# Coupling Trees, Words and Frames through XMG[⋆]

Timm Lichte[1], Alexander Diez[1], and Simon Petitjean[2]

[1] University of Düsseldorf, Germany
[2] University of Orléans, France

**Abstract.** This work presents first results on the integration of frame-based representations into the lexical framework of eXtensible MetaGrammar (XMG). Originally XMG supported tree-based syntactic structures and underspecified representations of predicate-logical formulae, but the representation of frames as a sort of typed feature structures failed due to reasons of usability and completeness. Therefore, after having explored strategies to simulate frames within the original XMG, we introduce an extension that is capable of handling frames directly by means of a novel `<frame>`-dimension. The `<frame>`-dimension can be also applied to recent accounts of morphological decomposition, as we show using a refined version of the `<morph>`-dimension from [4]. The presented extensions to XMG are fully operational in a new prototype.

## 1 Introduction

Recent work [9, 10, 16] has shown increasing interest in coupling a frame-based semantics with a tree-based syntax such as Tree Adjoining Grammar (TAG, [7]). While having lead to promising results on the theoretic side, it is unclear how to implement these ideas with existing grammar engineering tools, let alone how to bring them alive in natural language parsing. In this paper, we present first results on the integration of frame-based representations into the lexical framework of eXtensible MetaGrammar (XMG, [3]). XMG originally supported tree-based syntactic structures and underspecified representations of predicate-logical formulae, but the representation of frames as a sort of typed feature structures failed due to reasons of usability and completeness. Therefore we extend XMG by a novel `<frame>`-dimension that makes it capable of handling frames directly. This feature also paves the way for implementing recent accounts to morphological decomposition, such as in [16], where morphemes are linked to a frame-semantic representation.

The paper proceeds as follows. The next section briefly illustrates the lexical objects that we are concerned with, and Section 3 then shows the proposed

[⋆] This paper has greatly benefited from discussions with Laura Kallmeyer, Rainer Osswald and Yulia Zinova. We also thank the reviewers of HMGE 2013 for valuable comments. The work presented in this paper was financed by the Deutsche Forschungsgemeinschaft (DFG) within the CRC 991.

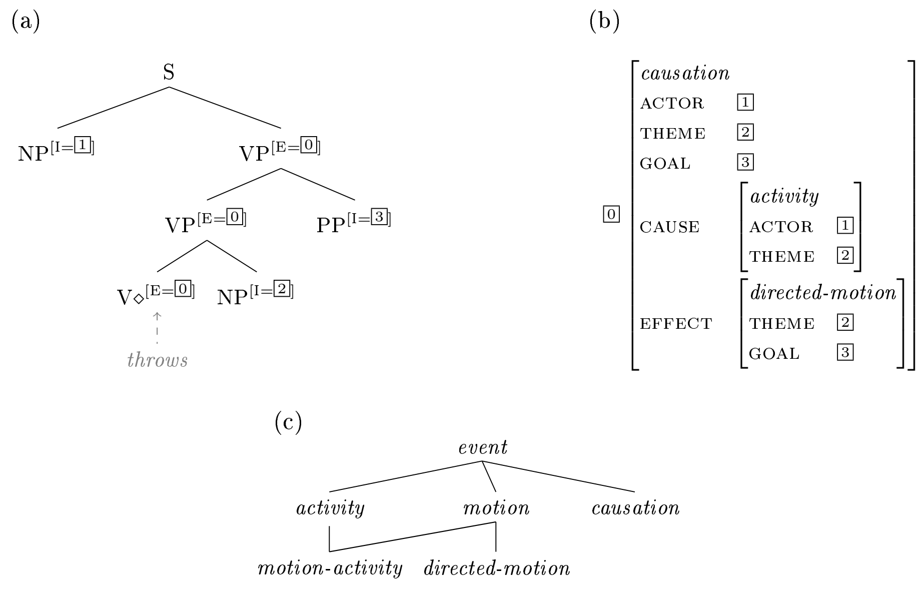


**Fig. 1.** A tree template, its frame-semantic counterpart and the associated type hierarchy.

factorization which crucially guides the implementation with XMG. After having explained the basics of XMG in Section 4, and after having explored strategies to simulate frames within the `<sem>`-dimension in Section 5, we present the `<frame>`-dimension in Section 6. Finally, Section 7 covers the application to a frame-based analysis of morphological decomposition.

## 2 A frame-based semantics for LTAG

We are concerned with lexical objects such as in Fig. 1, namely a (partial) phrase structure tree, a typed feature structure, and the associated type hierarchy, all of which are taken from [10, 11].

Phrase structure trees such as (a) make up the lexicon of a TAG, which is why they are also called *elementary trees*. TAG is a grammar formalism based on tree-rewriting, meaning that elementary trees can be combined (by means of two basic operations, substitution and adjunction) to generate larger trees.[3] A lexicalized TAG (LTAG) adds the constraint that every elementary tree includes at least one nonterminal leaf, the *lexical anchor*. Note however that throughout this paper we are rather concerned with so-called tree templates, which lack a

[3] Since in the present paper we are mainly concerned with the organization of the lexicon and thus focus on single elementary trees, we skip most details of the formalism here. See [7] or [1] for comprehensive presentations.

lexical anchor and from which elementary trees are lexically derived. The site of lexical insertion, here of *throws*, is marked by means of the ⋄-symbol. Finally the nodes of both elementary trees and tree templates carry (non-recursive) feature structures, as shown at least in some nodes of (b) in Fig. 1. This is relevant for the interface between tree and frame, since following [9, 10, 16] the interface is indicated by means of co-occurring boxed numbers. For example, this way the subject NP-leaf is linked with the ACTOR role(s) of the frame, eventually causing the unification of the ACTOR role and the frame of the incoming NP.

Typed feature structures such as (b), on the other side, are a common representation of so-called frames (see [14]), which, according to Frame Theory [5], are considered a proper representation of mental concepts. As can be seen from the example in Fig. 1, features describe semantic participants and components, while feature structures correspond to conceptual objects, restricted by the type (*causation*, *activity*, ...) that they are associated with. Moreover types are part of a type hierarchy, which determines the set of appropriate semantic features and moreover the unifiability of feature structures. Finally, boxed numbers indicate reentrancies, i. e. structure identity, which may cause a features structure to correspond to a graph rather than to a tree. This also holds for the feature structure in Fig. 1.

## 3 Factorization of tree templates and frames

Richly structured lexical objects like those in Fig. 1 make necessary some kind of metagrammatical factorization, once a large coverage grammar gets compiled and maintained [15]. Metagrammatical factorization roughly means to define recurring subcomponents of lexical objects, which can then be combined in several ways, for example in a purely constraint-based fashion as is the case in XMG. In addition to the benefit in terms of grammar engineering, however, [9–11] claim that metagrammar factorization can be also used to reflect constructional analyses in the spirit of Construction Grammar [12, 6]. Under this perspective the lexical material as well as the used "constructions" contribute meaning.

Taking these two aspects into account, [11] propose to factorize the tree template and the frame in Fig. 1 along the lines of Fig. 2, where boxes stand for the resulting factors or classes (i. e. classes in the sense of XMG), consisting of a tree and a frame fragment.[4] It illustrates that the tree-frame couple in Fig. 1 results from instantiating the class POConstr, which combines the classes n0Vn1 and DirPrepObj-to. Note that Fig. 2 shows a part of the proposed factorization only, as for example the class n0Vn1 would result from combining three other classes (Subj, VSpine, DirObj). Combining two classes essentially means that all associated information is unified, from which a minimal model is computed (see next section). Finally, one constructional facet can be found in the class POConstr in that it only contributes a frame fragment, but no tree fragment.

[4] Furthermore double edges indicate identity constraints, while within trees dashed edges represent non-strict dominance and $\prec^*$ non-strict precedence.

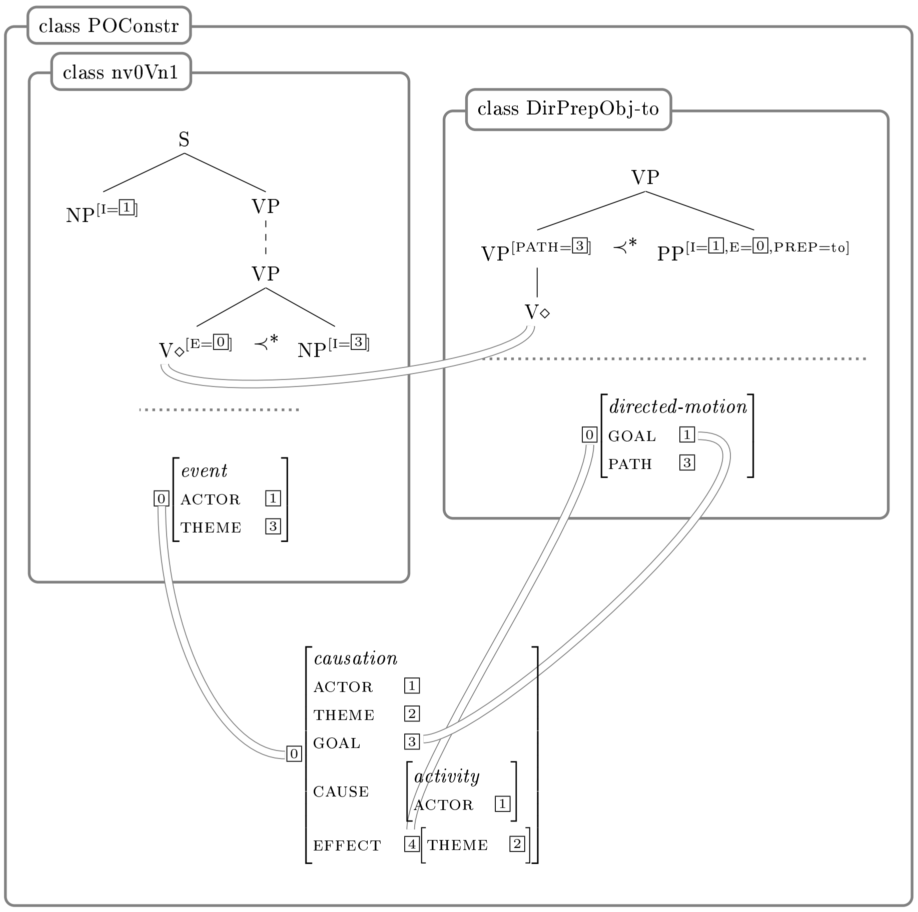


**Fig. 2.** Metagrammatical factorization of the tree template and the frame in Fig. 1.

The graphic representation of the metagrammatical factorization in Fig. 2 remains at a rather informal level and the question arises, how this could be translated into XMG code. We will see in Section 5 that the original XMG did not give a straightforward answer due to reasons of usability and completeness – other than the new `<frame>`-dimension, which is presented in Section 6.

## 4 A brief introduction to XMG

XMG (eXtensible MetaGrammar, [3]) stands both for a metagrammatical description language and the compiler for this language. Descriptions are organized into classes, that can be reused (i. e. "imported" or instantiated) by other classes. Borrowing from object oriented programming, classes are encapsulated, which

```
class n0Vn1
...
<syn>{
  node ?S [cat=s];  node ?VP1 [cat=vp]; node ?VP2 [cat=vp];
  node ?SUBJ [cat=np];  node ?OBJ [cat=np]; node ?V [cat=v];
  ?S->?SUBJ; ?S->?VP1; ?VP1->*?VP2; ?VP2->?OBJ; ?VP2->?V;
  ?V>>*?OBJ
}
...
```

**Fig. 3.** The `<syn>`-dimension of class n0Vn1.

means that each class can handle the scopes of their variables explicitly, by declaring variables and choosing which ones to make accessible for other instantiating classes (i. e. to "export"). The namespace of a class is then composed of the declared variables and all the variables exported by the imported classes.

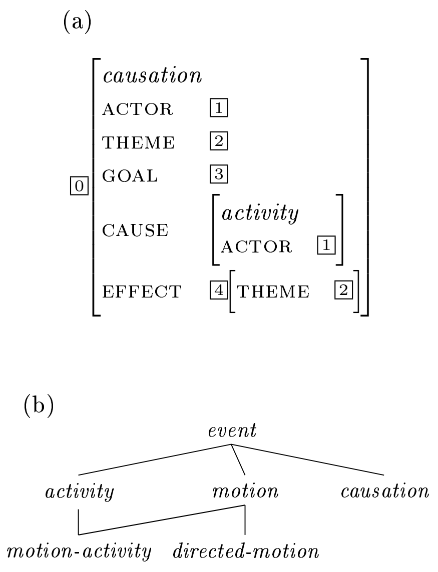
Crucial elements of a class are the so-called dimensions. Dimensions can be equipped with specific description languages and are compiled independently, therefore enabling the grammar writer to treat the levels of linguistic information separately. The `<syn>`-dimension, for example, allows to describe TAG tree templates (or fragments thereof). To give a concrete example, Fig. 3 shows the `<syn>`-dimension of class n0Vn1 from Fig. 2. It shows two sorts of statements, namely those like '`node ?S [cat=s];`' that instantiate nodes of the trees, and those like '`?S->?SUBJ;`' which determine the relative position of two nodes in the trees by referring to dominance and linear precedence.[5] In contrast, the `<sem>`-dimension includes descriptions of a different language as we will see in the following section.

When the metagrammar is compiled, first a set of descriptions for each class under evaluation is accumulated, and then the accumulated descriptions are resolved to yield minimal models. In the case of `<syn>`, the solver computes tree templates as minimal models, which is to say that only those nodes are included that are mentioned in the description. The final result can be explored with a viewer, or exported as XML file to feed a parser (e. g. the TuLiPA parser [8]).

## 5 Implementation within the `<sem>`-dimension

As mentioned before, the `<sem>`-dimension is designed to contain underspecified, flat formulae of predicate logic (borrowing from [2]). It is possible, however, to simulate frames by using binary predicates, such that they represent single semantic features. For example, a frame such as $\boxed{0}$[ACTOR $\boxed{1}$] is translated into the binary predicate `actor(?X0,?X1)`. A more detailed example for the class POConstr is shown in Fig. 4. The implementation of the syntax-semantics interface is straightforward, since the same variables can be used in both dimensions

[5] In XMG, variable names are prefixed with a question mark ('?').



(c)

```
class POConstr
...
<sem>{
  actor(?X0,?X1);
  theme(?X0,?X2);
  goal(?X0,?X3);
  cause(?X0,?X5);
  effect(?X0,?X4);
  actor(?X5,?X1);
  theme(?X4,?X2);

  % causation type
    event(?X0,+);
    activity(?X0,-);
    motion(?X0,-);
    causation(?X0,+);
  % activity type
    event(?X5,+);
    activity(?X5,+);
    directed-motion(?X5,-);
    causation(?X5,-)
}
...
```

**Fig. 4.** The feature structure of POConstr (repeated from Fig. 2), the corresponding type hierarchy (repeated from Fig. 1), and its simulation inside the `<sem>`-dimension.

`<syn>` and `<sem>`. Taking the class n0Vn1 from Fig, 2 as an example, the variable `?X1` would also appear in the XMG description of the subject-NP leaf, and the variable `?X2` in the description of the object-NP leaf (later identified with `?X3` in the class POConstr). We skip a code example due to lack of space.

While the simulation of the frame via binary predicates is straightforward, it is far less obvious how to simulate types and the type hierarchy. However, as can be seen again from Fig. 4, types can be also simulated by sets of binary predicates with a boolean second argument, or *type simulating sets* (TSS) as we call them, in the following way:[6] given a type hierarchy $T$ such as the one in Fig. 4, we say that $t$ in $T$ is simulated by the minimal set of predicates $P_t(?X)$ for some variable $?X$, if $P_t(?X)$ is assembled in the following way: for every $t'$ in $T$, if $t'$ reflexively and transitively dominates $t$ in $T$, then $t'(?X,+)$ is in $P_t(?X)$; else if $t$ and $t'$ have no common subtype, then $t'(?X,-)$ is in $P_t(?X)$. To give an example, $P_{\textit{directed-motion}}(?X)$ for the type *directed-motion* in the type hierarchy of Fig. 4 would be the set $\{\textit{activity}(?X,-), \textit{motion}(?X,+), \textit{causation}(?X,-),$

[6] Thanks to Laura Kallmeyer and Yulia Zinova for pointing this out.

*motion-activity*$(?X, -)$, *directed-motion*$(?X, +)\}$.[7] It is easily seen that the size of some $P_t(?X)$ crucially depends on the position of $t$ in $T$, and on the size of $T$.

One basic problem of this approach is that so far XMG does not interpret the predicates of the `<sem>`-dimension, but merely accumulates them for later use. Hence XMG allows for the coexistence of predicates `activity(?X,-)` and `activity(?X,+)`, which should be ruled out when simulating types as constraints on unification. But even if XMG was enhanced to verify the functionality of predicates, at least three disadvantages would remain: (i) TSS have to be provided by the grammar writer, (ii) they have to be included in the XMG descriptions as a whole, and (iii) unifying sister types with a common subtype will yield a TSS that does not immediately reveal the type of the common subtype. The latter disadvantage might be more of an aesthetic kind, but the first and the second one clearly have an impact on usability. Modifying the type hierarchy in the context of a large grammar would make necessary a meta-Metagrammar, that would automatically recompute the TSS and adapt the parts of the XMG descriptions, where TSS were used. Therefore we present a novel `<frame>`-dimension in the next section, which is adjusted to the peculiarities of frame structures and frame composition.

## 6 A new `<frame>`-dimension

The description language employed in the `<frame>`-dimension of the extended XMG follows the one of the `<syn>`-dimension in many respects.[8] Basically, a `<frame>`-dimension contains descriptions of nodes and edges, where nodes can be assigned a variable (with `var`) and a type, and edges can carry a semantic label. The example in Fig. 5 illustrates this.[9] Note that the `var`-equations correspond to the boxed numbers found in the AVM notation of frames (see Fig. 1 and 2). But comparing `<frame>`-descriptions with `<syn>`-descriptions also reveals several crucial distinctions: neither do `<frame>`-descriptions employ non-strict dominance, nor do they refer to linear precedence, as the daughters of a mother are generally unordered. Furthermore the edges in the `<frame>`-dimension can carry a semantic label, other than those in the `<syn>`-dimension.

The most essential difference, however, is found in the solvers, since the solver of the `<frame>`-dimension takes into account the type hierarchy, which is specified globally within the header of the code example in Fig. 5. It also computes and inserts the highest common subtype. Apart from that the solving of the `<frame>`-descriptions is relatively cheap, since nodes and edges are completely specified, and therefore the solving only consists of finding the root and traverse the edges top-down. In contrast, the solvers of the `<syn>`-dimension (and of the `<sem>`-dimension) rest upon the unification of untyped partial descriptions,

[7] Of course $P_{directed\text{-}motion}(?X)$ could be further minimized, since *motion-activity*$(?X, -)$ already follows from *activity*$(?X, -)$.

[8] To be exact, we extended XMG2 (http://wikilligramme.loria.fr/doku.php?id=xmg2).

[9] A bracket notation is also available, similarly to the one in the `<syn>`-dimension.

```
class POConstr
...
hierarchy TYPE = {(event,activity),(event,motion),(event,causation),
          (activity,motion-activity),
          (motion,motion-activity),(motion,directed-motion)}
...
<frame> {
  node (type=causation,var=?X0) ?ROOT;
  node (var=?X1) ?ACTOR;
  node (var=?X2) ?THEME;
  node (var=?X3) ?GOAL;
  node (type=activity) ?CAUSE;
  node (var=?X4) ?EFFECT;
  node (var=?X1) ?CAUSEACTOR;
  node (var=?X2) ?EFFECTTHEME;

  edge (label=actor) ?ROOT ?ACTOR;
  edge (label=theme) ?ROOT ?THEME;
  edge (label=goal) ?ROOT ?GOAL;
  edge (label=cause) ?ROOT ?CAUSE;
  edge (label=effect) ?ROOT ?EFFECT;
  edge (label=actor) ?CAUSE ?CAUSEACTOR;
  edge (label=theme) ?EFFECT ?EFFECTTHEME
}
...
```

**Fig. 5.** The `<frame>`-dimension of class POConstr.

which means they only take into account the values when unifying features or predicates.

The solvers of `<frame>` and `<syn>` do not differ, however, with respect to one crucial aspect: both resolve only tree structures. This might come as a surprise given that frames correspond to directed graphs [14], where nodes can be immediately dominated by more than one other node. Of course, in restricting itself to tree structures, the `<frame>`-dimension can model structures like the latter one only in an implicit way, namely by identifying separate nodes based on identical values in their respective `var`-property. Going back to the example in Fig. 5, the identification of nodes `?ACTOR` and `?CAUSEACTOR` is expressed through the `var`-value `?X1`. This account obviously borrows from the use of boxed-number variables in common AVM notation.

It remains to say that single connected frames with a unique root are resolved based on the descriptions within the `<frame>`-dimension. We do not see that solutions with more than one root could become necessary on the level of the lexicon.[10]

[10] Otherwise the solver of the `<frame>`-dimension might be modified analogously to the shift from TAG to multi-component TAG [13].

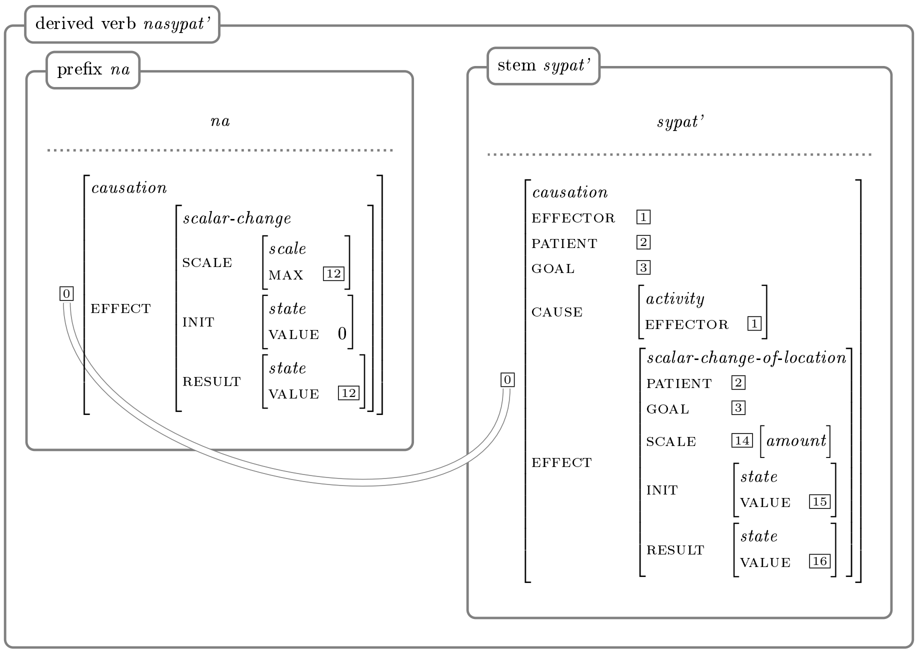


**Fig. 6.** Morphological decomposition of the Russian verb *nasypat'* ('to put') following [16]. The aspectual prefix *na* adds a perfective reading to the verbal stem *sypat'*. The perfective reading is encoded in the frame of *na* in terms of a specific scalar change.

## 7 An application to morphological decomposition

[16] not only present a metagrammatical factorization of elementary tree templates and their frame-based semantics, but they also briefly show that the same idea can be applied below the word level, namely to verbal prefixation in Russian: just as tree templates can be factorized (or decomposed) into tree fragments, complex morphological units can be decomposed into morphemes and their specific frame-semantic contribution. As an example, [16] decompose the perfective verb *nasypat'* ('to put') along the lines of Fig. 6. Again, we won't go into the linguistic details of the proposed analysis, but rather try to answer the question how this could be implemented by means of XMG.

Until recently, the morphological level of linguistic descriptions had not been attracting much attention within the framework of XMG. In general, morphology was (and still is) seen to lie outside its main focus, and that the word lexicon should be rather treated using other means. To our knowledge, the first work to deviate from this general picture is [4] in treating the system of agglutinative affixes in Ikota (a Bantu language). For this, [4] presents an extension of XMG which lets the grammar writer define linearly ordered "fields" for each type of affix. The specification and field assignment of affixes then takes place in a new dimension, called `<morph>`.

```
class nasypat
declare ?M1 ?M2 ?S1 ?S2
{
  ?M1 = na[];
  ?M2 = sypat[];
  ?S1 = ?M1.?S;
  ?S2 = ?M2.?S;
  <morph>{
        ?S1 >> ?S2
  }
}
```

```
class na                         class sypat
export ?S                        export ?S
declare ?S                       declare ?S
{ <morph>{                       { <morph>{
        morpheme ?S;                     morpheme ?S;
        ?S <- "na"                       ?S <- "sypat'"
  }                                }
}                                }
```

**Fig. 7.** The `<morph>`-dimension of the classes corresponding to the morphological decomposition of *nasypat'* in Fig. 6.

Considering the case of morphological decomposition shown in Fig. 6, it would be possible to adopt the fields-and-`<morph>` approach of [4] in a straightforward way: first two fields, say F1 and F2, would be globally specified, and then *na* would be assigned to F1 and *sypat'* to F2 within their respective `morph`-dimension. Given, however, that prefixation in Russian verbs is more flexible, allowing for, e. g., the stacking of several aspectual prefixes, we have chosen a more general approach which underlies the implementation shown in Fig. 7.[11] Instead of specifying a fixed number and order of fields, the linear order of *na* and *sypat'* is constrained locally inside the instantiating class `nasypat` using the common LP-operator ('`>>`'). Note that the operator `<-` assigns a surface string to a `morpheme` object. No matter what approach to morphology is chosen, the respective `<frame>`-dimension remains the same, along the lines of what has been presented in the last section. It is therefore omitted in Fig. 7.

The solver for the `<morph>`-dimension is rather simple compared to the one of `<syn>`, since the order of morphemes is constrained by immediate precedence only and the accumulated descriptions are supposed to be complete, meaning that no precedence between morphemes has to be inferred. After accumulating the `morpheme` objects and the precedence rules between them, the solver therefore just searches for the first morpheme (with no morpheme preceding it), and then

[11] A more flexible `<morph>`-dimension could be also advantageous in other cases, such as nominal compounds in German.

follows the line(s) of precedence rules. Finally it checks that no morpheme is left behind.

Of course the presented `<morph>`-dimension and its solver are very preliminary and designed specifically for the kind of analysis in Fig. 6. It needs to be clarified in subsequent research whether this is also applicable on a larger scale.

## 8 Conclusion

In this paper, we presented ongoing efforts to extend the grammar engineering framework XMG in order to deal with typed feature structures, respectively frames. We showed that the simulation of frames within the `<sem>`-dimension is doable, however there are disadvantages concerning the implementation of type hierarchies. Therefore a novel `<frame>`-dimension was developed which is adjusted to the peculiarities of frame structure and frame composition, and which should eventually reduce the burden for the grammar writer. We then showed that the `<frame>`-dimension can not only be combined with trees and tree fragments, but it can also be useful for the implementation of recent frame-based accounts to morphological decomposition, thereby considerably widening the scope of XMG.

The presented extensions to XMG are fully operational in a recent prototype. We further plan to make available a bracket notation for `<frame>`-descriptions that is closer to the common AVM notation, and to also include the `<frame>`-dimension in the XMG viewer.

## References

1. Abeillé, A., Rambow, O.: Tree Adjoining Grammar: An overview. pp. 1–68 (2000)
2. Bos, J.: Predicate logic unplugged. In: Dekker, P., Stokhof, M. (eds.) Proceedings of the Tenth Amsterdam Colloquium. pp. 133–143. Amsterdam, Netherlands (1996)
3. Crabbé, B., Duchier, D., Gardent, C., Le Roux, J., Parmentier, Y.: XMG : eXtensible MetaGrammar. Computational Linguistics 39(3), 1–66 (2013), http://hal.archives-ouvertes.fr/hal-00768224/en/
4. Duchier, D., Ekoukou, B.M., Parmentier, Y., Petitjean, S., Schang, E.: Describing morphologically rich languages using metagrammars: a look at verbs in Ikota. In: Workshop on Language Technology for Normalisation of Less-Resourced Languages (SALTMIL 8 - AfLaT 2012). pp. 55–59 (2012), http://www.tshwanedje.com/publications/SaLTMiL8-AfLaT2012.pdfpage=67
5. Fillmore, C.J.: Frame semantics. In: The Linguistic Society of Korea (ed.) Linguistics in the Morning Calm, pp. 111–137. Hanshin Publishing (1982)
6. Goldberg, A.: Constructions at Work. The Nature of Generalizations in Language. Oxford Univ. Press, Oxford (2006)
7. Joshi, A.K., Schabes, Y.: Tree-Adjoining Grammars. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 3, pp. 69–124. Springer, Berlin, New York (1997)
8. Kallmeyer, L., Lichte, T., Maier, W., Parmentier, Y., Dellert, J., Evang, K.: Tulipa: Towards a multi-formalism parsing environment for grammar engineering. In: Coling 2008: Proceedings of the workshop on Grammar Engineering Across Frameworks. pp. 1–8. Manchester, England (August 2008)

9. Kallmeyer, L., Osswald, R.: An analysis of directed motion expressions with Lexicalized Tree Adjoining Grammars and frame semantics. In: Ong, L., de Queiroz, R. (eds.) Proceedings of WoLLIC. pp. 34–55. No. 7456 in Lecture Notes in Computer Science LNCS, Springer (September 2012)
10. Kallmeyer, L., Osswald, R.: A frame-based semantics of the dative alternation in Lexicalized Tree Adjoining Grammars. In: Piñón, C. (ed.) Empirical Issues in Syntax and Semantics 9. pp. 167–184 (2012), iSSN 1769-7158
11. Kallmeyer, L., Osswald, R.: Syntax-driven semantic frame composition in Lexicalized Tree Adjoining Grammar (2013), unpublished manuscript
12. Kay, P.: An informal sketch of a formal architecture for Construction Grammar. Grammars 5, 1–19 (2002)
13. Parmentier, Y., Kallmeyer, L., Lichte, T., Maier, W.: XMG: eXtending Meta-Grammars to MCTAG. In: Actes de l'atelier sur les formalismes syntaxiques de haut niveau, Conférence sur le Traitement Automatique des Langues Naturelles, TALN 2007. Toulouse, France (June 2007)
14. Petersen, W.: Representation of concepts as frames. The Baltic International Yearbook of Cognition, Logic and Communication 2, 151–170 (2007)
15. Xia, F., Palmer, M., Vijay-Shanker, K.: Developing tree-adjoining grammars with lexical descriptions. In: Bangalore, S., Joshi, A.K. (eds.) Using Complex Lexical Descriptions in Natural Language Processing, pp. 73–110. MIT Press, Cambridge (2010)
16. Zinova, Y., Kallmeyer, L.: A frame-based semantics of locative alternation in LTAG. In: Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11). pp. 28–36. Paris, France (September 2012), http://www.aclweb.org/anthology-new/W/W12/W12-4604