A COMPUTATIONAL ARCHITECTURE FOR LEXICAL INSERTION OF COMPLEX NONCE WORDS

Bruce Mayo
Universität Konstanz*

Proceedings of the LFG06 Conference
Universität Konstanz
Miriam Butt and Tracy Holloway King (Editors)
2006
CSLI Publications
http://csli-publications.stanford.edu/

Abstract

Derivational morphology has been conspicuously neglected in the LFG literature and elsewhere. Existing proposals in HPSG treat derivational patterns that are constrained and regular but sidestep problems raised by derivations that require knowledge-based, pragmatic evaluation, and most ignore the problem of nonce derivations, which must be processed on-line, i.e., concurrently with syntax. While practical implementation remains a formidable challenge, it is possible to specify a computational architecture that accounts for some 'worst case' patterns of productive nonce derivation in Italian that require pragmatic evaluations. This architecture factors lexical insertion into two functions, c- and m-insertion, for inflection and derivation. A buffer between these functions and the syntax component is shown to explain lexicalization phenomena, and it is argued that it may be one of the cognitive sources of Lexical Integrity.

1 Introduction

For computational and theoretical linguists, morphology has by and large meant inflectional morphology, and derivational morphology has been seen as a set of static, irregular and unpredictable relations within the lexicon that do not merit synchronic analysis. An important exception is work that has been done in the HPSG framework, where a number of researchers have shown that some, usually rather specific, derivational patterns can be modelled using that formalism's inheritance relations (Koenig and Jurafsky 2004, Koenig and Davis 2006). In LFG, the macro facility available in the extended formalism of the XLE system could, in principle, be used to derive related sets of words. In both cases, however, what results is little more than a kind of data compression: suitable roots are expanded to their derivational variants at compile time, sparing the lexicon writer the effort of creating explicit entries for them. More recent work in HPSG has explored the use of semi-productive lexical rules for inserting complex, transparently derivable words into syntax at run-time (Briscoe and Copestake 1999). As will be shown in this paper, however, the underlying nature of derivation inevitably makes it difficult to predict the semantics, mapping relations and other properties of all transparently derivable words without recourse to a level of general knowledge representation and conceptual operations. The task is not hopeless, however. A growing body of work, e.g., (Corbin 1990, Mayo et al. 1995, Stiebels 1996, Lieber 2004) demonstrates that the conceptual operations underlying derivation as well as the relationships between semantic representation and syntactic expression (Levin and Rappaport Hovav 2005) can be grasped. A computational implementation, however incomplete, can help to clarify what kinds of information must be available at each of the several interfaces that make up the linguistic system.

1.1 The KLU Computational Model

For many years, theoretical linguists were inclined to regard the derivational relationships among words as irregular and unpredictable, the product of historical processes lying outside the purview of grammar. However, experimental studies of the representation of lexical items, e. g., (Marslen-Wilson et al. 1994), reveal that some derivational roots and affixes seem to have independent mental representations, much like stems and inflectional affixes, suggesting that they ought to be freely combinable; and from corpus statistical studies (Baayen and Renoulf 1996) we know that certain derivational patterns are continually producing new words that no dictionary could hope to anticipate. Where words are produced freely, like sentences, it must be possible to identify and model the grammatical processes that create them. To this end, a small linguistic workbench for derivational morphology, called KLU, was developed at the University of Konstanz and was used to implement small grammars for word and sentence comprehension (Mayo 2000). Specifically, it was meant to provide formal solutions to the following problems:

- Complex and apparently idiosyncratic words can appear ‘out of the blue’. Hence, derivation must be possible concurrently with sentence analysis, i.e., within syntax.
- Derivation is not syntax, but its structures look much more like syntax than those of inflection: derived words do not fill out paradigms; like sentences, they exhibit structural embedding and name a limitless range of objects and events.
- Unlike inflectional attributes, the meanings that arise from derivation are subject to lexical shift, so that derived words often have competing transparent and opaque meanings.

To accommodate these requirements, the KLU program introduced three formal devices:

- **c-insertion**, which performs inflectional analysis and has tacitly always been a part of LFG
- **m-insertion**, which does derivational analysis, supplying newly derived stems to c-insertion, and
- a **morphological buffer**, which serves as an interface between the syntactic and the morphological components. The buffer is necessary for processing efficiency, and it helps to model the process of lexicalization.

The program was required to construct the semantics of sentences containing nonce derivations, mainly Italian. Its favourite derivation was *disiscrivere*, an invented Italian word meaning to ‘unregister’, as in

(1) La fata disiscrive il cavaliere dal castello.

‘The fairy unregisters the knight at the castle’

To allow the parser to process a sentence containing the non-lexical item, the program could create an ‘on-the-fly’ lexical entry for the unknown word, containing a lexical form and a semantic formula that, in the program’s output, looked like this:

```
/ dis iscriv e / ================================
§ Dis_iscrivere(S,O,L) =>accomplishment(v1)
                              agent(v1,S)
                              theme(v1,O)
                              localization(v1,L)
                              phase(v1,CAUSE(S,
                                    CHANGE(LISTED(O,L),NOT(LISTED(O,L)))))

TpLex:Verb -> /TpLex:Verb/   [PRED: dis_iscriv((^ SUBJ),(^ OBJ),{(^ OBL)}]
                              [AUX: AVERE]
                              [μ^ INFLCLASS: Vkere]
                              (^ OBL PCASE ) =/cc Da
```

Using these, the parser and the semantic analyser constructed a semantic analysis of the embedding sentence.

```
The relation dis_iscriv(Fata,Cavaliere,Castello) =>
     ACCOMPLISHMENT(411)
     AGENS(411,Fata)
     THEMA(411,Cavaliere)
     LOCUS(411,Castello)
     PHASE(411,CAUSE(Fata,
                 CHANGE(LISTED(Cavaliere,Castello),
                         NOT(LISTED(Cavaliere,Castello)))))
```

1.2 Computational Overview

While the KLU program was little more than a makeshift toy, it had to deal with a wide range of phenomena involved in derivational morphology, from lexical phonology to discourse structure, and it gave considerable insight into the kinds of problems that nonce words present for a lexically-oriented theory like LFG. To get a quick overview of how the functions listed above might deal with a sentence containing a new word, consider (2).

(2) Please open the wine bottles with those unflaskers.

Now *unflaskers* is not to be found even in the OED, but speakers have little difficulty understanding what is meant. Hence, some sort of analysis below the word level must be making the word available to syntax. Figure 1 is meant to give an impression of what happens; the details will be fleshed out in sections 2 to 4.

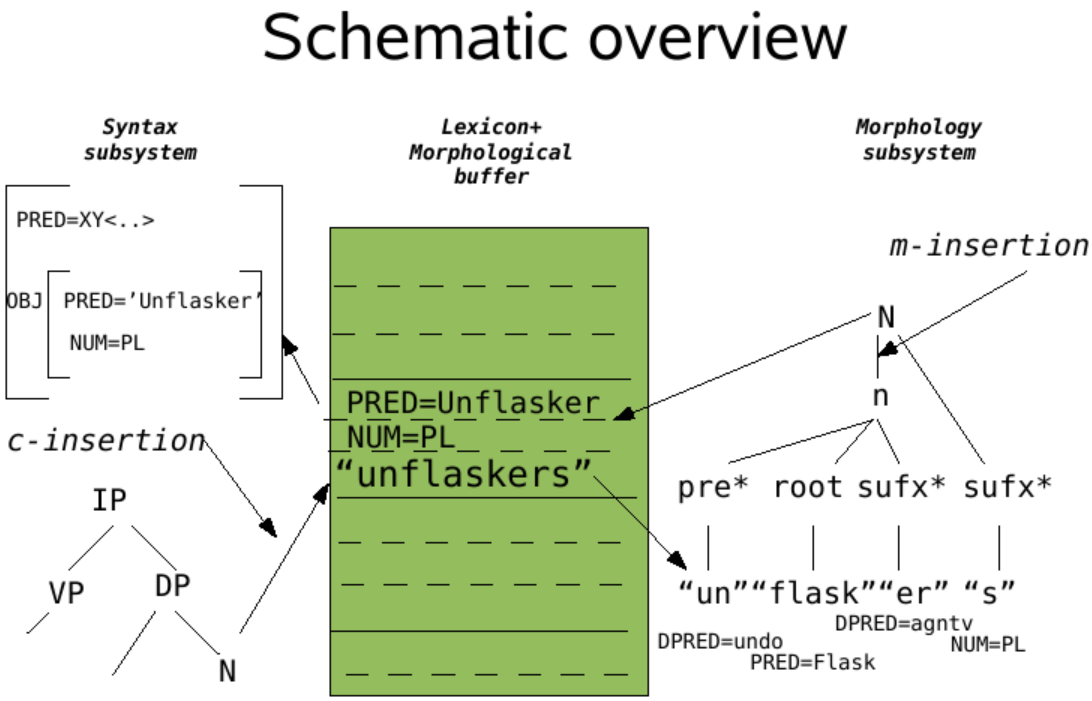

Figure 1. Schematic Overview of the KLU Model.

Consider a top-down parser working its way through the sentence, word by word. When it reaches a leaf node of the c-structure, in the proposed model it does not access the lexicon directly but calls the function **c-insertion** to obtain the lexical features of the item. For each word encountered, c-insertion places the surface form in a lexical buffer (shaded box), which queries the static lexicon. For words like *please*, *wine*, etc. it returns stored features without further ado. But when c-insertion encounters *unflaskers* there is no information in the lexicon to return. Thus, the buffer invokes morphology to decompose *unflaskers* into sublexical units (not really morphemes), the substrings "un" "flask", "er" and "s". Since these are not c-structure entities, their features cannot project to any c-structure nodes, and they do not even unify. Hence, they require a derivation. **c-insertion** calls a rather complex operation, called **m-insertion**, to answer a semantic riddle of the form "what object undoes something involving a flask". Presumably it finds a conceptual description of some sort of tool that is meant to open or empty flasks, or a relevant generic concept. Abstracting a semantic structure from this concept, Lexical Mapping Theory creates a corresponding lexical form and deposits a

PRED feature in the buffer. m-insertion also places a lexical-semantic formula in the buffer (not shown) as the referent of the PRED feature, so that the parser will be able to construct the semantics and discourse structure of the sentence. The morphological analyser then recognizes the segment "s" and deposits the feature Plural in the morphological buffer. This results in a morphologically complete lexical item. Now syntactic analysis can continue as if the form *unflaskers* had existed in the lexicon all along.

The following paragraphs describe these three components in more detail. For **c-insertion** (section 2) traditional LFG approaches to morphology are fine for inflection but not for derivation. For **m-insertion** (section 3), I shall fill in the outline just presented. Section 4 then sketches the entire process, showing how the Italian verb *sbobinare* 'unspool' would be inserted as a nonce derived word. For the **morphological buffer**, section 5 shows that, like c-insertion, it is an idea that has been around for a long time, but whose theoretical significance has never been comprehended. Computationally it is unavoidable, and it lets us model the effects of lexicalization and semantic drift of derived words. Its role in mediating between syntax and lexicon may be one of the reasons why we find a structural barrier between syntax and lexicon.

1.3 Lexical Integrity

It is evident that the model described in Figure 1 allows word analysis from within syntax but does not violate LFG's basic postulate of lexicalism. c-insertion mediates between syntax and morphology, constituting a kind of barrier between the two. The boundary between syntax and the lexicon is a rather complicated matter (Bresnan and Mchombo 1995); but in Bresnan's 2001 formulation it is simply the point where the structural principles of c-structure end and 'morphological completeness' begins:

> Morphologically complete words are leaves of the c-structure tree and each leaf corresponds to one and only one c-structure node (Bresnan 2001, 92).

Hence, one might see c-insertion as being the computational expression of this definition. But if it turned out to be necessary on other, independent grounds, we might want to see it as being a cognitive reason, or causal source, of Lexical Integrity. This is in fact a claim I want to make, but the justification will only unfold when we consider the last of the three proposed components, the lexical buffer, in section 5.

2 The c-insertion Function

Figure 2 describes c-insertion in more detail. It constitutes the interface from c-structure to the lexicon. At each c-structure leaf node, the syntactic parser passes the input string to c-insertion, which returns the string's lexical features if it can find or compute them. Seen from syntax, this is simply an access to the abstract lexicon. Whether the features are obtained from the static lexicon (upper shaded box) or from an analysis via the morphological buffer does not matter, and syntax cannot tell the difference. Formally, c-insertion can be described using an apparatus similar to that used in syntax (Börjars et al.1996) if the restrictions on word structure described in (Bresnan and Mchombo 1995) are born in mind.

c-insertion corresponds in the XLE system to the two-level morphology component, but in KLU its algorithms draw on ideas from lexical phonology. A significant feature is that segmentation is kept fully separate from morphological analysis proper, which deals only with the output from segmentation (details are discussed below and in more depth in Mayo 2000).

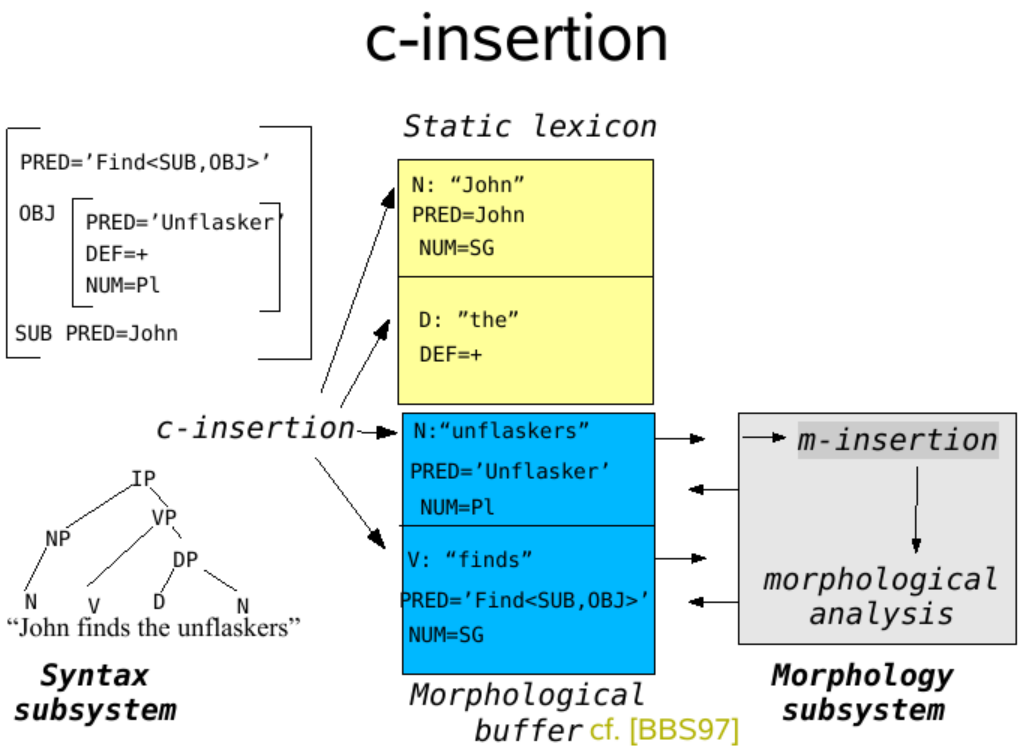

Figure 2. The c-insertion Function.

Monomorphemic words are fetched directly from the static lexicon; stems and inflectional segments unify locally within c-insertion and remain for a time as atomic morphemes in the buffer. If c-insertion cannot find a lexical stem for a string, it sends it to the derivational analyser (box on the right), which will try to return lexical features to the buffer.

In Figure 2, we see that the items "John" and "the" of sentence (2) are present with their lexical features in the upper shaded box, representing the static lexicon. In contrast, "finds" and "unflaskers" only appear in the buffer (lower box) after their constituents have undergone inflectional and derivational analysis. The unification takes place in local structures, and there is a test for morphological well-formedness (e.g, for the presence of inflection on categories that require it, and for the absence of features belonging to derivational morphemes). Once an analysed word is in the morphological buffer, it does not need to be recomputed, and it remains there for a while, indistinguishable from entries in the static lexicon, until it is eventually purged to make room for new computations. If a purged item is analysed again, it is purged more slowly the next time around. This suggests that frequently encountered inflected forms might stay in the buffer indefinitely and behave just as if they were monomorphemes in the static lexicon, in effect, as if they had no internal morphological structure. This is in fact what some experimental studies have found to be the case, e.g., (Baayen et al. 1997).

3 The m-insertion Function

The function m-insertion is actually only a sub-function of c-insertion, called when c-insertion encounters a derivational segment, whose features cannot be projected to syntax. Since derivation takes place inside inflection, many have been tempted to think of it as a sort of extension or elaboration of inflection, especially since inflection has proved to be computationally quite tractable. There are many kinds of derivational relations that seem to be regular and systematic, like passivization, causative formation, etc., especially in morphologically rich languages. Hence, one could think of derivation as filling out paradigmatic matrices, albeit

large ones, and therefore as being amenable to strategies that are effective for complex inflectional paradigms (cf. Karttunen 2003). If we want to be prepared for the worst cases, however, this would be much too simplistic. A representative 'worst case' is the denominal verb of removal in Italian. It creates not only a new semantic structure, more complex than that of its base, but also introduces a new argument structure and other morphological features like inflectional class. This is a type of derivation that is very productive and has been studied in detail, e.g., in (Mayo et al. 1995, von Heusinger and Schwarze 2006). An example is *sbonbinare*, from *bobina* 'spool'. It can be used transparently to mean 'pull wire from a spool' as in this example:

(3) *Un missile sbobina un filo* 'a missile unspools a wire' (Massari 2006)

Likewise, from *crema* 'cream' we can derive *scremare* 'to skim'; from *forno* 'oven' we get *sfornare* 'to take out of the oven'; *carta* 'paper' gives *scartare* 'unwrap' or 'remove from wrapping'. We can more or less get the compositional semantics of *sbobinare* using the word grammar shown below,

```
V →   DPrefx      Root
        ↑=↓      ( ↑Arg) = ↓

Root →  N
          ↑=↓

s-,   DPrefx     (↑μ DPRED) = 'RemoveXfromY<-o,-r, (↑Arg)>
                 (↑μ CAT)= V
                 (↑μ CLASS)= -are
```

which unifies the constituents *s-* and *bobin(a)* to yield the following features at (local) f- and m-structure:

```
(↑μ DPRED) = 'RemoveXfromY<-o,-r, (↑Arg)>'

(ARG PRED)='Spool'
 (↑μ CAT)= V
 (↑μ CLASS)= -are
```

The base of the derivation, *bobin(a)*, has a PRED feature, which will be projected to a local f-structure so as to furnish the argument Arg, subcategorised by RemoveXfromY. The derivational morpheme, *s-*, has a special DPRED feature that is allowed to appear only in morphology. Unlike PREDs, DPREDs seem never take more than one argument, at least in West European languages, and there are reasons to think that this argument is too primitive to be considered a true grammatical function (e. g., its position in word structure is fixed; it is not assigned case; it has no anaphoric links). As suggested in section 1, Lexical Integrity implies a barrier between morphological constituents and sentence-global f- and m-structure, so that word-internal constituents do not project features directly to syntax but only as mediated by the morphological component. Morphology is non-monotonic insofar as it is permitted to perform operations like substituting the evaluation of a function — in this case, RemoveXfromY — for the function, while deleting the function's arguments. Hence, the DPRED can be deleted as well as the PRED feature of the base noun, 'Spool', and both will not appear in sentence-level f-structure. Instead, the interface projects a *new* PRED that is not a constituent of the word but is created by evaluation of the DPRED at run-time, and it creates a lexical semantic formula to which this new PRED refers.

Once the word grammar (via m-insertion) has eliminated the DPRED and its argument from *sbobin-* from the local f-structure, the only morphemic segment remaining is the verbal inflection, *-a.* It adds the features NUM, PERS, and TENSE monotonically to the newly derived stem, as shown below.

NUM=SG
PERS=3
TENSE=Pres
PRED=??? Lexical Semantics ???

Hence, inflection is relatively easy to handle. The task of m-insertion, on the other hand, is to obtain an f-structure PRED by substituting arguments to the DPRED's derivational function (RemoveXfromY<-o,-r,(↑Arg)>), so as to obtain a semantics, an argument structure, a lexical form, and other required attributes. A daunting job that, at present, has not been solved for the general case but can be solved for some specific but highly complex derivational patterns, like that of *sbobina*, as we shall see in the next section.

4 c-insertion in Detail: *sbobinare*

Figure 3 sketches how the derivation of a denominal verb like *sbobinare* takes place in the KLU program. At the time KLU was written, much less had been said about Lexical Mapping Theory and about conceptual unification than is now the case, but a genuine implementation of these functions would still be a large piece of work. In KLU they were implemented as very sketchy dummies. Nevertheless, my impression is that most of the pieces exist; someone with ample resources just needs to put them together.

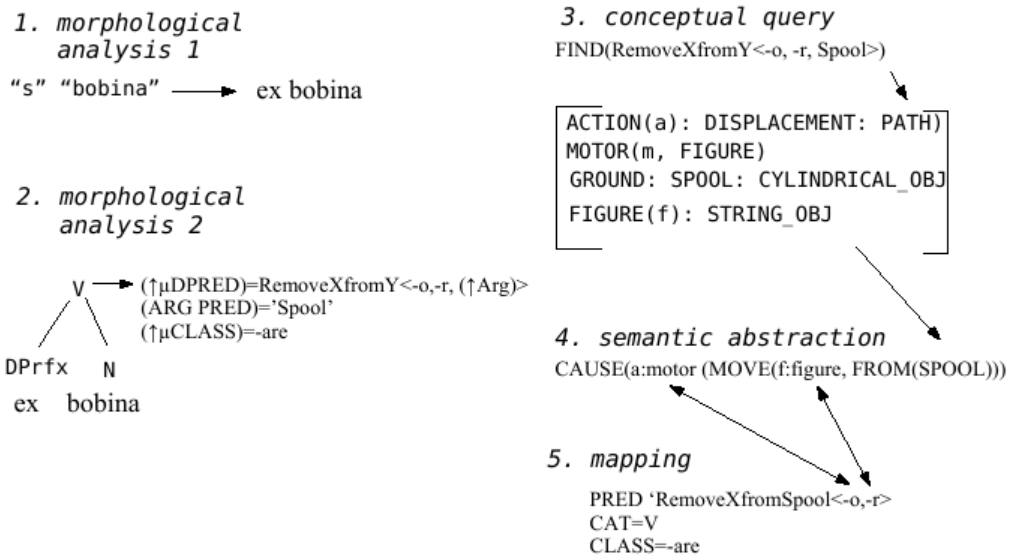

Figure 3. c-insertion Step-by-Step.

Apart from its job of filling the lexical buffer from the static lexicon and purging it, when it is called with an internally segmentable item, c-insertion can carry out the five steps **Morphological analysis 1 and 2, conceptual query, semantic abstraction** and **mapping**. For inflected stems, only Morphological analysis 1 and 2 are needed to add inflectional features to the stem. If a new stem needs to be derived, all steps must be carried out. At the end, m-insertion returns all required features, including a new lexical form and a semantic formula, to c-insertion, which copies the features relevant to syntax into the morphological buffer and finally returns them to the leaf node of c-structure that invoked c-insertion. Let me explain the steps one by one.

4.1 Segmentation (Morphological analyser 1)

To insert a nonce word to syntax, it is first necessary to obtain lexical features from each of the morphological constituents. This task, as has been mentioned, can be carried out by a conventional two-level analyser. KLU, however, was meant to explore ideas, not directly related to problems of computational morphology, about the overall structure of the mental lexicon. To this end, several two-level analysers were constructed in such a way as to mimic roughly the "domains" of lexical phonology, with separate analysers for the root and for derivational, inflectional, and clitic domains. (The nomenclature is a bit misleading because, during segmentation, nothing is known about the morphemic structure, but a rough correspondence exists, e. g., in the outer domain of the phonology we find mainly inflectional morpheme segments.This structure is probably reflected in the "continuation lexicons" of most two-level analysers). Segmentation thus enforces a number of lexical-phonological constraints on word structure, more or less prohibiting inflection inside derivation and the like, but it does not fetch or unify lexical features of the segments. This is the task of a separate word-grammar component, to which segmentation merely furnishes the input. Thus, segmentation and word grammar share the responsibility for enforcing the constraints on word structure that make it markedly different from sentence structure.

KLU's segmentation component uses hand-coded orthographic transducers to lemmatize words or parts of words ("surface strings") to strings found in the lexicon ("lexical strings"). Surface strings that are already present as lexical strings have precedence over strings that must be derived, long strings have precedence over short strings, and short derivations beat long derivations, roughly implementing the Panini or 'elsewhere' principle of lexical phonology (Kiparsky, 1982). In part 1 of Figure 3, "s" (an allomorph of "dis") is reduced to a lexical string, *ex*, while the surface string "bobina" remains the lexical string *bobina*. If a surface string cannot be lemmatized to any lexical string (neither as a full-form word nor as a known morpheme), segmentation fails. However, not knowing anything about true morphological structure, segmentation cannot distinguish between genuine derivational morphemes like *re-* in *retake* and pseudo-morphemes like the *re-* in words like *rejoice*. Since segmentation cannot distinguish pseudo from real morphemic segments, it is arranged that the segment list *re.joice* matches the lexical entry /re°joice/ immediately and blocks morphological decomposition.

A segment that can be found in the lexicon is not segmented further, or its segmentation is postponed. A motivation for this approach was the consideration that in speech or optical character recognition, the inputs can be many-ways ambiguous, and, computationally, morphological analysis is likely to be far more expensive than it is for computer-encoded input. It was supposed that limiting analysis to the word's periphery and giving the lexicon precedence over analysis might be nature's way of coping with this problem.

Unlike the well-known two-level transducer techniques, KLU works from both ends of the word toward the middle until it identifies a single root segment. Because affixes are separated without knowledge of the underlying morphology, segmentation can obtain misleading embeddings, as in *unhappier*. The suffix *-er* cannot be removed from *unhappy* because *unhappy* has three syllables, and *-er* attaches only to words of one or two syllables. This forces segmentation to first remove *un-* and then *-er* (both belonging to the derivational domain). This would give the segmental bracketing [un-[[happi]-er]], which would mean NOT(MORE(HAPPY)) instead of MORE(NOT(HAPPY)). Therefore segmentation returns only flattened, non-embedded lists of lexical strings to the word grammar (morphological analysis).

Compounds (as in German) are not accepted, as it was felt that they present a separate problem.

4.2 Word Grammar (Morphological analyser 2)

The input to the second stage of morphological analysis is the flattened (non-bracketed) list produced by segmentation, shown in part 2 of Figure 3. "ex bobina" looks like a tiny syntactic phrase, and the morphological parser looks like a miniature version of sentence analysis. It projects features of affixes and roots from

the lexicon to local f- and m-structures and unifies them. But there are important constraints. The phrase-structure rules must describe regular grammars and in general obey the principles outlined in (Bresnan and Mchombo 1995), although the compiler does not enforce most of these rules. But the range of computable forms is still immense.

As mentioned earlier, the resulting feature set is purely local, i.e., it does not unify immediately with sentence level f- or m-structure. If the resultant f-structure is well-formed (e.g., does not contain a DPRED), the results are deposited in the buffer and returned to the leaf node of c-structure that called c-insertion. If not, the following steps, 3 to 5, are taken to produce a new, derived stem. This stem can then be unified with any inflectional affixes, as if it had been drawn from the lexicon directly.

4.3 Conceptual query

After evaluation of the morphological structure, the DPRED's lexical form 'RemoveXfromY<-o,-r, (↑Arg)>' for *sbobinare* is presented as a query to a knowledge data base, illustrated by the function FIND in part 3 of Figure 3. Note that the argument list of RemoveXfromY does not assign thematic roles. The [-o] argument will probably be an agentive subject, but [-r] can be a theme or a localization. This leaves two interpretations open, one in which the spool (theme) is taken from something, and one in which something is taken from a spool. Hence, in this kind of derivation it sometimes appears that we get two readings back from the knowledge base. For example, in English *unhand* seems to mean 'take a hand away from something' or 'release the hand's grasp on something'. But such cases are fairly rare.

In a forthcoming article, von Heusinger and Schwarze (2006) show that the semantic ambiguity of Italian removal verbs must usually be resolved within the derivation, because it fails to carry over into the sentence semantics. In fact, derivational rules like those we see here usually produce predicate-argument structures that are very vague, such as 'something-typically-done-with-spaghetti' (for Italian *spaghettata*), but the meanings and argument structures that result tend to be very specific ('a meal with spaghetti'). This means that conceptual evaluation is an important part of derivation. In the case of *sbobinare*, we must expect the query function FIND to return something like the result shown in 3 of Figure 3.

4.4 Semantic abstraction

I assume that what the knowledge base returns is a purely conceptual, framelike structure. Using a logic of proto-roles, perhaps like that of Dowty (2001), it should be possible to obtain simplified semantic abstractions that can be the base for conventional mapping algorithms; cf. (Kelling 2001), which shows how this can be done for two classes of French nominalizations. For *sbobinare* the abstraction would produce a semantic formula like that of 4 in Figure 3, containing semantic relations and typed argument variables, some of which may be bound (as is the argument of FROM in this case, which is bound to the semantics of SPOOL).

Needless to say, the implementation of the required data base and abstraction rules would not be trivial, and most of the (very extensive) work in computational knowledge representation has been done without a clear idea of what outputs might be useful at the interface to lexical semantics and mapping. The KLU knowledge base was, of course, just a dummy that provided a few pre-arranged answers to conceptual queries.

4.5 Mapping

Mapping must create the lexical form (the PRED value) and its argument structure and establish the mappings from the lexical form's argument structure to the participants of the associated semantic representation, as shown by the arrows between 4 and 5 of Figure 3. In KLU the input to mapping is always a semantic structure, not a lexical form. However, an often-voiced opinion is that derived words inherit their argument structure directly from the argument structure of the base, not from the semantics. A point in favour of this

view is that nonsense words can be used as the bases of derivatives. Thus if I can *frobble* something, I can say that it is *frobblable*, without ever having found out what it is to *frobble*. It is conceivable, however, that the knowledge base has a default class of generic transitive actions and returns an abstract generic concept for *frobble* which could be the basis for mapping.

A study by Meinschaefer (to appear) shows that certain nominalizations from verbs derive their argument structures not from the argument structures of the base verb but from an underlying, semantic structure shared with the verb. Moreover, the often cited restrictions on passivization (e.g., *the hat fits you well* vs. **you are fitted well by the hat*) suggest that even in the very regular passive, more is involved than reorganizing the syntactically visible argument structure of the base. The task of formulating the semantic decompositions that the mapping algorithm from semantics will require is far from completion, as Levin and Rappaport Hovav (2005) point out. The value of attempting a computational implementation, however sketchy, along the lines indicated here is that it forces the contributing theories to pay attention to the entire gamut of interfaces involved.

In the KLU system, mapping was also made a catch-all for creating features like inflectional class and declension, aspect, gender, etc. Clearly some of these require access to information in the morphological analysis that gets lost in conceptual analysis. For example, some Italian diminutives take the grammatical gender of their bases, regardless of any physical gender properties of the base, but in other cases it is the affix which determines gender and inflectional class.

At this point we can account for what happens when the grammatical system encounters a spontaneous derivation like *sbobinare*. The syntactic parser does not concern itself with the word-internal structure but turns the job over to c-insertion. c-insertion can perform inflectional analysis in well-understood ways. What it cannot do is derivational analysis, but it contains a sub-function that can, at least in principle.

5 The Morphological Buffer

Now I turn to the last of the three components, the morphological buffer, the interface between syntax and morphology. Strictly speaking, the buffer is only a data structure within c-insertion, but c-insertion might not be necessary if there were no buffer to administer. To help explain why the buffer is there, let me again return to *sbobinare*, but in a different usage meaning ‘transcribe’ rather than ‘pull from a spool’.

(4) La prego, mi dica: lo ha *sbobinato* soltanto, o lo *h*a scritto lei? (Scarpa 2004)
‘I ask you, please, tell me: did you only *transcribe* it or did you write it [yourself]?’

Judging from my searches in Google, this is now by far the most common reading of *sbobinare*. It must have arisen at a time when wire or tape recorders were commonly used for taking dictation, and secretaries transferred the spoken texts to paper by ‘pulling the speech’ from the spool. At a time when the word was unlikely to be in the Italian mental lexicon, a derivation in the way described earlier would have been easy enough for most speakers because a typical action involving pulling something from a spool was transcription. Each time a speaker made up or heard this use of *sbobinare*, a lexical entry would appear in her morphological buffer, and the more often it happened, the longer it would stay there. The entry in the lexical buffer, however, is a pure Saussurian sign. It has no internal structure; it is simply a pairing of a surface form with a semantic item. At some point, it will be learned and used by other, younger speakers, and will become a part of the Italian lexicon. At the same time, wire recorders and tape recorders will give way to cassette recorders and MP3 players, and the conceptual connection between pulling something from a spool and transcription disappears. The lexical data repeatedly created in the lower shaded box of Figure 2 gradually move to the upper shaded box, the static lexicon. (Computationally, this is the same sort of process that takes place in memory management systems when chunks or pages of memory become increasingly ‘non-purgable’, i.e.,

permanent parts of the loaded system.) Young speakers who do not learn *sbobinare* as an opaque sign will not produce it in this sense spontaneously, and they will probably have difficulty understanding sentences like (3).

Thus the morphological buffer would seem to be the mechanism by which lexicalization takes place. Its psychological reality has been confirmed in many experiments, and buffers of this form are an indispensable data structure in virtually all large-scale computer programs. That it has not been a part of computational LFG is a historically curious accident.

Interestingly enough, in the early days of LFG it was recognized that even where derivation is relatively simple and systematic, as it is with the passive, it can be computationally very expensive. It was apparently assumed that something like a morphological buffer must exist to retain the results of this expensive computation when it was unavoidable. The "Introduction" to *The Mental Representation of Grammatical Relations* Bresnan and Kaplan wrote

> ...lexical computations are not required in generating sentences, since ... lexical rules, as long as they have a finite output, can always be interpreted as redundancy rules ... As such, the rules could be applied to enter new lexical forms into the mental lexicon, and the derived lexical forms could subsequently be retrieved for lexical insertion rather than being re-derived (Bresnan and Kaplan 1982, xxxiii).

The picture we how have of lexical rules and derivation is, if anything, only more complex that the transformational accounts of 25 years ago. All the more reason why we should expect the synchronic grammar to use a buffering mechanism to avoid expensive computations as much as possible, even if it has access to the mechanisms that produce and analyse new words. Bresnan and Kaplan did not identify the lexical buffer as an entity distinct from the lexicon itself, with its own storage-managing regime. This seems to have led to endless misunderstandings and to a wide-spread impression that LFG and similar unification-oriented models of grammar cannot give a formal account of spontaneous word formation. Across research traditions, lexicalism has unfortunately been identified with a conception of the lexicon as a static set of well-formed words that cannot participate in the creative, spontaneous introduction of new forms.

To be sure, some computational projects in the 1990s did in fact consider, but did not implement, what would have been a morphological buffer. The authors of the Alvey Natural Language Tools, for example, thought a word-formation cache would speed up processing, but "would be of little linguistic interest" (Ritchie et al. 1992, 177).

The moral of the story is this: Computational models are not just the servants of theory; they also strongly influence how we think about theory. We should bear in mind that formal descriptions of computational systems are simplifications of underlying reality. Computational systems are full of buffer-like structures, and cognitive psychology tells us that our own brains are, too. The buffer between syntax and morphology allows the cognitive grammar to avoid the enormous costs that would result from continuously recomputing all complex words, and there is evidence that even very frequent inflected forms are stored rather than computed (Baayen et al. 1997). However, the inherent, context-dependent flexibility of conceptual interpretation can cause frequently buffered words to lose their connection to their compositional semantics, leading to lexicalized forms that cannot be reconstructed easily once the conceptual context that engendered them is not generally available.

6 An Afterthought: Separable Prefixes in Derivation

The model I have described depends crucially on the one-to-one correspondence of c-structure nodes to morphologically complete words, as required by Lexical Integrity. On a word-and-paradigm view of inflectional morphology, syntactic paraphrases are part of the inflectional system, so that this correspondence is

not always given: the surface constituents realizing a cell of the paradigm lie under different c-structure nodes, in apparent violation of Lexical Integrity. The same can be found in derivation. In Germanic languages, the so-called particle or separable prefix verbs defy analysis in the model I have shown. Consider (5), a non-lexicalized but semantically transparent derivation.

(5) Max wird seinen Mitgleidsbeitrag fur den Alpenverein abwandern.
'Max will hike off his dues to the Alpine Association' (Stiebels 1996, 143)

In the framework I have described, we can account for this derivation by associating the particle *ab* with a DPRED Decrement<-o,-r, (↑Arg)>. This requires, however, that (↑Arg) be within the boundaries of the morphologically complete word given to c-insertion. This is not the case when the particle is separated, as in (6).

(6) Max wandert seinen Beitrag zum Alpenverein ab.
'Max is walking off his dues to the Alpine Club.'

A possible solution might be to loosen the barrier imposed by c-insertion in the following sense: Following the suggestion of Frank and Zaenen (2004), we assume a projection logic that carries purely morphological features beyond the lexicon-syntax barrier, but assembles them in a sentence level m-structure. This would let *ab* find its base *wandern* in f-structure by specifying the path to its argument with functional uncertainty, i.e.,writing (↑ X* Arg) insead of (↑ Arg) in the DPRED of *ab*. An unpleasant consequence of this strategy is that, after the derivation, the lexical form of the base must be replaced in f-structure by the newly derived lexical form, and the derivation itself, as we have seen, cannot be fully accomplished within the existing formal apparatus of LFG. A call to m-insertion, with affix and base, is necessary from some point above lexical insertion. What's worse, the replacement must happen prior to the tests for completeness and coherence, because *wandern*, which is syntactically like English *wander*, cannot take a direct object. Conceivably, the derivation could be implemented in an extension to the constraint tests.

On the positive side, the length of the path from the base to its affix might provide a measure of grammaticality. This is useful for the following reason: Distributionally, the particle is similar to an adjunct, but in an interesting study Jochen Zeller (2003) shows that the position of the derivational particle in German is actually more restricted than that of an adjunct. Where the first sentence is fully acceptable, the second is judged as marginal.

(7) Laut quietschte die Ziehharmonika 'Loudly screeched the accordion'

(8) ?*Auf schrie die Zieharmonika ' The accordion shrieked' (Zeller 2003, 188)

From a statistical study of grammaticality judgments Zeller concludes that the particle must "be strictly head-governed by the verb" (p. 199), while admitting that it's difficult to give a precise movement analysis that would predict the degree of ungrammaticality. It would be interesting to see if path length in f-structure might provide the required quantitative measure.

References

Baayen, Harald; Cristina Burani; and Robert Schreuder (1997). Effects of semantic markedness in the processing of regular nominal singulars and plurals in Italian. In *Yearbook of Morphology* 1997, pp. 13-33.

Baayen, R. Harald and Antoinette Renoulf (1996). Chronicling the times: Productive lexical innovation in an English newspaper. *Language*, 72:1, pp. 69–96.

Bresnan, Joan, editor (1982). *The Mental Representation of Grammatical Relations*. Cambridge, MA: MIT Press.

Bresnan, Joan (1982). *Lexical Functional Syntax*. Malden, MA: Blackwell, 2001

Bresnan, Joan and Ronald M. Kaplan (1982). Introduction: Grammars as mental representations of language. In (Bresnan, ed., 1982) pp. xvii–lii.

Bresnan, Joan and Samuel A. Mchombo (1995). The lexical integrity principle. Evidence from Bantu. *Natural Language and Linguistic Theory* 13, pp. 181-254.

Briscoe, Ted and Ann Copestake (1999). Lexical Rules in Constraint-based Grammars. *Computational Linugistics* 25:5, pp. 487-526.

Börjars, Kersti, Nigel Vincent, and Carol Chapman (1996). Paradigms, periphrases and pronominal inflection: a feature-based account. In *Yearbook of Morphology*, pp. 155–180.

Corbin, Danielle (1990). *Associativité et stratification dans la represéntation des mots construits*. Contemporary Morphology. Berlin: de Gruyter.

Dowty, David (1991). Semantic Proto-roles and Argument Selection. *Language* 67:3, pp. 547-619.

Frank, Anette and Annie Zaenen (2004). Tense in LFG: Syntax and Morphology. In *Projecting Morphology*. Stanford: CSLI, pp. 23-65.

von Heusinger, Klaus and Christoph Schwarze (2006). Underspecification in the Semantics of Word-Formation. The Case of Denominal Verbs of Removal in Italian. *Linguistics* 44:6.

Karttunen, Lauri (2003). Computing with Realizational Morphology. In: Computational Linguistics and Intelligent Text Processing, Proceedings of the 4th International CICLing2003. Lecture Notes in Computer Science 2588. pp. 203-214. Berlin: Springer.

Kelling, Carmen (2001). Agentivity and suffix selection. In: Butt, Miriam and Tracey Holloway King, eds., Proceedings of LFG'01. University of Hong Kong. Stanford: CLSI, 147-162 (http://csli-publications.stanford.edu).

Kiparsky, Paul (1982). Word-formation and the lexicon. In: Frances Ingemann, editor. 1982 Mid-America Linguistics Conference, number 1982, pp. 3-29. Lawrence, KS, 1982. Linguistics Dept., Univ. of Kansas.

Koenig, Jean-Pierre and Anthony Davis (2006). The KEY to lexical semantic representations. *Journal of Linguistics*. 42, pp. 71-108.

Koenig, Jean-Pierre and Daniel Jurafsky (1994). Type underspecification and on-line type construction in the lexicon. In Raul Aranovich, Willian Byrne, Susanne Preuss, and Martha Senturia, editors, Thir-

teenth West-Coast Conference on Formal Linguistics, number 1994, pp. 270–285, University of Califorina at San Diego. Stanford: CLSI.

Levin, Beth and Malka Rappaport Hovav (2005). *Argument Realization*. Cambridge: Cambridge University Press.

Lieber, Rochelle (2004). *Morphology and Lexical Semantics*. Cambridge: Cambridge University Press.

Marslen-Wilson, William and Lorraine Komisarjevksy Tyler, Rachelle Waksler, and Lianne Older (1994). Morphology and meaning in the English mental lexicon. *Psychological Review*, 101:1, pp. 3–33.

Massari, Alessandro (2006). Il Great Barracuda. http://www.seaspin.com/magazine/articolo.php?idart=42

Mayo, Bruce (2000). A Computational Model of Derivational Morphology. Dissertation, University of Hamburg,. http://deposit.ddb.de/cgi-bin/dokserv?idn=961769629

Mayo, Bruce, Marie-Theres Schepping, Christoph Schwarze, and Angela Zaffanella (1995). Semantics in the derivational morphology of Italian: Implications for the structure of the lexicon. *Linguistics*, 33, pp. 883–938.

Meinschaefer, Judith (to appear). The syntax and argument structure of deverbal nouns from the point of view of a theory of argument linking. In: Dal, G.; Miller, P.; Tovena, L.; Van de Velde, D. (eds.) *Deverbal nouns*. Amsterdam: Benjamins.

Ritchie, Graeme D. and Graham J. Russell, Alan W. Black, and Stephen G. Pulman, editors (1992). *Computational Morphology: Practical Mechanisms for the English Lexicon*. ACL-MIT Press Series in Natural Language Processing. Cambridge, MA: MIT Press.

Scarpa, Tiziano (2004). Posted in vasicomunicanti on April 19th, 2004. http://www.nazioneindiana.com/2004/04/19/i-narrificatori/

Stiebels, Barbara (1996). *Lexikalische Argumente und Adjunkte*. studia grammatica 39. Berlin: Akademie-Verlag.

Zeller, Jochen (2003). Moved preverbs in German: Displaced or misplaced? In *Yearbook of Morphology* 2003, pp. 179-212.

* Until 1998. Author's current e-mail address: bruce.mayo@arcor.de