Encoding Constituent Structure in Feature Structures

James Kilbury
Seminar für Allgemeine Sprachwissenschaft
Heinrich-Heine-Universität Düsseldorf
Universitätsstraße 1
D-40225 Düsseldorf, Germany
kilbury@ling.uni-duesseldorf.de

Keywords: constituent structure, data structures, typed feature structures

Abstract

The paper discusses alternative representations of constituent structure with typed feature structures and introduces the notion of the perspective of a representation. Mirrors are then presented as a device for defining the relationship between perspectives. Examples are given in ALE. To illustrate the linguistic use of such alternative representations a data type is proposed which facilitates the description of subcategorization and phraseolexemes.

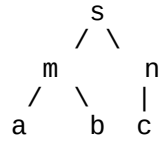
Das Papier behandelt alternative Repräsentationen von Konstituentenstruktur mit getypten Merkmalsstrukturen und führt den Begriff der Perspektive einer Repräsentation ein. Spiegel werden dann als Mittel zur Definition der Beziehung zwischen Perspektiven vorgestellt. Beispiele werden in ALE angegeben. Um die linguistische Verwendung solcher alternativer Repräsentationen zu illustrieren, wird ein Datentyp vorgeschlagen, der die Beschreibung von Subkategorisierung und Phraseolexemen erleichtert.

Encoding Constituent Structure in Feature Structures

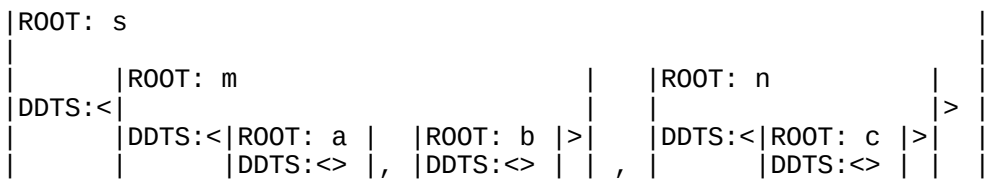
0 Introduction

It is well known in the framework of unification-based grammar formalisms that lists can be encoded in feature structures with the use of features *FIRST* and *REST* and some end-marker value *end* or *nil* (cf Shieber 1986:29). This simply borrows a fundamental representational device from LISP. Since nested lists can be used to encode phrase-structure trees representing constituent structure, the latter can also be encoded in feature structures. Thus, the phrase-structure tree of figure 1 can be encoded in the feature structure of figure 2:

(figure 1)



(figure 2)



Here explicit representation with the features *FIRST* and *REST* has been avoided with a conventional notation for lists employing pointed brackets "<" and ">" and commas to make the matrix notation of feature structures more readable. The empty list is designated with "<>", and the attribute *DDTS* stands for 'directly dominated trees'.

Representations like that of figure 2 have become so familiar within computational linguistics that they now are largely taken for granted. Nevertheless, the given representation, while itself purely *declarative*, embodies a particular *perspective* in its organization of the structural information, and this perspective is relatable to particular *procedural analysis strategies*, namely, top-down algorithms for the recognition of context-free languages.[1] The privileged position of the sort of representation in figure 2 is therefore illusory. This paper will present alternative representations of constituent structure embodying different perspectives, some corresponding to other algorithms for the analysis of context-free languages.

[1] In a similar vein, Gerdemann (1994) relates algorithms for parsing to those for tree traversal. The observation that declarative representations may show structural correspondences to procedural strategies is in itself no more paradoxical than the fact, documented in Pereira & Shieber (1987), that algorithms for context-free parsing can be encoded declaratively in pure Prolog.

Since the data types we shall define employ distinct attributes, the different perspectual representations of a phrase-structure tree can be encoded in a single feature structure encompassing all the perspectives. A formal device will be introduced to allow the transformation of each perspective into the other, equivalent perspectives. Just as in relativity theory within physics, no frame of reference or perspective is taken to be absolute or privileged. Thus, questions as to the absolute superiority of a single perspective are essentially misguided. Different perspectives will prove to be better or worse for particular purposes. Thus, a parser analyzing input from left to right may pose different representational requirements than one working from right to left. Compilation processes are likely to transform representations from one perspective to another without any awareness of the grammar writer.

1 Alternative Perspectives

Now let us consider in detail some alternative perspectives for representing constituent structure. As we have already seen, the representation of figure 2 corresponds to a top-down, in particular, a top-down breadth-first analysis; the perspective or data type of the feature structure will therefore be called a *top-down breadth-first tree* (*tdbf*). Using the notion of typed feature structures of Carpenter (1992) and the formalism ALE as presented in Carpenter & Penn (1994), the data type *tdbf* can be defined in the following partial type signature:

```
tdbf sub []
     intro [root:node, ddts:ddt_list].          % directly dominated trees

ddt_list sub [e_ddt_list, ne_ddt_list].         % (non)empty ddt-list
   e_ddt_list sub [].
   ne_ddt_list sub []
               intro [first:tdbf, rest:ddt_list].
```

The definition states that a feature structure of type *tdbf* contains specifications of the attributes *ROOT* and *DDTS*. The latter has a list of feature structures of type *tdbf* as its value, while the former has a value of type *node*. This may simply be an atomic categorial label such as *NP* or a bundle of phonological, syntactic, and semantic information.

The following two data-type definitions simply serve to illustrate alternative perspectives:

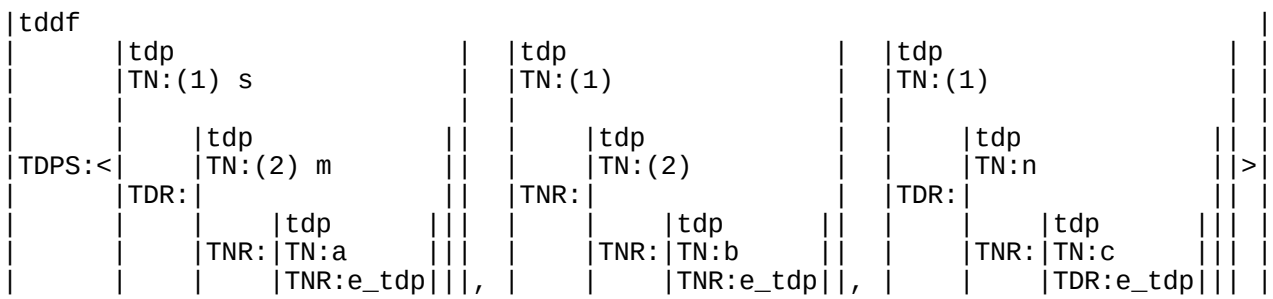
```
tddf sub []                                     % top-down depth-first tree
     intro [tdps:tdp_list].                     % top-down paths

tdp_list sub [e_tdp_list, ne_tdp_list].
   e_tdp_list sub [].
   ne_tdp_list sub []
               intro [first:tdp, rest:tdp_list].

tdp sub [e_tdp, ne_tdp].                          % top-down path
   e_tdp sub [].
   ne_tdp sub []
          intro [tn:node, tdr:tdp]              % top node, top-down rest
```

```
but sub []                                          % bottom-up tree
    intro [bups:bup_list].                          % bottom-up paths

bup_list sub [e_bup_list, ne_bup_list].
   e_bup_list sub [].
   ne_bup_list sub []
               intro [first:bup, rest:bup_list].

bup sub [e_bup, ne_bup].                            % bottom-up path
   e_bup sub [].
   ne_bup sub []
          intro [bn:node, bur:bup].           % bottom node, bottom-up rest
```

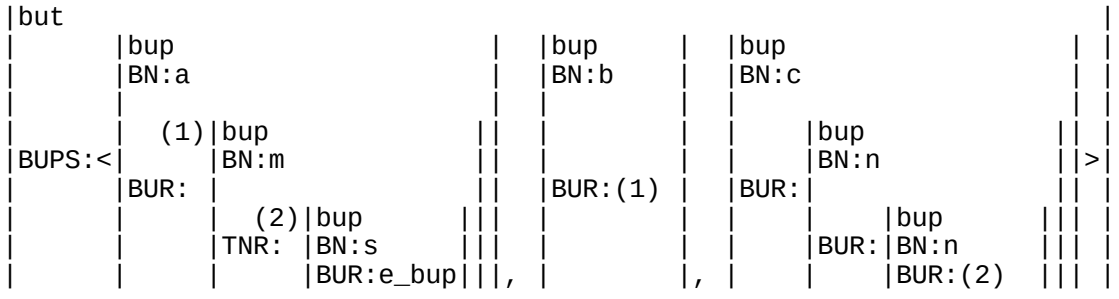
The latter data types simply amount to lists of paths, although the directions of the paths differ. All the definitions could be simplified if ALE made parametrized types available for lists containing elements of a given type.

Typed feature structures illustrating the two perspectives are given in figures 3 and 4. The latter is of interest because of the generalizations it captures with integer indices for shared values, whereas the indices of figure 3 are rather clumbsy:

(figure 3)



(figure 4)

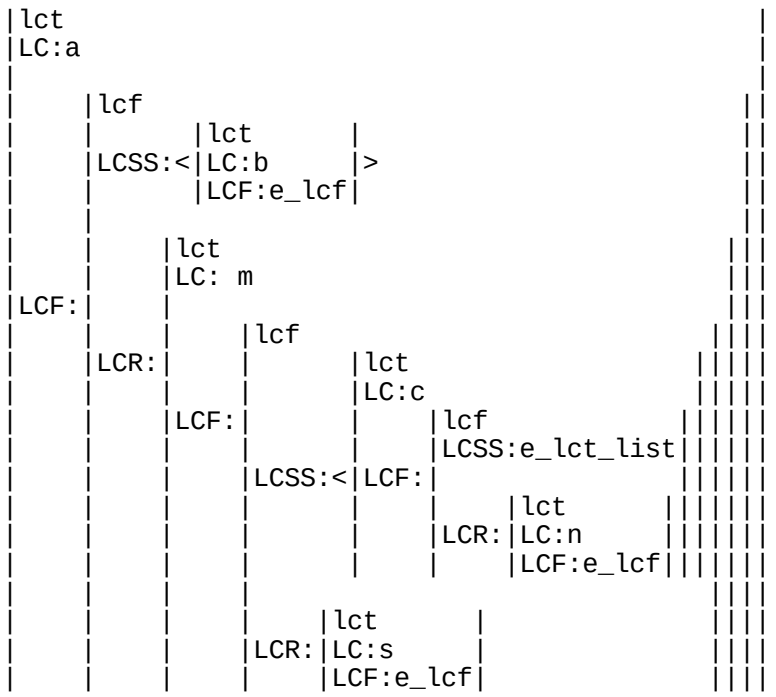


The following, fourth data type, with its illustration in figure 5, shows a remarkable reorganization of the information about constituent structure:

```
lct sub []                                          % left-corner tree
    intro [lc:node, lcf:lcf].                       % left-corner frame

lcf sub [e_lcf, ne_lcf].                            % (non-)empty
   e_lcf sub [].
   ne_lcf [lcss:lct_list, lcr:lct].                 % lc-sisters, lc-rest
```

```
lct_list sub [e_lct_list, ne_lct_list].
   e_lct_list sub [].
   ne_lct_list sub []
               intro [first:lct, rest:lct_list].
```

(figure 5)



Like the data type *but*, *lct* corresponds to a type of bottom-up analysis strategy but is *centered* around a constituent of particular significance for a certain parsers -- the left corner. This notion of a *centered representation* will be taken up again later in connection with syntactic heads. Note that the data type *lct*, like *tdbf* but unlike *tddf* and *but*, makes no use of shared values corresponding to indices.

2 Mirrors Relating Different Perspectives

The four data types *tdbf*, *tddf*, *but*, and *lct* can be subsumed by a general type *tree* so that the different perspectives are superimposed on each other in a single feature structure. Special definite clause definitions of ALE that we shall call *mirrors* then serve to relate the different perspectives to each other. In order to illustrate such mirrors, we first present an ALE signature that relates forward-viewed lists to backward-viewed lists in typed features structures that represent doubly-linked lists:[2]

[2] See Walther (1995) for a closely related treatment of typed feature structures for doubly-linked lists in a phonological application. The definite clause definition given here corresponds to that of Clocksin & Mellish (1984:150). The ALE signatur should be correct, but in fact the definite clause leads to an endless loop in execution; we suspect that this stems from an error in the ALE implementation.

```
list sub [flist, blist].
   flist sub [e_flist, ne_flist].            % forward list
      e_flist sub [].                        % empty
      ne_flist sub [ne_list]                 % nonempty
               intro [fi:bot, ri:flist].     % fi(rst), ri(ght)
   blist sub [e_blist, ne_blist].            % backward list
      e_blist sub [].
      ne_blist sub [ne_list]
               intro [la:bot, le:blist].     % la(st), le(ft)
         ne_list sub [ulist, llist].
            ulist sub [].                    % unary list (one element)
            ulist cons (fi:X, la:X,
                        ri:e_flist, le:e_blist).
            llist sub [].                    % longer list (more elements)
            llist cons (fi:F, ri:R, la:L, le:LE)
                  goal mirror((fi:F, ri:R), e_blist, (la:L, le:LE)).


mirror(e_flist, BList, BList) if true.

mirror((ne_list, fi:F, ri:R), BL1, BL2) if
   mirror(R, (ne_list, la:F, le:BL1), BL2).
```

[[** An analogous signature will capture the relation between the data types *tdbf* and *lct* in the final version of this paper. **]]

Mirrors are thus constraints expressing general principles of correspondence between perspectives. When the appropriate mirror is applied recursively to a feature structure of type *tdbf* it produces a *reflection* in the perspective of the data type *lct* (and *vice versa*).

Note that the perspectives differ in the amount of information (measurable in bits, perhaps) that they require in order to represent a phrase-structure tree. The feature structures of figures 2, 3, 4, and 5 require 12, 19, 13, and 18 feature specifications, respectively, to represent the constituent structure of one and the same tree. While this clearly has both theoretical and practical consequences for the representation and storage of information, the point will not be further pursued here.

3 Alternative Perspectives for Subcategorization

Adequate linguistic representations must take considerations into account that have not yet been mentioned here. In particular, it is desirable for the representations of endocentric constructions to be centered explicitly on their syntactic *heads*. Such *head-centered representations* have their analogs in Head-Driven Phrase Structure Grammar (HPSG). In particular, the data type *headed-structure* (i.e. endocentric construction) draws a structural distinction between head daughters and complement daughters (Pollard & Sag 1987:56). Ignoring fillers and many other points of HPSG not directly relevant to this discussion, we can abstract out a data type *top-down head-centered tree* as the fundamental perspective for the representation of endocentric constituent structure in HPSG:

```
tdhc sub []                                          % top-down head-centered
     intro [mthr:node, dtrs:dtrs].                   % m(o)th(e)r, d(augh)t(e)rs

dtrs sub [e_dtrs, ne_dtrs].
   e_dtrs sub [].
   ne_dtrs sub []
           intro [head_dtr:node, comp_dtrs:dtr_list].

dtr_list sub [e_dtr_list, ne_dtr_list].
   e_dtr_list sub [].
   ne_dtr_list sub []
               intro [first:tdhc, rest:dtr_list].
```

The *SUBCAT* feature of HPSG specifies a list of subcategorized constituents for each head; the relation of this subcategorization list to the list of complement sisters of the head is then determined by the Subcategorization Principle (ibid., 71). The compositional semantic relation between mothers and heads is governed by the Semantics Principle (ibid., 99), which is essentially the semantic counterpart of the syntactic Head-Feature Principle (ibid., 58).

The organization of subcategorization information in HPSG and similar formalisms is shaped by the top-down head-centered perspective taken for the representation of constituent structure. This choice leads to difficulties in representing the dependencies between the mothers and complements of lexical heads that are found in collocations, idioms, and phraseolexemes in general for which the semantics is not strictly compositional. Thus, the semantics of English *make* covaries with that of the lexical head of its complement (*pottery* ~ 'create through manipulation', *decisions* ~ 'arrive at', *money* ~ 'acquire', *mistakes* ~ 'undergo', *sense* ~ 'exhibit', etc.), but HPSG provides no device to capture such dependencies.

Help can be provided by introducing an additional perspective for the representation of constituent structure. As in conventional HPSG, the new perspective is head-centered but bottom-up rather than top-down so that mothers and complements of lexical heads may be grouped together naturally; moreover, the perspective is *lexically centered*.[3] This additional data type for representing endocentric constructions may be defined in ALE as follows:

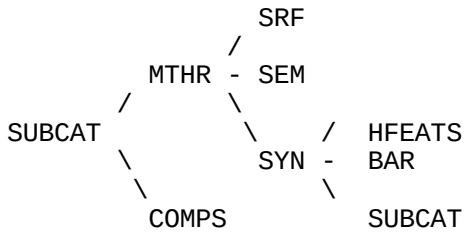
```
buhc sub []                                          % bottom-up head-centered
     intro [head:node, hf:hf].                       % head frame

hf sub [e_hf, ne_hf].
   e_hf sub [].
   ne_hf sub []
         intro [mthr:buhc, comps:buhc_list].         % complements

buhc_list sub  [e_buhc_list, ne_buhc_list].
   e_buhc_list sub [].
   ne_buhc_list sub []
                intro [first:buhc, rest:buhc_list].
```

[3] Note the related material on alternative representation of constituent structure in HPSG in König (1994) and unpublished work by Mark Johnson.

As discussed above, mirrors may be defined to relate the data types *tdhc* and *buhc*.

Now that the constituent structure has been, so to speak, turned on its side, the subcategorization information may be regrouped so as to capture mother-complement dependencies. The feature *SUBCAT* is made recursive and now appears in the configuration shown in figure 6:

(figure 6)



HPSG itself already provides for the subcategorization of both lexical and phrasal heads (ibid., 68). Since the subcategorization of a lexical head in this treatment has the subcategorizations of the head's projections embedded within it, a separate Subcategorization Principle relating subcategorizations to constituent structure becomes superfluous; adjuncts, fillers, and linearization can be dealt with separately.

With such a modification, HPSG takes on an appearance much more like that of Categorial Grammar. The treatment of subcategorization sketched here provides structural positions for the representation of noncompositional phraseolexemes. Thus, the semantic representation of *kick the bucket* is associated with the entire verb phrase, and a natural account is possible for modifications like *kick the bucket suddenly/*hard* (since *suddenly'* but not *hard'* applies to *die'*). At the same time, the lexical entries for phraseolexemes can be embedded within the entries of their lexical heads.

4 Conclusion

The paper has presented alternative representations of constituent structure and shown how such perspectives can be related to each other with mirrors within the formalism ALE. The linguistic application of the particular perspective of bottom-up head-centered representation has been proposed as an aid in dealing with problems involving subcategorization and phraseolexemes. Representations analogous to those discussed here are currently being explored in the area of word formation.

References

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

Carpenter, Bob / Penn, Gerald (1994) *ALE: The Attribute Logic Engine User's Guide*, Version 2.0.1. Carnegie Mellon University.

Clocksin, W. F. / Mellish, C. S. (1984) *Programming in Prolog*, second edition. Berlin et al.: Springer.

Gerdemann, Dale (1994) Parsing as Tree Traversal, *Proceedings of COLING 94*.

König, Esther (1994) *A Study in Grammar Design* (= Report Nr. 54 - 1994, SFB 340). Heidelberg.

Pereira, Fernando C. N. / Shieber, Stuart M. (1987) *Prolog and Natural-Language Analysis* (= *CSLI Lecture Notes*, No. 10). Stanford, Calif.: CSLI.

Pollard, Carl / Sag, Ivan A. (1987) *Information-Based Syntax and Semantics*, Vol 1. (= *CSLI Lecture Notes*, No. 13). Stanford, Calif.: CSLI.

Shieber, Stuart M. (1986) *An Introduction to Unification-Based Approaches to Grammar* (= *CSLI Lecture Notes*, No. 4). Stanford, Calif.: CSLI.

Walther, Markus (1995) A Strictly Lexicalized Approach to Phonology, in *Integrative Ansätze in der Computerlinguistik*, ed. by James Kilbury & Richard Wiese, 108-113. U. Düsseldorf.