(WG 1 and WG 2)

A unified metagrammar approach to the implementation of MWEs in LTAG

Timm Lichte
CRC 991, University of Düsseldorf
lichte@phil.hhu.de

Simon Petitjean
CRC 991, University of Düsseldorf
simon.petitjean@hhu.de

Abstract In Lexicalized Tree-Adjoining Grammar (LTAG), multi-word expressions (MWEs) with an idiomatic semantics are usually implemented in an anchor-driven fashion: the idiomatic semantics is coupled with the lexical anchor and both are then inserted into appropriate tree templates. However, with the advance of analyses where richly structured semantic representations are flexibly linked to syntactic structures, this implementational approach appears to be too cumbersome. In this work (in progress) we instead propose a unified metagrammar approach where lexical anchors, their idiomatic semantics and the tree templates are jointly treated within the metagrammar. In our case, the metagrammar framework is eXtensible MetaGrammar (XMG).

MWEs in LTAG Lexicalized Tree-Adjoining Grammar (LTAG, [3]) is renowned to provide elegant accounts to a range of multi-word expressions with idiomatic meaning (e. g. [1]). The reason is that elementary trees of an LTAG can be made as large as is necessary to span any multi-word expression, even discontinuous or clausal ones, as elementary trees come with an extended domain of locality (EDL). An example is shown in Figure 2 with a frame-based semantics following [4]. Due to the flexible linking of syntax and semantics by means of interface variables (see boxed numbers), internal and external modification can be adequately handled.

Implementation with XMG In order to turn paper-and-pencil analyses such as in Figure 2 into an electronic resource, we make use of eXtensible MetaGrammar (XMG, [2]). XMG provides description languages and dedicated compilers for generating linguistic resources. The metagrammatical descriptions of XMG are packed into CLASSES, which have the following general shape:

$$\begin{aligned} \mathit{Class} &:= \mathit{Name} \rightarrow \mathit{Content} \\ \mathit{Content} &:= \langle \mathit{Dimension} \rangle \{ \mathit{Description} \} \mid \mathit{Name} \mid \\ &\quad \mathit{Content} \vee \mathit{Content} \mid \mathit{Content} \wedge \mathit{Content} \end{aligned}$$

The first rule corresponds to the notion of abstraction: a class allows to associate some content to an identifier. The second rule features the three other central concepts of XMG, namely dimension, conjunction and disjunction. Thus, the content of a class can be (i) a description attributed to some dimension, (ii) the content inherited from another class, (iii) the conjunction of contents, or (iv) the disjunction of contents.

DIMENSIONS are the crucial elements of a class. They can be equipped with specific description languages and are compiled independently, thereby enabling the grammar writer to treat the levels of linguistic information separately. In the following will be using the standard dimension `<syn>` for the syntax, and the more recently introduced `<frame>`-dimension for the frame-based semantics [5]. A code example is provided in Figure 1 on the next page.

Implementation with external anchoring The standard approach to the implementation of LTAG analyses is to dissociate the LEXICAL ANCHORS (e.g. *kicked*, *the*, *bucket* in Figure 2) from the TREE TEMPLATE (i.e. the unlexicalized elementary tree). The tree template is then described by the metagrammar, whereas the lexical anchors are dealt with as part of a two-level lexicon. In the latter one, roughly speaking, full forms are mapped onto lemmas, and lemmas are again mapped onto tree templates or tree families (i.e. sets of tree templates). This general procedure, which is for example applied in XTAG [6], is shown in Figure 3. From our point of view, the problem of this approach is that it does not easily allow for attaching the lexical semantics in such way that the intended linking is established. For example, during lexi-

cal insertion, the subject node of the tree template would have to be explicitly addressed by the lexical anchor(s) in order to link it to the PATIENT of *dying*. This might be doable, but it crucially depends on the interface of lexical insertion.

Implementation with internal anchoring Instead we propose a unified approach in which the anchoring happens inside the metagrammar. Hence, the metagrammar describes complete elementary trees and their semantics, such as in Figure 2, not just the underlying tree templates.

An example for implementing *kicked the bucket* this way is shown in Figure 1, using XMG code, and Figure 4. It consists of two classes: `nx0Vnx1`, which contributes the generic syntactic structure of a transitive verb, and `kicked_the_bucket`, which reuses `nx0Vnx1` and adds to it the lexical anchors and their idiomatic semantics.

```
class nx0Vnx1
export ?S ?Subj ?VP ?V ?Obj
declare ?S ?Subj ?VP ?V ?Obj ?X0
{
  <syn>{
    node ?S [cat=s, e=?X0] {
      node ?Subj [cat=np]
      node ?VP [cat=vp, e=?X0] {
        node ?V [cat=v, e=?X0]
        node ?Obj [cat=np] }}}
}

class kicked_the_bucket
import nx0Vnx1[]
declare ?X0 ?X1
{
  <syn>{
    node ?Subj [i=?X1];
    node ?V {
      node [lex=kicked, e=?X0] };
    node ?Obj {
      node [cat=d] {
        node [lex=the] }
      node [cat=n, i=?X0] {
        node [lex=bucket] }}
  }
  ;
  <frame>{
    ?X0[dying,
      patient:?X1]
  }
}
```

Figure 1: XMG code of the metagrammatical description of *kicked the bucket*

The **<syn>**-dimensions contains tree descriptions, using a bracket notation: **node** `?S{` **node** `?NP` **node** `?VP}` means that a node, associated to the variable `?S`, immediately dominates the nodes `?NP` and `?VP`, which are linearly ordered according to the order of description. Nodes can also be decorated with features structures. Here, for example, `?S` comes with a feature structure meaning that its syntactic category is `s`, and the value of the feature `e` is the unification variable `?X0`. The **<frame>** dimension holds typed feature structure descriptions wherein the variables `?X0` and `?X1` reappear. This sharing of variables across **<frame>** and **<syn>** is responsible for the linking between syntactic positions and positions in the semantics.

Perspectives There are many more details and alternatives to be explored. For example, one could consider weakly external anchoring where the lexical insertion sites would be morphologically more underspecified compared to the presented internal anchoring solution. Moreover, we think that XMG could be useful as a general tool for describing MWEs, as it allows for factorized and multi-dimensional descriptions in a very flexible manner. For this reason, we are currently also trying to improve XMG by simplifying, e.g., the description language for syntactic trees.

References

[1] Abeillé, Anne & Yves Schabes. 1996. Non-compositional discontinuous constituents in tree adjoining grammar. In H. Bunt & A. van Horck (eds.), *Discontinuous constituency*, 279–306. Berlin: Mouton de Gruyter.

[2] Crabbé, Benoit, Denys Duchier, Claire Gardent, Joseph Le Roux & Yannick Parmentier. 2013. XMG: eXtensible MetaGrammar. *Computational Linguistics* 39(3). 1–66.

[3] Joshi, Aravind K. & Yves Schabes. 1997. Tree-Adjoining Grammars. In G. Rozenberg & A. Salomaa (eds.), *Handbook of formal languages*, vol. 3, 69–124. Berlin, New York: Springer.

[4] Kallmeyer, Laura & Rainer Osswald. 2013. Syntax-driven semantic frame composition in Lexicalized Tree Adjoining Grammar. *Journal of Language Modelling* 1. 267–330.

[5] Lichte, Timm & Simon Petitjean. 2015. Implementing semantic frames as typed feature structures with XMG. *Journal of Language Modelling* 3(1). 185–228.

[6] XTAG Research Group. 2001. A Lexicalized Tree Adjoining Grammar for English. Tech. rep. Institute for Research in Cognitive Science, University of Pennsylvania Philadelphia, PA.

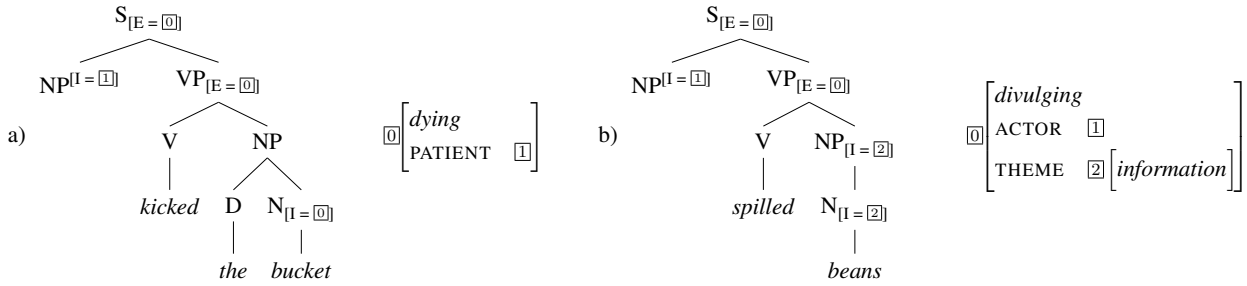

Figure 2: Elementary tree and frame semantics of the idiomatic multi-word expression *kick the bucket* and *spill the beans*

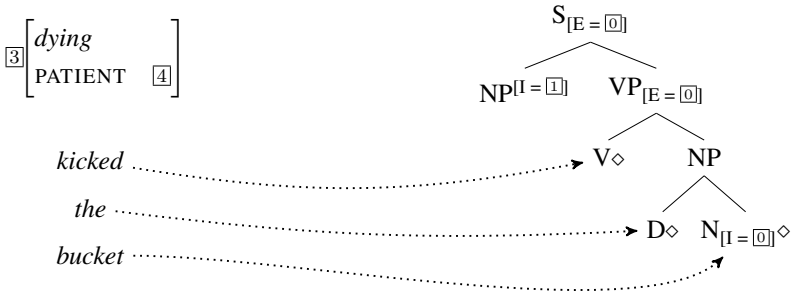

Figure 3: Implementation with external anchoring, which yields the elementary entry for *kicked the bucket* in Figure 2. The the tree template is taken from [6], yet slightly modified. The leaves with ◇-symbol mark the sites of lexical insertion.

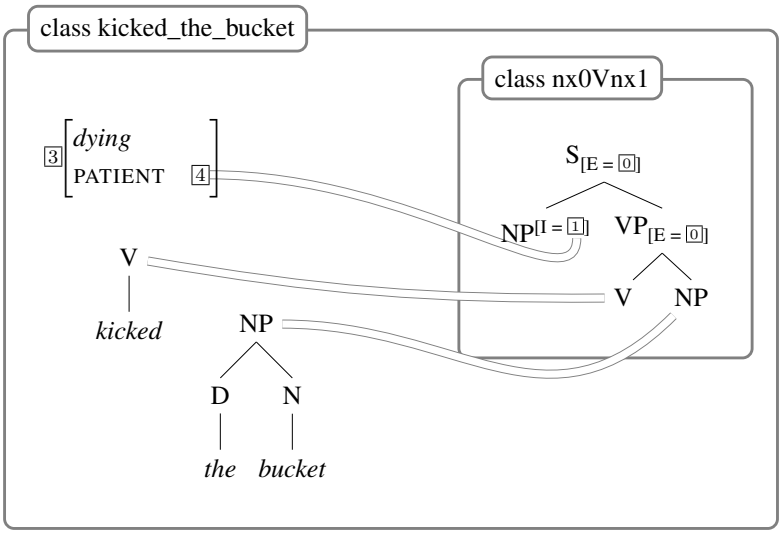

Figure 4: Implementation with internal anchoring according to the XMG code in Figure 1. Boxes stand for classes and double edges indicate identity constraints