Redundancy in morphology is highly restricted: Natural language morphology is notorious for 'multiple exponence', i.e. more than one *VI* realizes identical features of the same lexical item[5]. From the upper bound on vocabulary insertion steps it follows that each *FS* with m features can be realized by at most m *VIs*. Further the insertion of two *VIs* VI_1, VI_2 with identical target specifications is impossible[6]: Suppose VI_1 is inserted in some *FS*. Since the deletion part of $VI1$ is a subset of its target features at least one of these features is deleted and can't be subsumed by the target features of VI_2 anymore. This blocks further insertion of VI_2.

- All discussed rule types obey the same narrow restrictions. In standard DM different rule types are subject to different restrictions: Vocabulary insertion follows the specificity hierarchy, impoverishment doesn't. Fusion applies to complexes of *FSs*, most rule types apply only to single *FSs*. In the given formalization every operation follows the hierarchy and applies to single *FSs*, since there is only one operation type, which is defined accordingly.

[5]'realize' is meant here technically as 'occuring in the target part of an VI'.
[6]This is true regardless of the context specifications.

10 Limitations and Extensions

10.1 Context Specifications

The main limitation of the developed formalization is the way in which contexts are determined. First, only lexical items are assumed to serve as contexts, but not *VIs*. Halle & Marantz(1993:119) assume something like the latter in their analysis of Georgian. However the analysis can be easily done with reference to the features that trigger *VI* insertion instead. Note that assuming cyclic insertion of *VIs* the actual *VI* can be relevant in our terms only as left context. Left contexts in our formalization however are essentially stipulated. (see (7)). if it turns out that *VIs* are the relevant (left) context as argued in Bobaljik(1999) this can be accommodated in the formalism without problems.

Secondly, in our formalization only the two *FSs* that are structurally 'nearest' to a *FS* are considered as possible contexts. Halle & Marantz(1993, 1994) however claim that any *FS* that stands in a government relation with an *FS* can serve as relevant context. For example in the following form from Potawatomi(Halle & Marantz,1993: 155) the appearance of the plural marker *-uk* is bleeded in their analysis by an impoverishment rule taking the feature specification of the 1st plu marker *-mn-* as context.

(22) *n-wapm-a-mn-(w)apun/*n-wapm-a-mn-(w)apunin-uk*
'we saw them.'

Since the tense marker *(w)apun(in)* intervenes between the two items the *FS* triggering *-mn-* can't be structurally adjacent to the impoverished *FS*. hence the analysis is impossible in the given formalization. It doesn't seem to be difficult to adjust the model in a way that allows a more extended set of *FS* as left contexts[7] by introducing a stack that 'memorizes' spelled-out *FSs* in either generation and parsing. However it would certainly increase the complexity of parsing to do the same for right contexts.

At the current understanding however it seems problematic to determine the correct locality domains for contexts. Government itself is a concept that has been largely abandoned in syntactic research (Chomsky, 1995). Even the interpretation of (22) is dubious. Note that *-mn-* and *-uk* spell out features of the same argument. Assuming that they also realize the same agreement node (*FS*) would entail that spell-out cannot be cyclic, since the tense *FS* would intervene. Halle & Marantz(1993:145) avoid this conclusion

[7] as would be necessary for analyzing the Potawatomi data

by stipulating two AGR heads that agree with the same argument, but without further evidence it is unclear if (22) is an argument against cyclicity or against a restricted form of locality like in our formalism.

10.2 The Ineffable and Default *VIs*

In most work on DM morphology is assumed to be interpretive in the sense that each output from syntax will get some spell-out at MS. Put differently: there are no purely morphological cases of ungrammaticality. Related to this is the speculation that "Universal Grammar provides a zero spell-out as the default phonological realization of a morpheme in the unmarked case"(Halle & Marantz(1993:134-35). Such a *VI* containing no feature specifications is technically impossible according to the definition in 3.2[8]. A default zero can be mimicked by a set of zeros targeting and deleting exactly one feature for every morpho-syntactic feature. However if morphology is interpretive, there is reason to suppose any such zero only if (in terms of standard DM) each X^0 must be filled with a *VI* or (in terms of our formalism) if all morpho-syntactic features must be deleted. Conceptually this latter fits well with the idea that features irrelevant for an interface (in this case PF) have to be removed for a structure to be interpretable (Chomsky, 1995), but in an interpretive approach to morphology I see no empirical reasons why complete deletion of morpho-syntactic features should be necessary. If MS operations *aren't* interpretive, as suggested in Marantz(1999), i.e if there are "ineffable" forms like the past participle form of "stride"[9], the lack of suitable *VIs* could be the reason for morphologically ungrammatical forms under the assumption that features have to be deleted. Obviously a wellformedness condition requiring just that can be integrated straightforwardly in the current formalization but it has to be seen if this approach is the right one to explain ineffability.

10.3 Late insertion of stems

Marantz(1995) argues that the lexical items treated by syntax don't contain idiosyncratic semantic information like the one that would distinguish 'cat' and 'dog'. Features differentiating in such cases are introduced by vocabulary insertion. There is no problem in our formalism to introduce *semantic* features (phonological features are introduced as well), but it is technically impossible to introduce morpho-syntactic features, since vocabulary insertion by definition(see 3.3) deletes such features. There are two possible

[8] Intuitively: It wouldn't be feature-consuming.

[9] ?I had strode, ?I had stridden, see Marantz(1999:5) for further examples

moves to avoid the problem. First, we can claim that context specifications of *VIs* can refer to the semantic properties of adjacent *VIs*. E.g. the plural *VI -en* would be restricted to the context of the *VI* with the semantics of *ox*. Secondly, it can be assumed that vocabulary insertion of stems is of a different nature than that of functional morphemes. This move is independently necessary, when stems aren't identified by lexical items, since no competition is possible between stems that can be inserted in identical syntactic environments (see Harley & Noyer, 1998). Thus anyway a formal analysis of stem insertion will have to be worked out.

10.4 Feature deletion and readjustment rules

Readjustment rules are claimed in DM to cause morpho-phonological alternations. Such a rule changes e.g. the vowel quality in the stem (-*VI*) of 'steal' in the context of certain tense morphemes('stolen', Halle & Marantz(1993:127-29)). This seems to require at least some modification of our treatment of vocabulary insertion, since *VIs* aren't separate entities after insertion. A further problem is the derivation point where readjustment rules are applied: When vocabulary insertion deletes (possibly all) features of *FSs* and readjustment rules are sensitive to the content of *FSs*, readjustment rules can't apply after all *VIs* are inserted. A natural solution to both problems would be to intersperse vocabulary insertion and the application of readjustment rules. Whenever a *VI* is chosen for insertion all readjustment rules that can apply to it in the actual derivation context are carried out, and the modified *VI* is inserted afterwards.[10] Like all modifications considered in this section this has still to be worked out technically in its consequences for generation and parsing.

References

BOBALJIK, Jonathan D. (1999) The Difference between *-nin* and *-nen*: Constraints on Contextual Allomorphy. Talk presented at GLOW'99, Berlin.

[10]If one wants to maintain that all readjustment follows all vocabulary insertion, there are further possibilities: Instead of deleting features, these could be marked in some way as "discharged"(cf. Noyer, 1997) and be thus available for later readjustment. It could turn out that readjustment rules are sensitive not to lexical items(*FSs*) but to the *VIs* inserted. The formalism than would have to keep track of the (morpho-syntactic feature content of the inserted *VIs*.

CHOMSKY, Noam (1995) The Minimalist Program, MIT Press.

HALLE, Morris & Alec MARANTZ (1993) Distributed Morphology and the Pieces of Inflection. In: The View from Building 20, ed. Kenneth Hale and S.Jay Keyser. MIT Press, Cambridge, 111-176.

HALLE, Morris & Alec MARANTZ (1994) Some key Features of Distributed Morphology. In: MITWPL 21: Papers on phonology and morphology. ed. Andrew Carnie and Heidi Harley. MITWPL, Cambridge, 275-288.

HALLE, Morris (1997) Distributed morphology: Impoverishment and fission. In: MITWPL 30: Papers at the Interface, ed. Benjamin Bruening, Yoonjung Kang and Martha McGinnis. MITWPL, Cambridge, 425-449.

HARLEY, Heidi & Rolf NOYER (1998) Licensing in the non-lexicalist lexicon: nominalizations, vocabulary items and the Encyclopedia. In: MITWPL 32: Papers from the Roundtable on Argument Structure and Aspect, ed. Heidi Harley, Cambridge, 119-137.

KAPLAN, Ronald M. & Martin KAY (1994) Regular Models of phonological Rule Systems. Computational Linguistics, 20(3):331-378.

MARANTZ, Alec (1995) A late note on late insertion. In: Young-Sun, Kim et al.(ed.) Explorations in Generative Grammar, Seoul, 396-413.

MARANTZ, Alec (1999) Morphology as syntax: Paradigms and the Ineffable (The Incomprehensible and the Unconstructable). Talk given at the University of Potsdam.

NOYER, Rolf (1997) Features, Positions and Affixes in Autonomous Morphological Structure. Garland Publishing, new York. revised version of 1992 MIT Doctoral Dissertation.