The AAAI 2017 Spring Symposium on
Computational Construction Grammar and Natural Language Understanding
Technical Report SS-17-02

Tree-Adjoining Grammar: A Tree-Based Constructionist Grammar Framework for Natural Language Understanding

Timm Lichte and **Laura Kallmeyer**
CRC 991, University of Düsseldorf, Germany

Abstract

In this paper, we propose to use Tree-Adjoining Grammar (TAG) as a means to formalize central tenets of Construction Grammar (CxG). We will show that TAG, beyond its lexicalized elementary trees, provides several levels of grammatical abstraction that correspond to constructions. In other words, within the specification of a TAG one can find specifications of non-lexicalized syntactic tree fragments with their attached meaning. This allows to capture many of the insights from CxG in an explicit way. Moreover, TAG is a well-established framework in mathematical and computational linguistics. These properties make TAG a highly relevant contender for hosting constructionist implementations of natural language understanding.

1 Introduction

Tree-Adjoining Grammar (TAG, Joshi, Levy, and Takahashi 1975; Joshi and Schabes 1997; Abeillé and Rambow 2000) is one of the major grammar formalisms (Müller 2016) with a rich history that dates back to the early 1970's. Originally, it was developed by engineers and further studied by theoretical computer scientists and computational linguists. Its use for linguistic modeling started in the 1980's (see, for instance, Kroch 1987; Kroch 1989; Kroch and Joshi 1987) and lead to a large amount of work on linguistic analyses within TAG and in particular to large implemented grammars for several languages (e.g., for English in XTAG Research Group 2001). In this context, parsers (Schabes and Joshi 1988; Bangalore and Joshi 1999), implementation tools (Candito 1996; Crabbé and Duchier 2005; Crabbé et al. 2013) and grammar induction tools (Xia 2001) have been developed.

So far, CxG was not really in the focus of the TAG community, and vice versa,[1] which is surprising given the rather obvious connections. This paper tries to remedy this mutual ignorance by directly relating the two grammatical frameworks. We will present TAG as a grammar formalism that shares central ideas with (some versions of) CxG that have been highlighted by Goldberg (2013): namely the only use of *surface structure*, that is, by virtue of dismissing a transformational component, the possibility to specify *grammatical constructions* and to order them in a hierarchical way, yielding *a network of constructions* "which nodes are related by inheritance links" (Goldberg 2013).

The paper is structured accordingly: we will first introduce the relevant aspects of TAG and its syntax-semantics interface. We will then furnish evidence for the compatibility of TAG with Goldberg's tenets of constructionist approaches. Following this, we will briefly go into implementation tools and NLP applications based on TAG and finally compare TAG to already approved constructionist approaches.

2 Tree-Adjoining Grammar (TAG)

TAG is a tree-rewriting system where a grammar consists of a set of elementary trees that are combined by means of two tree rewriting operations: substitution and adjunction. A simple example is provided in Figure 1 showing the derivation of the sentence *John always walked*. Replacing the NP leaf node in the *walked* tree with the *John* tree is called substitution. In contrast, replacing some non-leaf node such as the VP node of *walked* with the *always* tree is called adjunction. An elementary tree can be either substituted or adjoined, but not both: adjoining trees (also known as *auxiliary trees*) such as the one of *always* include one special leaf node decorated with an asterisk, the *footnode*, which sub-

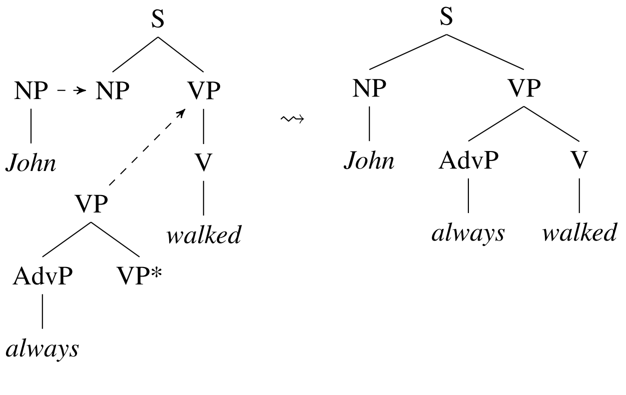

Figure 1: TAG derivation with intransitive *walked*

[1]For example, van Trijp (2013) does not mention TAG in his detailed landscape of grammar formalisms.

stituting trees (also called the *initial trees*) lack. The footnote is necessary to control where the subtree dominated by the rewritten target node ends up after adjunction. The label of the target node and the label of the root node of the attaching tree have to match, and so do the labels of root and footnode in auxiliary trees. Furthermore, corresponding to string-rewriting systems such as CFG, a complete, valid derivation yields a derived tree that only has leaf nodes with terminal labels, hence words. This also holds for the derived tree shown on the right side of Figure 1.

TAG as such does not further restrict the structure of elementary trees – except for the fact that a terminal node label such as *John* may only appear on a leaf node and may not be the target of substitution or adjunction. Yet there are certain conventions when modeling linguistic objects: (i) elementary trees are lexicalized in the sense that every elementary tree includes at least one terminal node label, also called the *lexical anchor*; (ii) an elementary tree encodes the valency frame of the lexical anchor, if it has one. Commonly, arguments of the lexical anchor are represented as non-terminal leaf nodes. Hence, in the intransitive case such as in Figure 1, the tree of *walked* only bears the non-terminal NP leaf node of its subject. In contrast, adverbials, such as *always* in Figure 1, and other recursively attachable constituents are represented in separate auxiliary trees (also knows as *factoring of recursion*) and thus are adjoined into the verbal tree.

In order to model other valency frames of the lexical head, further elementary trees must be used. For example, the elementary tree of *walked* with a directional PP is shown in Figure 2. Moreover, different linearizations of one and the same valency frame are accounted for by specific elementary trees as well, for example, when the directional PP or the subject NP appears in a topicalized position. An elementary tree of this sort will be later shown in Figure 5.

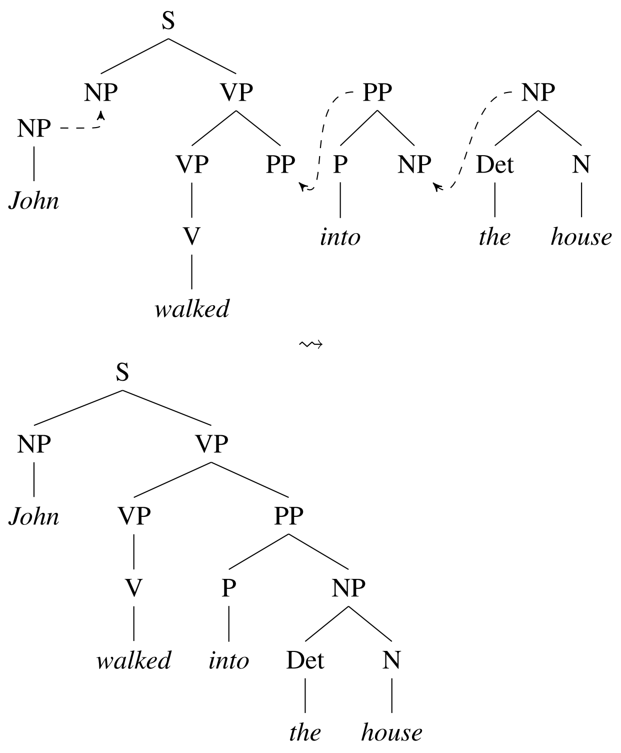

Figure 2: TAG derivation with directional *walked*

A verb such as *walked* eventually anchors a set of elementary trees that are derivationally unrelated. This implies that TAG as such lacks widespread notions of derivational asymmetry: neither is the passive derived from the active, nor does extraction start out from some "base" linearizations. TAG doesn't know movement, even though empty words may appear as leaf nodes (see Figure 5). On top of this, there is no predefined order in which arguments are saturated, other than in grammar formalisms that rely on valency lists such as HPSG. Summing up: even though, at its formal basis, TAG is generative-enumerative, it has a strong declarative flavor to it due to its considerable derivational laissez fair and the general equality of elementary trees.

While the properties of TAG sketched so far represent the core understanding of the formalism, there is an abundant number of variants (and acronyms denoting them) that extend or restrict this core in all possible directions. For example, a popular family of variants, Multi-Component TAG (MCTAG), builds on sets of elementary trees rather than single elementary trees. Fortunately, the ideas brought forward in this paper are applicable to the popular variants at least, so we can safely ignore them here. Yet one of the extensions, which uses feature structures as node labels, will be briefly introduced in the next section when the syntax-semantics interface is detailed.

3 Interfacing TAG and frame semantics

Conceiving constructions as form-meaning pairs makes it necessary to also briefly explain the specific syntax-semantics interface that we will be assuming in the rest of the paper. Following Kallmeyer and Osswald (2013), we couple TAG elementary trees with frame semantic representations and model composition by unifications triggered by substitution and adjunction. The general mechanics of this approach can be already found in previous works on TAG-based meaning composition (e.g., Gardent and Kallmeyer 2003; Kallmeyer and Joshi 2003; Kallmeyer and Romero 2008), yet they adapt other semantic frameworks.

The frames in Kallmeyer and Osswald (2013) are formalized as multi-rooted typed feature structures with multiple base labels. In other words, some of the nodes are labeled with base labels $\boxed{0}, \boxed{1}, \ldots$, which give access to these nodes.[2] Furthermore, there is no explicit type hierarchy. Instead, nodes in the frames can have several types; dependencies between types such as subtype relations and type incompatibilities are formulated in constraints in the feature logic. In our AVM representations of frames, structure sharing is expressed using boxed letters $\boxed{v}, \boxed{w}, \ldots$, to be distinguished from the base labels that allow not only structure sharing but also direct access. As an example consider the directional elementary tree anchored by *walked*, paired with the frame with base label $\boxed{0}$ in Figure 3.

[2]Note that when using an elementary tree with its frame in a derivation, we always use a copy with fresh base labels.

In order to have syntactic composition trigger semantic unification, the elementary trees are enriched with interface features (I for "individual" and E for "event"). The values of these features are base labels in the semantic frame. If two of them get equated via syntactic unification, the corresponding frames unify in the semantic frame. In this way, syntactic trees and frames compose in parallel. The tree-frame pair for *walked* in Figure 3, for instance, specifies that the frame of the subject tree contributes the ACTOR via unification with the base label $\boxed{1}$ while the tree substituted into the PP node contributes the GOAL (base label $\boxed{2}$) and can contribute further specification to the event via unification with base label $\boxed{0}$. Figure 3 shows the derivation from Figure 2 including frames.

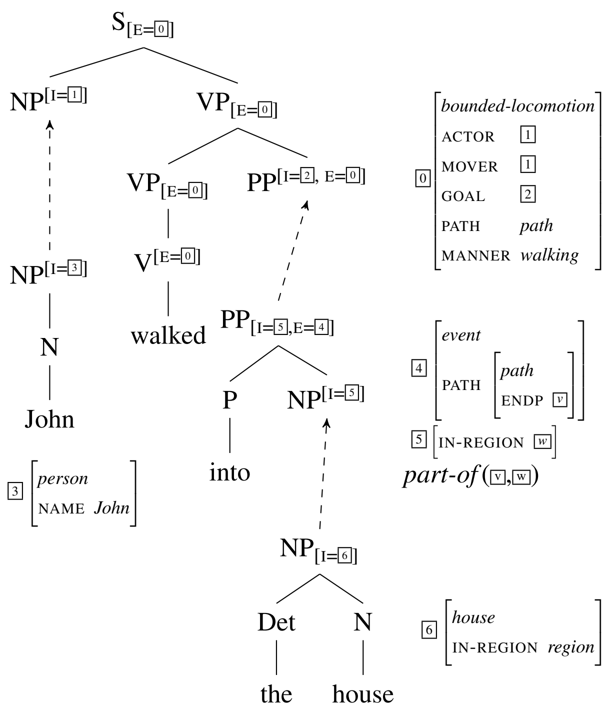

Figure 3: Derivation of Figure 2 with semantic frames

Note that the frame paired with *into* contains a relation between two frame nodes, *part-of*($\boxed{v}$, $\boxed{w}$) that is not functional, i.e., that is not one of the attributes usually allowed in feature structures. We allow for such relations on frame nodes (for details, see Kallmeyer and Osswald 2013).

The substitutions in Figure 3 lead to the base label unifications $\boxed{0} = \boxed{4}$, $\boxed{1} = \boxed{3}$, $\boxed{2} = \boxed{5} = \boxed{6}$. As a result, we obtain the derived tree frame pair in Figure 4.

4 TAG is a constructionist framework

After having introduced the general mechanics of TAG and the TAG-frame interface, this section will elaborate on the more specific properties of TAG that we think make it a constructionist grammar framework. For this, we will be guided by three of the "major tenets" of constructionist approaches that Goldberg (2013) defines: surface-orientation, grammat-

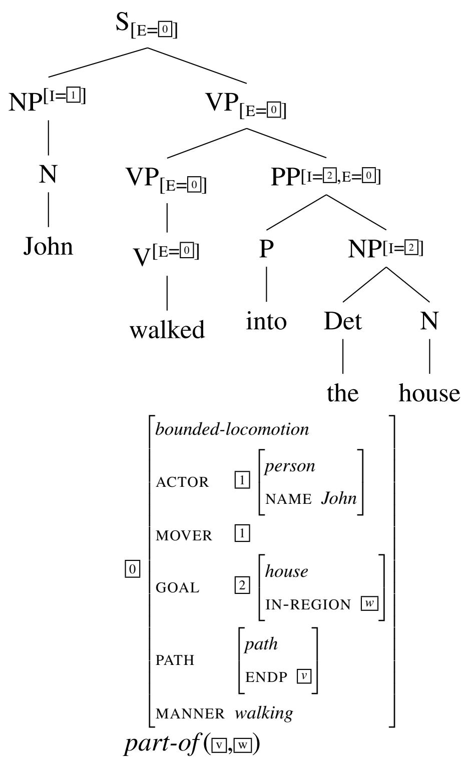

Figure 4: Resulting derived tree and frame for the derivation in Figure 3

ical constructions "at varying levels of complexity and abstraction" and the networking of constructions in terms of "inheritance links". In our view, the two remaining tenets ("crosslinguistic variability and generalization" and "usage-based") do not immediately concern the formal properties of TAG.[3]

4.1 TAG is surface-oriented

The term *surface-oriented* is borrowed from Sag and Wasow (2011). Goldberg (2013) instead uses the terms "surface structure" and "surface generalizations", but we reckon that they amount to the very same assumption: that there is no "deep" structure, acting as an autonomous and perhaps primary object of syntactic theory, from which the "surface" structure emerges. Instead, the surface structure is granted sole priority and focus. Of course, this assumption can be implemented in very different ways, also depending on the grammar formalism at hand. It seems to be consensus, though, that structure destroying operations known from Transformational Grammar are to be banned. Moreover, derivational dependencies between constructions (e.g., between active and passive) are seen critically.

Now, since TAG is a tree-generating formalism that crucially makes use of derivational operations, namely substitution and adjunction, it might seem at risk to be not as

[3]Crosslinguistic variation can actually be captured within the metagrammar introduced in Section 4.2 by distinguishing between universal and language-specific parts.

surface-oriented as required. However, this is not the case as a quick look at the standard analysis of long-distance extraction reveals, an instance of which is shown in (1):

(1) Who does Mary say walked into the house.

Here the *wh*-element *who* is separated from its governing verb (*walked*) by some intervening material that governs *walked*, respectively the whole *wh*-clause. In approaches that are not surface-oriented, cases like (1) are commonly treated with the *wh*-element being base generated in a position adjacent to *walked* and then being moved to the sentence initial position. The standard TAG analysis is different in that the extraction is *directly* generated by using a dedicated elementary tree for *walked*. It is shown in Figure 5 together with the auxiliary tree of *does say*, with which the intervening material is adjoined between *who* and *walked*. Note that this adjunction step could be recursively repeated

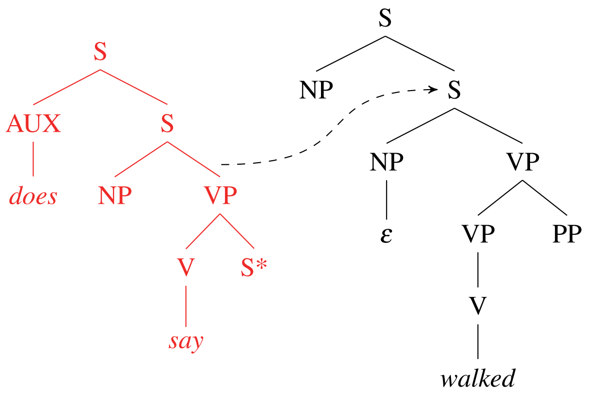

Figure 5: Part of the TAG derivation of a long-distance extraction such as in (1)

in order to make the extraction arbitrarily distant.

The elementary trees in Figure 5 highlight two other interesting facets of TAG that we want to touch briefly: firstly, elementary trees can comprise multiple anchors. This is particularly rewarding when dealing with multi-word expressions (Abeillé and Schabes 1996; Lichte and Kallmeyer 2016). Secondly, the elementary trees may include empty words as leaves. This is not to be confused with the "traces" of a movement operation; empty words in elementary trees are not essential for the derivation and oftentimes could be omitted. Empty words are mainly used to make the derived trees look like common phrase structure trees and, in some cases, to provide internal nodes as potential adjunction sites, for instance for stranding phenomena.

4.2 TAG bears constructions

Constructions are pairs of form and meaning that appear not only on the level of words, but also based on bigger syntactic units (phrasal constructions), or unlexicalized abstractions thereof (argument structure constructions). We have already learned in one of the last sections how meaning is coupled with elementary trees, which can therefore be seen as constructions on single words or phrasal units (see the remarks on multi-word expressions in the last section). What is left to be explained is how to define unlexicalized constructions with TAG, that is, how to gain the flexibility to capture constructions "at varying levels of complexity and abstraction" (Goldberg 2013).

The first step is rather straightforward: one can deanchor the elementary trees, which will yield *tree templates*, and then disentangle the meaning contributions of the anchor and the tree template. This is done for directional *walked* in Figure 6. The ◇ marks the place where the lexical anchor

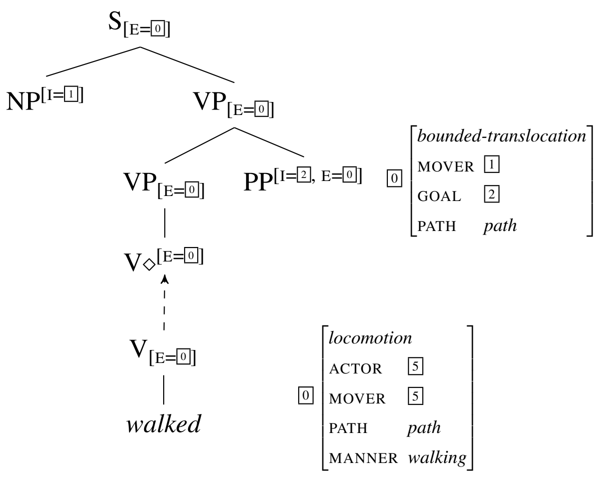

Figure 6: Lexical anchoring in TAG

has to be attached in the tree template. Of course, the meaning contribution of the tree template can be very general, but this can be nicely captured within frame semantics. A more specified template would be the one for caused motion shown in Figure 7. It should not be hard to see that these tree

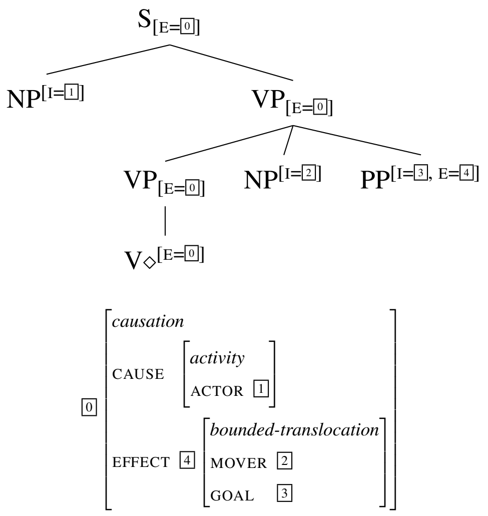

Figure 7: Caused-motion construction

templates together with a linked frame already give an accurate representations of specific linearizations of argument structure constructions.

But how can we further zoom in and describe the parts of tree templates and semantic frames? This is the domain of the *metagrammar* (MG, Candito 1996; Crabbé et al. 2013). Within the MG, sets of tree templates are described in a principled and factored way. This means that they are decomposed into tree fragments that capture linguistic generalizations and that can be paired with meaning fragments. This is the level where syntactic constructions and the meaning they contribute come in, either at the level of unanchored trees or at the level of MG tree fragments. A simple example for the decomposition of a tree template is given in Figure 8. On the right we have the unanchored tree for an intransitive verb. On the left we have its MG decomposition into a tree fragment for the subject and a tree fragment for the verbal spine. Note that the metagrammar uses descriptions and the description language for trees also contains terms for expressing non-immediate dominance and precedence. For example, the dotted edge in the subject fragment means that the VP node dominates the V node, but not necessarily immediately; furthermore, the tube edge connects nodes that get identified. Hence, while the fragments on the right side of Figure 8 are tree descriptions, the (minimal) model on the left side is a proper resolved tree.

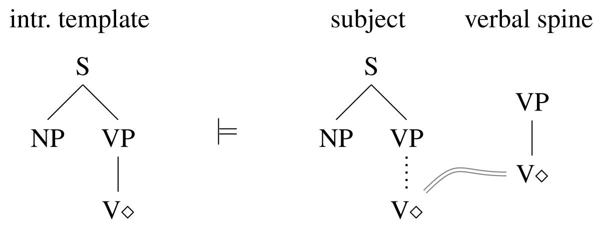

Figure 8: Metagrammar decomposition of the intransitive tree template

Speaking of descriptions, the metagrammar is not fixed for one description language – every linguistic dimension may come with a dedicated description language. The frame dimension, for example, would include descriptions that are more suitable for typed feature structures than tree descriptions (see below). However, what is generally available is disjunction, and this is crucial to describe sets of trees, also called *tree families*. Tree families are useful to abstract away from specific realizations, and they are therefore the actual counterpart of argument structure constructions. For example, the tree family representing the caused-motion construction would not only contain the tree template and the frame in Figure 7, but also all its variants in terms of linearization and diathesis.[4]

Another fundamental ingredient of the kind of metagrammar we are dealing with is the possibility to express *inheritance* (or rather subsumption) among bundles of descriptions. This is what allows one to implement the last of the three tenets of constructionist approaches.

[4]Note that the lexical anchor may select a subset of a tree family by means of feature unification during anchoring (see Figure 6).

4.3 TAG constructions form an inheritance network

One obvious drawback of surface-oriented approaches is that they tend to enumerate constructions: a construction is seen as an autonomous unit that does not emerge from other constructions during syntactic derivation. Leaving it as such, however, important linguistic generalizations will be impossible to capture, which is, of course, unacceptable when looking at the bushes of unordered constructions that one would have to assume.[5] One way to remedy this and to give the set of constructions more structure is the following: we can order constructions according to the information or properties they share. If construction A properly entails the properties of construction B, A is said to inherit from B. What we get is an inheritance network that is basically a partial order over the powerset of properties and hence over the set of constructions.

As for TAG metagrammars, the notion of inheritance is bound to the partial order on the powerset of descriptions – so it comes basically for free. An example for an inheritance hierarchy of tree fragments and tree templates is given in Figure 9. As usual, inheritance is encoded as dominance, hence the dominated nodes inherit from the dominating nodes. Therefore, the tree fragments, from which all the tree templates (directly or indirectly) inherit, make up the first row. Tree templates such as “transitive” can directly inherit from both a fragment and a template. Of course, the example is very rudimentary and skips all the frame semantic descriptions. But the general availability of inheritance networks in TAG metagrammars should be gotten clear enough.

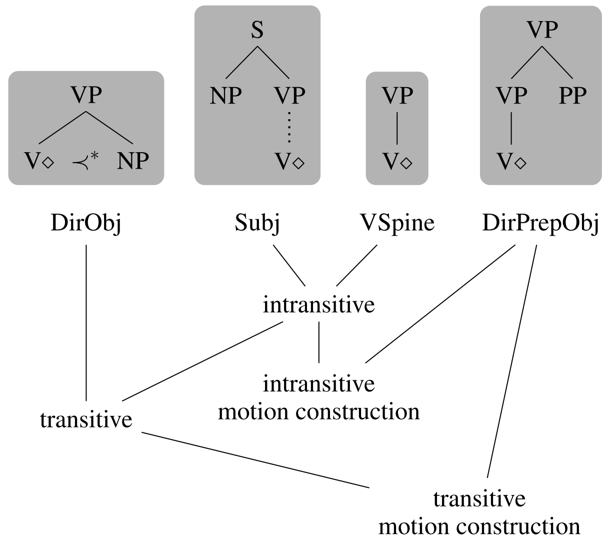

Figure 9: Inheritance hierarchy of tree fragments and tree templates as part of a TAG metagrammar

[5]In fact, transformations were originally introduced for the very same technical reason, namely to factorize a sprawling CFG (see Chomsky 1965).

It should be kept in mind, though, that the nodes of the inheritance hierarchy in Figure 9 are bundles of tree descriptions. Before they can be used within a TAG, they have to be resolved, i.e., their minimal models have to be computed. Once they get resolved, they correspond to tree (template) families, and when parsing a sentence, one of the contained tree templates will be lexicalized and finally combined with other lexicalized (hence elementary) trees through substitution and adjunction.

5 Implementation tools and NLP applications

When introducing the underlying concepts of a TAG metagrammar, we already had an existing implementation tool in mind that integrates all these concepts: eXtensible MetaGrammar (XMG, Crabbé et al. 2013; Petitjean 2014). XMG provides description languages and dedicated compilers for generating a wide range of linguistic resources. Descriptions are organized into *classes*, alluding to the class concept in object-oriented programming. Similarly, classes have encapsulated name spaces and inheritance relations may hold between them.

The crucial elements of an XMG class are *dimensions*. 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. Among them are the dimensions **`<syn>`** for syntactic trees and **`<frame>`** for semantic frames, which are used in the code example in Figure 10.[6] We will not go into the details here, but note that the variables **?**X0 ... **?**X4 (which need to be declared first) can be shared across dimensions. This is crucial for implementing the syntax-semantics interface, that is, the links between a tree template and a semantic frame. What the code example in Figure 10 does not exemplify, for the sake of brevity, is how the caused-motion construction can inherit from more general constructions as proposed in the inheritance hierarchy in Figure 9. For this, an **`import`** construct would be used next to **`declare`**.

```
class caused_motion_construction
declare ?X0 ?X1 ?X2 ?X3 ?X4
{<syn>{
  [cat=s,bot=[e=?X0]]{
    [cat=np,top=[i=?X1]](mark=subst){}
    [cat=vp,bot=[e=?X0]]{
      [cat=vp,bot=[e=?X0]]{
        [cat=v,top=[e=?X0]](mark=anchor)
            {} }
      [cat=np,top=[i=?X2]](mark=subst){}
      [cat=pp,top=[i=?X3,e=?X4]](mark=
          subst){} } }
 }
 <frame>{
  ?X0[causation,
      cause:[activity,
             actor:?X1],
      effect:?X4[bouned-translocation,
                 mover:?X2,
                 goal:?X3] ]
}}
```

Figure 10: XMG encoding of the caused-motion construction in Figure 7

[6]See Lichte and Petitjean (2015) for more details on the **`<frame>`** dimension.

The approach chosen by XMG is to collect and resolve descriptions monotonously: underspecified descriptions can be specified, but not removed. This is desirable because such deletions would add something of the power of transformations to the metagrammar. One approach that has this power is the *metarule* approach for TAG (Vijay-Shanker and Schabes 1992; Becker 1994), which first defines core tree templates in a way XMG does and then, using metarules, derives from them further tree templates. In doing so, metarules can also remove parts of core tree templates, for example, when deriving passive tree templates from active ones. This is impossible in XMG.

Using TAG, in combination with a metagrammar framework such as XMG, for formalizing and implementing CxG opens up new perspectives towards a computational use of CxG since TAG has been extensively used in natural language processing. Several grammar engineering and parsing frameworks have been developed for LTAG, including the original XTAG grammar and parser (XTAG Research Group 2001) (which is not maintained anymore). Parsers for TAG that are still used and maintained are for instance the DyALog system (Villemonte de la Clergerie 2005) and the TuLiPA system (Kallmeyer et al. 2010; Parmentier et al. 2008). TuLiPA can be used in combination with XMG. However, it does not process semantic frames yet; a corresponding extension is planned for the near future. The combined XMG-TuLiPA framework will then be a very useful implementation and parsing tool for constructionist analyses formulated in TAG.

Besides manual grammar development, TAG has also been used for grammar induction and probabilistic parsing, for instance for English (Chiang 2004), German (Kaeshammer and Demberg 2012) and Korean (Park 2006). There is a large amount of work on TAG parsing, including competitive results in data-driven parsing, in particular for constrained forms of TAG (see, e.g., Shen 2006; Carreras, Collins, and Koo 2008; Hayashi, Suzuki, and Nagata 2016). These techniques might also be put to use for CxG.

As a counterpart to parsing, TAG has also been successfully used in generation (Gardent and Kow 2005). Due to the transformation-free constraint-based grammar specification, the same grammar can actually be used for both directions (Kow, Parmentier, and Gardent 2006).

Moreover, TAG has also been shown to allow for psycholinguistically adequate incremental processing (PLTAG, Demberg, Keller, and Koller 2013).

6 Comparison to other constructionist frameworks

This is not the right place to give a detailed and comprehensive comparison of TAG and other constructionist approaches such as *Sign-based Construction Grammar* (SBCG, Boas and Sag 2012), *Embodied Construction Grammar* (ECG, Bergen and Chang 2005) and *Fluid Construction Grammar* (FCG, Steels 2011). Instead, we want to focus on one particular property that strikes us as central, namely the treatment of long-distance dependencies such as in (1). Basically, there exist two general strategies: (i) the *movement strategy*, including not only destructive transformations, but also bookkeeping methods such as the slash feature in HPSG (Pollard and Sag 1994, Ch. 4); and (ii) the *discontinuous constituent strategy*, which allows for capturing long-distance dependencies directly.[7] It has been noted by van Trijp (2013) that SBCG uses the movement strategy, while FCG implements the discontinuous constituent strategy. This view is shared by Müller (2016, Sec. 10.6; to appear) and he furthermore locates ECG next to SBCG (and HPSG). TAG, as shown above, can derive long-distance dependencies directly, and thus is similar to FCG, at least in this respect.

Having said this, it is still a matter of debate whether this distinction is actually relevant for the notion of a constructionist framework – we believe that it *is* relevant, because discontinuous constituents seem to make the framework more surface-oriented.

7 Summary

In this paper, we have argued that TAG is compatible with central tenets of CxG, and that interfacing TAG with frame semantics therefore makes it a powerful framework for modeling natural language understanding in a constructionist way. One important component is the metagrammar, which allows for flexible yet monotonic descriptions of various linguistic domains and their interactions below the level of elementary trees. Due to the long history of TAG, the framework is both mathematically well-understood and linguistically well-tested on a broad range of languages and phenomena. On the practical side, there is a wealth of existing tools for implementation, induction, parsing and generation. Furthermore, TAG has been shown to be a relevant framework for psycholinguistically realistic language modeling. On the whole, we therefore believe that the TAG perspective on constructions is valuable with respect to both, the sharpening of the formal notions and the development of language processing applications. Needless to say, the paper has shown that this fertilization also holds in the opposite direction.

Acknowledgments

The work presented in this paper was financed by the Deutsche Forschungsgemeinschaft (DFG) within the CRC 991 "The Structure of Representations in Language, Cognition, and Science." We would like to thank two anonymous reviewers for their valuable comments.

[7] Another strategy that is used with complex predicates and raising constructions is merging valency frames (Lichte 2015, Ch. 6).

References

Abeillé, A., and Rambow, O. 2000. Tree Adjoining Grammar: An Overview. In Abeillé, A., and Rambow, O., eds., *Tree Adjoining Grammars: Formalisms, Linguistic Analyses and Processing*. Stanford, CA: CSLI Publications. 1–68.

Abeillé, A., and Schabes, Y. 1996. Non-compositional discontinuous constituents in Tree Adjoining Grammar. In Bunt, H., and van Horck, A., eds., *Discontinuous Constituency*. Berlin, Germany: Mouton de Gruyter. 279–306.

Bangalore, S., and Joshi, A. K. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics* 25(2):237–265.

Becker, T. 1994. *HyTAG: A New Type of Tree Adjoining Grammars for Hybrid Syntactic Representations of Free Word Order Languages*. Ph.D. Dissertation, Universität des Saarlandes.

Bergen, B., and Chang, N. 2005. Embodied Construction Grammar in simulation-based language understanding. In Östman, J.-O., and Fried, M., eds., *Construction Grammars: Cognitive grounding and theoretical extensions*. Amsterdam: John Benjamins. 147–190.

Boas, H., and Sag, I., eds. 2012. *Sign-Based Construction Grammar*. Number 193 in Lecture Note. CSLI.

Candito, M.-H. 1996. A principle-based hierarchical representation of LTAGs. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING 96)*.

Carreras, X.; Collins, M.; and Koo, T. 2008. TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, CoNLL '08, 9–16.

Chiang, D. 2004. Statistical parsing with an automatically extracted Tree Adjoining Grammar. In Bod, R.; Scha, R.; and Sima'an, K., eds., *Data-Oriented Parsing*. CSLI Publications. 301–318.

Chomsky, N. 1965. *Aspects of the Theory of Syntax*. Cambridge, MA: The MIT Press.

Crabbé, B., and Duchier, D. 2005. Metagrammar redux. In Christiansen, H.; Skadhauge, P. R.; and Villadsen, J., eds., *Constraint Solving and Language Processing*, volume 3438 of *Lecture Notes in Computer Science*. Springer. 32–47.

Crabbé, B.; Duchier, D.; Gardent, C.; Le Roux, J.; and Parmentier, Y. 2013. XMG: eXtensible MetaGrammar. *Computational Linguistics* 39(3):591–629.

Demberg, V.; Keller, F.; and Koller, A. 2013. Incremental, predictive parsing with psycholinguistically motivated Tree-Adjoining Grammar. *Computational Linguistics* 39(4):1025–1066.

Gardent, C., and Kallmeyer, L. 2003. Semantic Construction in FTAG. In *Proceedings of EACL 2003*, 123–130.

Gardent, C., and Kow, E. 2005. Generating and selecting grammatical paraphrases. In *ENLG*.

Goldberg, A. 2013. Constructionist approaches. In Hoffmann, T., and Trousdale, G., eds., *The Oxford Handbook of Construction Grammar*. Oxford, UK: Oxford Univ. Press. 15–31.

Hayashi, K.; Suzuki, J.; and Nagata, M. 2016. Shift-reduce spinal TAG parsing with dynamic programming. *Transactions of the Japanese Society for Artificial Intelligence*.

Joshi, A. K., and Schabes, Y. 1997. Tree-Adjoining Grammars. In Rozenberg, G., and Salomaa, A., eds., *Handbook of Formal Languages. Vol. 3: Beyond Words*. Berlin: Springer. 69–123.

Joshi, A. K.; Levy, L. S.; and Takahashi, M. 1975. Tree Adjunct Grammars. *Journal of Computer and System Science* 10:136–163.

Kaeshammer, M., and Demberg, V. 2012. German and english treebanks and lexica for Tree-Adjoining Grammars. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2012)*.

Kallmeyer, L., and Joshi, A. K. 2003. Factoring Predicate Argument and Scope Semantics: Underspecified Semantics with LTAG. *Research on Language and Computation* 1(1–2):3–58.

Kallmeyer, L., and Osswald, R. 2013. Syntax-driven semantic frame composition in Lexicalized Tree Adjoining Grammar. *Journal of Language Modelling* 1:267–330.

Kallmeyer, L., and Romero, M. 2008. Scope and situation binding in LTAG using semantic unification. *Research on Language and Computation* 6(1):3–52.

Kallmeyer, L.; Maier, W.; Parmentier, Y.; and Dellert, J. 2010. TuLiPA - parsing extensions of TAG with Range Concatenation Grammars. *Bulletin of the Polish Academy of Sciences* 58(3):377–392.

Kow, E.; Parmentier, Y.; and Gardent, C. 2006. SemTAG, the LORIA toolbox for TAG-based parsing and generation. In *Proceedings of the 8th International Workshop on Tree Adjoining Grammar and Related Formalisms*, 115–120.

Kroch, A. S., and Joshi, A. K. 1987. Analyzing extraposition in a Tree Adjoining Grammar. In Huck, G. J., and Ojeda, A. E., eds., *Syntax and Semantics: Discontinuous Constituency*. Academic Press, Inc. 107–149.

Kroch, A. S. 1987. Unbounded dependencies and subjacency in a Tree Adjoining Grammar. In Manaster-Ramer, A., ed., *Mathematics of Language*. Amsterdam: John Benjamins. 143–172.

Kroch, A. S. 1989. Asymmetries in long-distance extraction in a Tree Adjoining Grammar. In Baltin, M. R., and Kroch, A. S., eds., *Alternative Conceptions of Phrase Structure*. Chicago, IL: University of Chicago Press. 66–98.

Lichte, T., and Kallmeyer, L. 2016. Same syntax, different semantics: A compositional approach to idiomaticity in multi-word expressions. In Piñón, C., ed., *Empirical Issues in Syntax and Semantics 11*, 111–140.

Lichte, T., and Petitjean, S. 2015. Implementing semantic frames as typed feature structures with XMG. *Journal of Language Modelling* 3(1):185–228.

Lichte, T. 2015. *Syntax und Valenz. Zur Modellierung kohärenter und elliptischer Strukturen mit Baumadjunktionsgrammatiken*. Number 1 in Empirically Oriented Theoretical Morphology and Syntax. Berlin: Language Science Press.

Müller, S. 2016. *Grammatical Theory: From Transformational Grammar to Constraint-Based Approaches*. Number 1 in Textbooks in Language Sciences. Berlin: Language Science Press.

Müller, S. to appear. HPSG, SBCG, and FCG: Commonalities and differences. *Constructions and Frames*.

Park, J. 2006. Extraction of Tree Adjoining Grammars from a treebank for Korean. In *Proceedings of the COLING/ACL 2006 Student Research Workshop*, 73–78. Sydney, Australia: Association for Computational Linguistics.

Parmentier, Y.; Kallmeyer, L.; Maier, W.; Lichte, T.; and Dellert, J. 2008. TuLiPA: A syntax-semantics parsing environment for mildly context-sensitive formalisms. In *Proceedings of TAG+9*, 121–128.

Petitjean, S. 2014. *Génération Modulaire de Grammaires Formelles*. Thése de doctorat, Université d'Orléans, Orléans, France.

Pollard, C., and Sag, I. A. 1994. *Head-Driven Phrase Structure Grammar*. Chicago and London: University of Chicago Press.

Sag, I. A., and Wasow, T. 2011. Performance-compatible competence grammar. In Borsley, R., and Börjars, K., eds., *Non-Transformational Syntax – Formal and Explicit Models of Grammar*. Oxford: Blackwell. 359–376.

Schabes, Y., and Joshi, A. K. 1988. An Earley-type parsing algorithm for Tree Adjoining Grammars. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics (ACL)*, 258–269.

Shen, L. 2006. *Statistical LTAG Parsing*. Ph.D. Dissertation, University of Pennsylvania.

Steels, L. 2011. A first encounter with Fluid Construction Grammar. In Steels, L., ed., *Design Patterns in Fluid Construction Grammar*. Amsterdam: John Benjamins. 31–68.

van Trijp, R. 2013. A comparison between Fluid Construction Grammar and Sign-Based Construction Grammar. *Constructions and Frames* 5:88–116.

Vijay-Shanker, K., and Schabes, Y. 1992. Structure sharing in Lexicalized Tree-Adjoining Grammars. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-92)*.

Villemonte de la Clergerie, E. 2005. DyALog: A tabular logic programming based environment for NLP. In *Proceedings of CSLP05*.

Xia, F. 2001. *Automatic grammar generation from two different perspectives*. Ph.D. Dissertation, University of Pennsylvania, Philadelphia, PA.

XTAG Research Group. 2001. A Lexicalized Tree Adjoining Grammar for English. Technical report, Institute for Research in Cognitive Science, University of Pennsylvania, Philadelphia, PA.