Synchronous Tree Unification Grammar

Timm Lichte
Collaborative Research Center 991
University of Düsseldorf
lichte@phil.hhu.de

Abstract

This paper presents a novel grammar formalism, Synchronous Tree Unification Grammar (STUG), that borrows ideas from two rather distinct exemplars of tree-based grammar formalisms, namely Synchronous Tree Adjoining Grammar and Tree Unification Grammar. At the same time STUG differs considerably from those in that it allows for a clean separation of syntax and valency. Exploiting this potential in the modelling of natural language grammar has a number of interesting consequences that we will sketch in the course of this paper.

1 Motivation

The underlying motivation for the development of Synchronous Tree Unification Grammar (STUG) is to model syntax and valency as separated, yet linked dimensions of natural language signs. This sharply contrasts with the lexical amalgamation of syntax and valency found within the TAG framework (and other main stream syntactic frameworks). Very generally speaking, we take *valency* to be a mapping from semantic roles to sets of morpho-syntactic properties and some marker for indicating necessity. Following common terminology, realizations of valency roles are also called *arguments*.

In TAG elementary trees, valency properties of the lexical anchor are commonly mapped bijectively onto non-terminal leaves due to well-formedness conditions (Abeillé, 1988; Frank, 1992; Abeillé and Rambow, 2000; Frank, 2002), while the realization of optional valency roles is reflected across the set of elementary trees with the same lexical anchor. However, the correspondence between elementary tree and valency frame is blurred by functional items such as complementizers, determiners and auxiliary verbs, which commonly anchor an elementary tree of their own.

This way of amalgamating elementary trees and valency information can be held responsible for a couple of difficulties that have shown up in various aspects of the TAG framework – amongst them the following:

Since elementary trees for predicative verbs enforce the surface realization of the verbal head and its obligatory arguments, TAG accounts have to cope with *elliptical structures* (i.e. with gapping) by means of more or less far reaching concessions: either tangled trees are generated from elementary trees using a non-trivial contraction operation (Sarkar and Joshi, 1996; Sarkar and Joshi, 1997), or one falls back on an infinitely ambiguous lexicon (Seddah, 2008; Seddah et al., 2010), or one includes empty words as a result of a deletion-like operation (Lichte and Kallmeyer, 2010) or as a result of lexical insertion using extra elementary trees (Sarkar, 1997; Seddah and Sagot, 2006).

Since alternative valency frames and alternative linearizations thereof multiply the set of (yet unanchored) tree templates (Prolo, 2002), the use of a metagrammar system in broad-coverage grammars is practically inevitable (XTAG Research Group, 2001; Duchier et al., 2004). Important syntactic generalisations are therefore not expressed directly in a TAG, but emerge indirectly across elementary trees. Furthermore the factorization by means of metagrammars makes the inclusion of empty words attractive, since they increase the reusability of tree fragments.

Finally, since elementary trees span over an ex-

Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11), pages 46–54,
Paris, September 2012.

tended domain of locality and may relate a lexical anchor to more than one preceding constituent, it is far from obvious how incremental parsing can be performed. Proposals so far either add unlexicalized trees ("prediction trees") and a verification operation (Demberg and Keller, 2008; Demberg, 2010), or place the lexical anchor at the left edge of an elementary tree, the head of which can be left underspecified (Mazzei et al., 2007).

Each of these difficulties may seem resolvable in one of the mentioned ways. But in my view the sum of necessary adaptations and concessions makes it worth thinking about accounts that relate syntax and valency more indirectly. STUG represents the first result of this line of thought.

2 The STUG formalism

2.1 The elementary structures

STUG and Synchronous Tree-Adjoining Grammar (STAG) (Shieber and Schabes, 1990; Shieber, 1994; Nesson and Shieber, 2008) share the idea, that syntactic and semantic representations are joined in the lexicon by making up a set of pairs of trees or multi-component structures. Furthermore there is some way of directly linking nodes of the syntactic domain and the semantic domain within an elementary pair.

To provide an example, a STUG pair for the verb *laughs* is shown in Figure 1. We call the first element of the STUG pair the *syntactic tree*, and the second element the *valency tree*. While the syntactic tree corresponds to an elementary and derived tree known from TAG, the valency tree resembles a TAG derivation tree in that it is unordered and may have edge labels, which here indicate semantic roles. The valency tree for *laughs* in Figure 1 mentions two argument roles, namely A(GENT) and P(ATIENT), that are specified along the feature structures in the respective nodes. Note that features can be polarized in sense of (Guillaume and Perrier, 2009): features that must be specified carry an exclamation mark, while the #-symbol is attached to "neutralized" features which may not unify with another neutralized feature. Links, finally, are represented by circled numbers, i e. ① . . . ⓝ.

Speaking more formally, a STUG consists of tuples $\langle \sigma, \{\phi_1, \ldots, \phi_n\}, \frown \rangle$ with σ being a syntactic tree, with a set of valency trees $\{\phi_1, \ldots, \phi_n\}$ and a linking relation $\frown$, for which

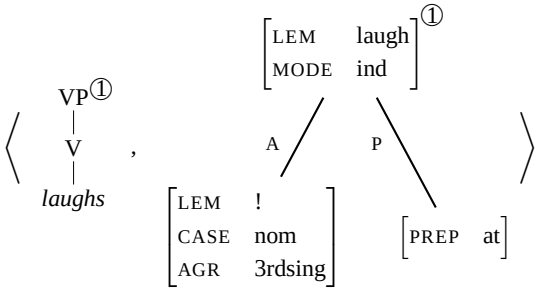


Figure 1: STUG pair for *laughs*.

the following holds:

- $\frown$: $P(V_\sigma) \rightarrow 2^{V_{\phi_1} \cup \ldots \cup V_{\phi_n}}$, where $P(V_\sigma)$ is the partition of the set of nodes from σ, and where $2^{V_{\phi_1} \cup \ldots \cup V_{\phi_n}}$ is the power set of the union of the sets of nodes from $\phi_1, \ldots, \phi_n$.
- For every valency tree $\phi_i \in \{\phi_1, \ldots, \phi_n\}$ with nodes V_i there exists at least one $\langle V_{syn}, V_{val} \rangle \in \frown$ with $V_i \cap V_{val} \neq \emptyset$.

In other words, (i) a link relates two sets of nodes, (ii) every syntactic node participates at most in one link and (iii) every valency tree must be linked with the syntactic tree. Note that we omit set braces around valency trees whenever a STUG pair includes only one valency tree, as is the case in Figure 1.

2.2 The combinatorial operations

While STAG uses substitution and adjunction in both domains, STUG combines syntactic trees by substitution and fusion, and valency trees by tree unification.

The *fusion operation* (Lichte, 2010) is a kind of tree unification where only single nodes unify, hereafter further limited to root nodes: When syntactic trees γ_i, γ_j with root nodes v_i, v_j are fused, the resulting syntactic tree γ' with root node v' only includes γ'_i and γ'_j, where γ'_i is γ_i with v_i replaced by v', and where γ'_j is γ_j with v_j replaced by v'. Furthermore every node from γ'_i linearly precedes every node from γ'_j. A sample derivation of *John sometimes laughs* using fusion and substitution is shown in Figure 2, which also demonstrates, that fusion allows the generated syntactic structures to be flat.

The order of fusion is controlled via finite state automata (FSA) that are assigned to nodes based on their syntactic label. A sample FSA for label VP is depicted in Figure 3. It restricts the linear

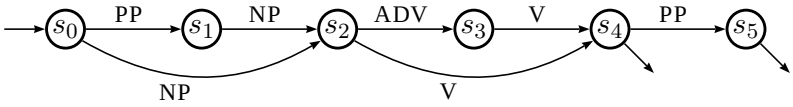


Figure 3: Sample finite state automaton for category VP.

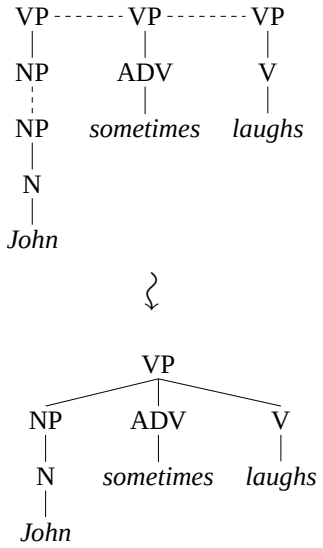


Figure 2: Derivation of *John sometimes laughs*. Dashed horizontal lines stand for fusion and dashed vertical lines stand for substitution.

order of daughter nodes in the following way: Let $l_V(v)$ be the label of v and let $fsa(l_V(v))$ be the FSA assigned to v. If $v_1 \ldots v_n$ is the sequence of daughter nodes of v in the final derived tree, the word $l_V(v_1) \ldots l_V(v_n)$ must be in $L(fsa(l_V(v)))$. To come back to our example, the derived tree in Figure 2 is licit, since the label sequence NP ADV V of the daughters of the VP node is in the language of the corresponding FSA shown in Figure 3.

In the process of fusion, links in the syntactic trees are collapsed in the following way: Let $\langle V_{syn_i}, V_{val_i} \rangle$ and $\langle V_{syn_j}, V_{val_j} \rangle$ be the links associated with nodes v_i, v_j that are replaced by v'. After fusion there is a new link $\langle V_{syn_i} \setminus v_i \cup V_{syn_j} \setminus \{v_j\} \cup \{v'\}, V_{val_i} \cup V_{val_j} \rangle$. Something similar applies during substitution.

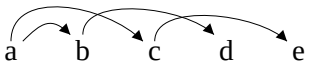
The combination of valency trees falls back upon the more general notion of *(tree) unification*, as is known, e. g., from Tree Unification Grammar (TUG) (Popowich, 1989; Gerdes, 2004). Unifying trees γ_1 and γ_2 to obtain tree γ_3 implies that the unifying and the resulting nodes form one isomorphic subtree in γ_1, γ_2 and γ_3 respectively. In order to narrow down the space of results, only tree unifications with a maximal number of unified nodes are considered. In STUG, the linking structure poses further constraints on tree unification: (i) unifying valency trees must be co-linked; (ii) if two unifying nodes carry links, they must be co-linked. When two nodes unify, their features structures unify, just like the label of unifying edges.

Finally, it needs to be specified, what happens to the links when unifying nodes of valency trees. Roughly, the links of the resulting node form the set union of the links of the unifying nodes. More precisely, if nodes v_i, v_j are replaced by v, every link $\langle V_{syn}, V_{val} \rangle$ with $v_i \in V_{val}$ or $v_j \in V_{val}$ is replaced by $\langle V_{syn}, V_{val} \setminus \{v_i, v_j\} \cup \{v\} \rangle$.

Coming back to the STUG derivation of the sentence *John sometimes laughs*, Figure 4 displays the required elementary STUG pairs. Note that the valency structure of *sometimes* specifies a T(ENSE)-role and that the unlexicalized STUG pair solely serves to embed NPs in a VP. The STUG derivation can be processed in two steps: first the syntactic tree is generated according to Figure 2, and after that the collected valency trees get unified into one. This two step approach is pursued in Figure 5.

3 Expressive power

STUG is powerful enough to account for ill-nested dependencies such as in (1):

(1) 

Assuming that the language contains only this string, the corresponding STUG in Figure 6 exploits the fact that all words differ and every word has a unique valency structure. Then in a flat structure, licensed by a simple FSA directly on the words, every node of the valency structure is linked to the S node, so that polarized features bring about the intended dependency relations and nothing more.

On the other side, STUG seems not capable of generating the counting language $\{a^n\ b^n | n > 0\}$ with crossed dependencies (and thus also

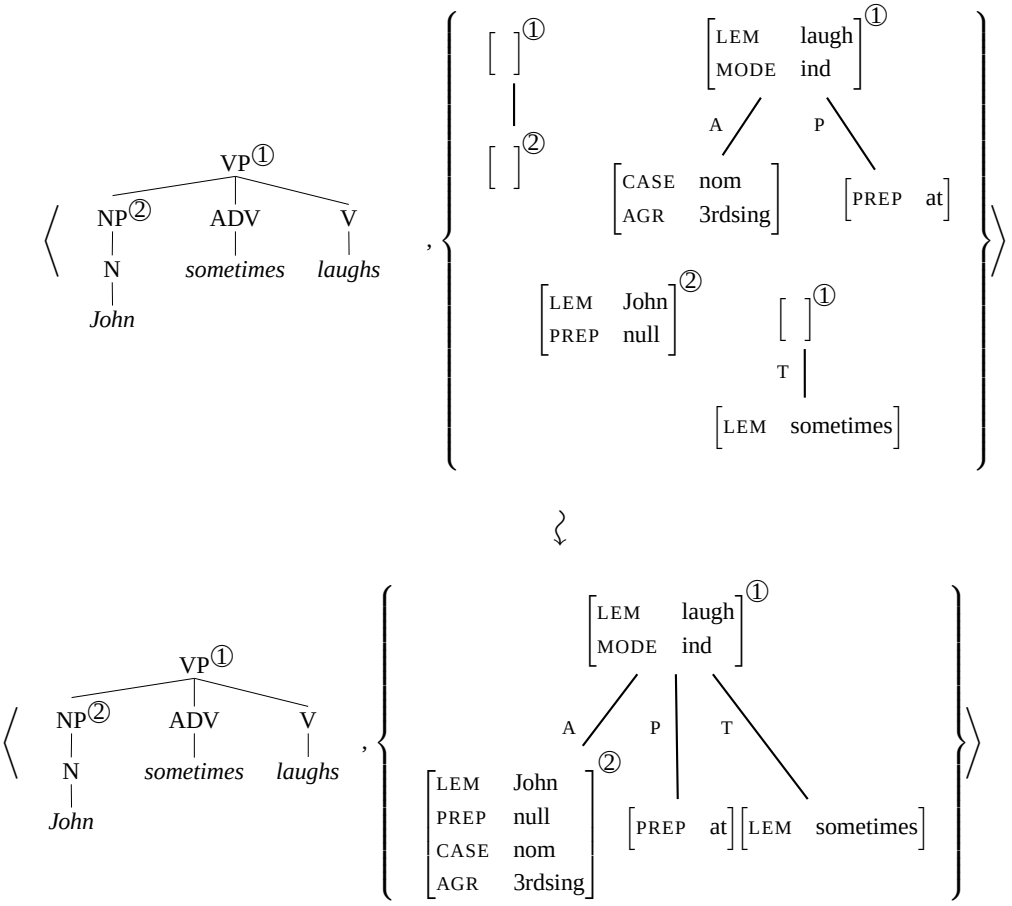


Figure 5: STUG derivation of *John sometimes laughs*.

not the copy language). Apparently linking is not selective enough to relate a^i only with b^i, while both can be in arbitrarily distant parts of the syntactic tree. Something similar also holds for the scrambling language $\mathsf{SCR}^{ind} = \{\sigma(NP_1, \ldots, NP_m)V_m \ldots V_1 | m \geq 1$ and σ is a permutation $\}$, where every $V_i + 1$ is supposed to govern V_i and which is beyond the expressive power of LCFRS (Becker et al., 1991). The counting language $\{a^n\ b^n | n > 0\}$ with nested dependencies is different is this respect, as can be seen from the STUG in Figure 7.

If we make use of neutralized polarity in features (indicated by #), which prevents unification of neutralized features and therefore helps to keep apart certain nodes in the valency tree, it is possible to generate the MIX-language, i.e. $\{w | w \in \{a, b, c\}^*, |a|_w = |b|_w = |c|_w\}$, as Figure 8 proves. Furthermore the grammar in Figure 8 can be easily adapted to derive the counting language $\{a^n\ b^n\ c^n\ d^n\ e^n | n > 0\}$ which also lies beyond the expressive power of TAG.

Hence, STUG seems to be both more and less powerful than TAG.

4 Formalism related questions

4.1 Valency structure = dependency structure?

The valency structure of a sentence and its dependency structure are not isomorphic. As Figure 9 shows, the contribution of functional elements such as the complementizer *that* and the passive auxiliary *is* can be diverse: *that* only contributes to the morpho-syntactic properties of the predicate node that is linked to the VP node in syntax; *is* furthermore specifies certain roles of the corresponding predicate. In both cases, however, functional elements do not get represented as nodes in the valency tree. This follows from the concept of valency as a mapping based solely on semantic roles. In contrast, the dependency structure would include also functional elements as single nodes, as it is a graph over words by definition.

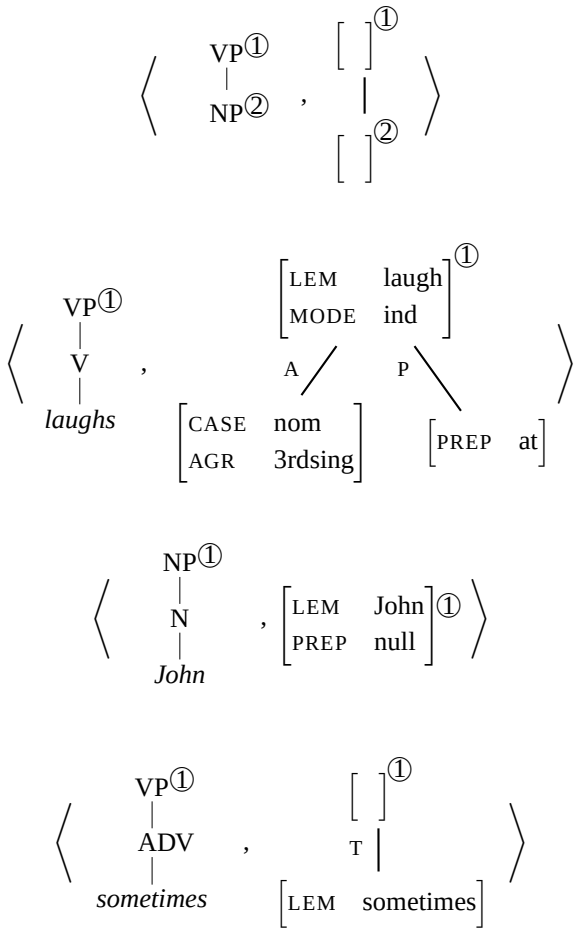


Figure 4: Elementary STUG pairs for *John sometimes laughs*.

One of the benefits in choosing valency structures might be that it circumvents the notoriously unclear (i.e. somewhat arbitrary) status of functional elements in dependency representations.

Note that derived valency structures do not only include semantic roles of arguments, but also semantic roles of adjuncts (see Figure 5). Consequently, they seem to have much in common with descriptions known from scenes-and-frames semantics (Fillmore, 1977a; Fillmore, 1977b).

4.2 Sister adjunction instead of fusion?

Both fusion and sister adjunction (Rambow et al., 1995; Chiang, 2003) support the generation of flat structures. However, they do not seem to be interchangeable in the context of STUG (see (Lichte, 2010) for a related discussion concerning TT-MCTAG). While fusion merges the root nodes of trees, sister adjunction rather draws a new edge between nodes. Choosing sister adjunction instead of fusion therefore considerably reduces the

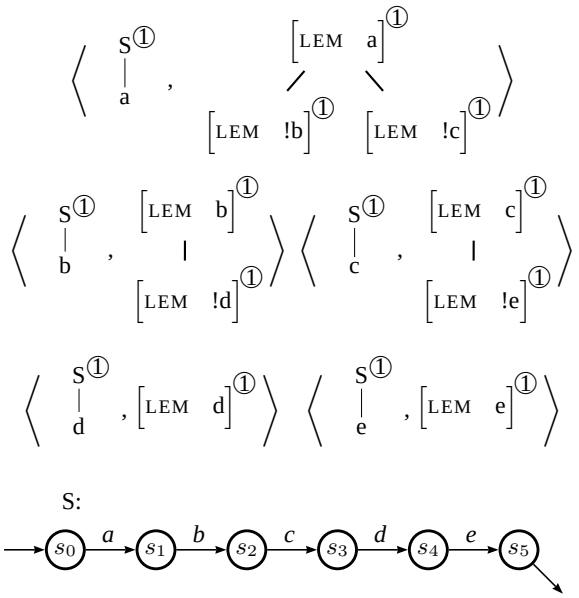


Figure 6: STUG for the ill-nested dependency structure in (1).

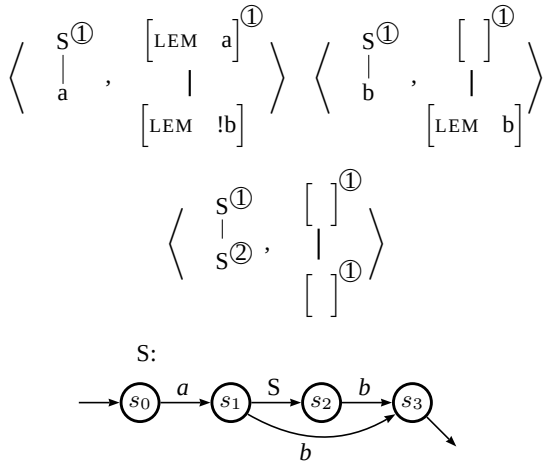


Figure 7: STUG for the string $a^n\ b^n$ with nested dependencies.

number of nodes, to which links can be attached in the lexicon. But since linking is a cornerstone of STUG, it is hard to see, how a STUG with sister adjunction for the mentioned cases would look like, let alone whether this would perform any better.

4.3 Feature structures instead of trees?

So far valency structures are represented as unordered trees whose edges carry role labels and stand for semantic relations, some of them functional in nature. It is thus debatable, whether one should choose a representation based on features structures instead, as they account for functional relations more straightforwardly. To give an ex-

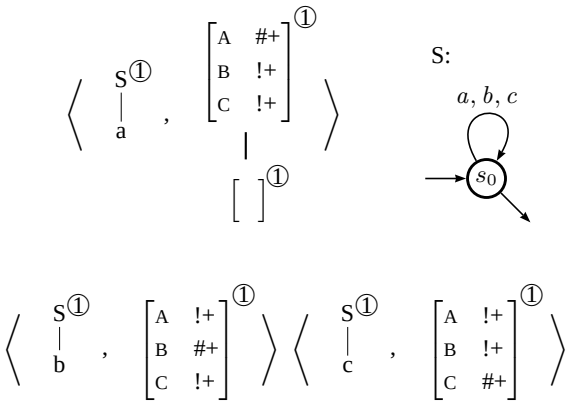


Figure 8: An STUG with neutralized features, marked up by exclamation marks, for deriving the MIX-language.

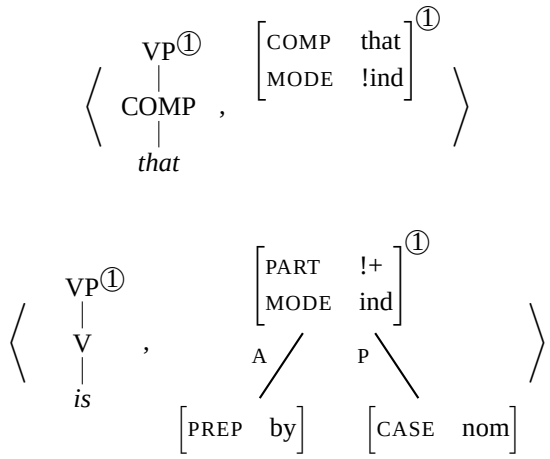


Figure 9: STUG pairs for complementizer *that* and passive auxiliary *is*. Exclamation marks indicate polarized features.

ample, the valency tree for *laughs* in Figure 1 could be replaced by the feature structure representation in Figure 10.

On the other side, a tree-based representation seems to pay off in cases where semantic roles are to be underspecified or where relations are non-functional, i.e. a single semantic role gets assigned to several constituents. The latter happens most prominently with temporal or locational roles.

5 Some consequences: ellipsis, grammar size and incrementality

Contrary to the amalgamation of syntax and valency in the TAG framework, STUG allows for a clear separation of syntax and valency, according to which syntax is only concerned with the

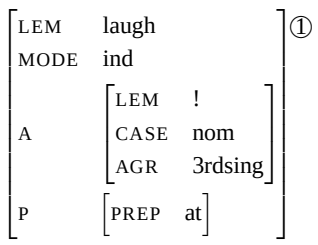


Figure 10: The valency tree of Figure 1 as feature structure.

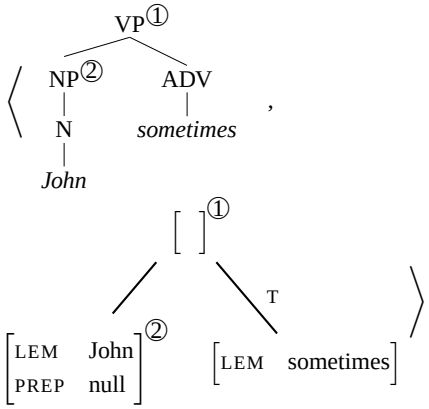


Figure 11: Derived STUG pair for *John sometimes*, which could be the fragmentary answer to the question *Who laughs?*, based on the elementary STUG pairs in Figure 4.

linearization of words (based on their syntactic category) and the determination of the local domain (based on the linking). Argument linking, however, completely rests on the unification of valency trees. This kind of separation has interesting consequences for the model of syntax. In the following we will discuss three of them involving ellipsis, grammar size and incrementality.

5.1 Ellipsis

Since syntactic trees can combine without there being any direct valency relation, STUG supports the base generation of ellipsis straightforwardly, as is shown in Figure 11 using the example of gapping. While the root node in the valency tree remains unspecified, the syntactic tree does not include any kind of empty placeholder for the missing verb. Furthermore neither extra rules nor extra lexical entries are involved in the process of derivation. Reconstruction, however, has to be guided by context, which can be thought of as set of more or less salient valency structures.

This analysis differs fundamentally from syntactic models that adhere to the amalgamation of

syntax and valency. They start out from complete syntactic representations of valency frames, which they adapt in cases of incompleteness. In this respect TAG accounts behave similar to accounts from GB, CCG or HPSG. Either two complete syntactic representations are "contracted" (Sarkar and Joshi, 1996; Sarkar and Joshi, 1997), or one complete syntactic representation is augmented with empty words as a result of deletion (Lichte and Kallmeyer, 2010) or insertion (Seddah and Sagot, 2006), or the lexicon includes incomplete syntactic representations as defective variants (Sarkar, 1997). None of these strategies is pursued in STUG.

In favour of contraction accounts one could argue, that contraction and reconstruction of ellipsis go hand in hand. Hence no extra mechanism for reconstruction is required. However, contraction is only applicable in cases where ellipsis and its antecedent can be located in the same sentence. It is therefore hardly applicable to fragementary answers or fragmentary corrections (Ginzburg and Sag, 2001; Schlangen, 2003; Ginzburg and Cooper, 2004), not to speak of discourse-inital fragments (Stainton, 1998; Stainton, 2006). For these cases a separate reconstruction mechanism based on the surrounding discourse is needed anyways. (Lichte and Kallmeyer, 2010) propose to relate reconstruction to the derivation tree of the antecedent sentence.

Summing up it can be said that the sketched STUG account to ellipsis looks promising, as no extra syntactic mechanism is used and reconstruction relates to valency structures, which seem at least as suitable as derivation trees.

5.2 Grammar size

For generating and maintaining a large-coverage TAG, the use of a metagrammar system is almost inevitable due to the size of the grammar. Regarding, for example, XTAG (XTAG Research Group, 2001), (Prolo, 2002) counts 97 tree templates for intransitive, transitive and ditransitive subcategorization frames. Taking all subcategorization frames (including e. g. those for idioms) into account, XTAG contains even 1008 verbal tree templates. This is due to the fact, that alternative argument realizations tend to be derived by means of different tree templates. In other words, the set of tree templates is some subset of the "Cartesian product" (Prolo, 2002) of subcategorization frames, alternative linearizations and active/passive alternations.

On top of that, optional arguments further increase lexical ambiguity by joining different tree families. For example, the finite verb *laughs* anchors intransitive tree families with and without PP argument, thus at least ten tree templates: two of the base configuration, three for extraction including preposition stranding, and five for relative clauses.

STUG helps to eliminate these two sources for large grammars and lexical ambiguity. Alternative linearizations can be represented in one FSA, i. e. outside elementary structures, while optional and obligatory arguments can be differentiated locally within a valency tree. Therefore the number of elementary structures reduces substantially compared to TAG. This already becomes apparent with regard to the lexical STUG pair in Figure 1, which, in combination with the simplistic FSA in Figure 3, suffices to cover four out of the ten mentioned tree templates for the finite verb *laughs*.

But it is not only the reduction of elementary structures that makes STUG attractive. Another source for grammar complexity lies in the rich feature structures with which nodes of TAG elementary trees may be equipped. XTAG defines around 50 features and uses, for example, no less than nine features to get the sequencing of determiners in NPs right. But also the verbal projection is equipped with an impressive number of features. Some of them help to constrain linearization (e. g. INV and COMP), while others pass on case or agreement restrictions, such as ASSIGN-CASE and AGR, or just display morphological properties of a phrasal head, auch as MODE.

By contrast, the snippets of STUG presented above have already shown, that no features get percolated around in the syntax tree of a STUG pair. Instead their main purpose is to specify nodes in the valency tree directly. Accordingly, mediating features like ASSIGN-CASE seem to be obsolete, as even raising verbs can directly access the raised subject in the valency structure. This is exemplified in Figure 12. Finally, the work of features that constrain linearization is now done elsewhere, namely in FSAs, making them obsolete as well. Hence, STUG seems to allow for a more precise, more transparent use of features, and it also seems to require a smaller number of features compared to TAG/XTAG.

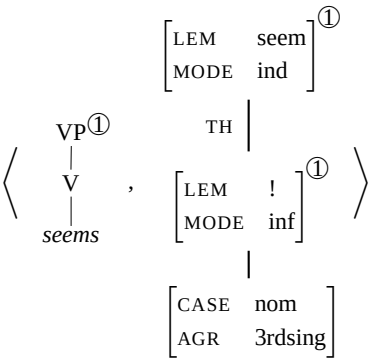


Figure 12: Lexical STUG pair for the raising verb *seems*.

5.3 Incrementality

Following (Sturt and Lombardo, 2005) incremental processing does not only imply that a sentence is parsed left-to-right on a word-by-word basis, but also that a connected syntactic representation is available for every prefix of the sentence. (Sturt and Lombardo, 2005) call this property *full connectedness*. Parsing with TAG, however, does not meet full connectedness out of the box, since, e. g., elementary trees of an argument and a modifier cannot be connected when preceding their governor. To fill this gap, two TAG variants have been proposed so far, namely Dynamic TAG (Mazzei, 2005; Mazzei et al., 2007) and PLTAG (Demberg and Keller, 2008; Demberg, 2010). Note that this is not at all a problem specific to TAG, but also comparable grammar formalisms such as CCG and Dependency Grammar are affected (Demberg, 2010), which similarly pursue an amalgamation of syntax and valency.

Contrary to TAG, parsing with STUG can be conducted in a fully connected manner even for sentences like *John sometimes laughs*, as the derivation in Figure 2 can be read off from the left to the right. This is mainly due to three factors: (i) syntactic trees of governors are spinal (i. e., the extended domain of locality is not found in the syntactic trees, but in the valency trees) and (ii) arguments and modifiers can always be connected without mediation of the governor. Finally, (iii) the derivation of the syntactic tree and the valency tree can be done asynchronously. It remains to be seen, how STUG compares to advanced psycholinguistic models such as PLTAG. This seems particularly interesting, for STUG does not use traces and, moreover, has a stronger affinity for free word order languages.

6 Conclusion

We presented a novel tree-based grammar formalism, Synchronous Tree Unification Grammar (STUG), which differs significantly from usual grammar formalisms such as TAG in that it allows for a separation of syntax and valency. After having described the functionality of STUG and having explored its expressive power, we briefly discussed some prospects concerning the modeling of ellipsis, the size and complexity of the grammar, and incremental, fully connected parsing. As encouraging as they may be, there is no doubt that many details are still unclear and need to be elaborated on in future work.

References

Anne Abeillé and Owen Rambow. 2000. Tree adjoining grammar: An overview. pages 1–68.

Anne Abeillé. 1988. A lexicalized tree adjoining grammar for French: The general framework. Technical Report MS-CIS-88-64, Department of Computer and Information Science , University of Pennsylvania.

Tilman Becker, Aravind K. Joshi, and Owen Rambow. 1991. Long-distance scrambling and tree adjoining grammars. In *Proceedings of EACL-91*.

David Chiang. 2003. Statistical parsing with an automatically extracted tree adjoining grammar. In Rens Bod, Remko Scha, and Khalil Sima'an, editors, *Data Oriented Parsing*, pages 299–316. CSLI Publications.

Vera Demberg and Frank Keller. 2008. A psycholinguistically motivated version of TAG. In *Proceedings of TAG+9*, pages 25–32, Tübingen.

Vera Demberg. 2010. *A Broad-Coverage Model of Prediction in Human Sentence Processing*. Ph.D. thesis, The University of Edinburgh.

Denys Duchier, Joseph Le Roux, and Yannick Parmentier. 2004. The Metagrammar Compiler: An NLP Application with a Multi-paradigm Architecture. In *Second International Mozart/Oz Conference (MOZ'2004)*.

Charles J. Fillmore. 1977a. The case for case reopened. In Peter Cole and Jerrold M. Sadock, editors, *Grammatical Relations*, volume 8 of *Syntax and Semantics*, pages 59–81. Academic Press, New York.

Charles J. Fillmore. 1977b. Scenes-and-frames semantics. In Antonio Zampolli, editor, *Linguistic Structures Processing*, volume 5, pages 55–81. North Holland, Amsterdam.

Robert Frank. 1992. *Syntactic Locality and Tree Adjoining Grammar: Grammatical, Acquisition and Processing Perspectives*. Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania.

Robert Frank. 2002. *Phrase Structure Composition and Syntactic Dependencies*. MIT Press, Cambridge,MA.

Kim Gerdes. 2004. Tree Unification Grammar. In Lawrence S. Moss and Richard T. Oehrle, editors, *Electronic Notes in Theoretical Computer Science*, volume 53. Elsevier. Proceedings of the joint meeting of the 6th Conference on Formal Grammar and the 7th Conference on Mathematics of Language.

Jonathan Ginzburg and Robin Cooper. 2004. Clarification, ellipsis, and the nature of contextual updates. *Linguistics and Philosophy*, 27(3):297–366.

Jonathan Ginzburg and Ivan A. Sag. 2001. *Interrogative Investigations: The Form, Meaning, and Use of English Interrogatives*. CSLI Publications, Stanford, CA.

Bruno Guillaume and Guy Perrier. 2009. Interaction Grammars. *Research on Language and Computation*, 7(2–4):171–208.

Timm Lichte and Laura Kallmeyer. 2010. Gapping through TAG derivation trees. In *Proceedings of TAG+10*, pages 93–100, New Haven, CT.

Timm Lichte. 2010. From partial VP fronting towards Spinal TT-MCTAG. In *Proceedings of TAG+10*, pages 85–92, New Haven, CT.

Alessandro Mazzei, Vincenzo Lombardo, and Patrick Sturt. 2007. Dynamic TAG and lexical dependencies. *Research on Language and Computation*, 5(3):309–332.

Alessandro Mazzei. 2005. *Formal and Empirical Issues of Applying Dynamics to Tree Adjoining Grammars*. Ph.D. thesis, University of Torino.

Rebecca Nesson and Stuart Shieber. 2008. Synchronous vector-TAG for natural language syntax and semantics. In *Proceedings of TAG+9*, Tübingen, Germany, 7–8 June.

Fred Popowich. 1989. Tree Unification Grammar. In *Proceedings of ACL-89*, pages 228–236, Vancouver, British Columbia, Canada, June.

Carlos A. Prolo. 2002. Generating the XTAG English grammar using metarules. In *Proceedings of COLING-02*, pages 814–820, Taipei. Taiwan.

Owen Rambow, K. Vijay-Shanker, and David Weir. 1995. D-tree grammars. In *Proceedings of ACL-95*, Cambridge, MA.

Anoop Sarkar and Aravind Joshi. 1996. Coordination in tree adjoining grammars: Formalization and implementation. In *Proceedings of COLING-96*, pages 610–615, Copenhagen, August 5-9.

Anoop Sarkar and Aravind Joshi. 1997. Handling coordination in a tree adjoining grammar. Longer version of paper in Proceedings of COLING 1996. Draft of August 19, 1997.

Anoop Sarkar. 1997. Seperating dependency from constituency in a tree rewriting system. In Tilman Becker and Hans-Ulrich Krieger, editors, *Proceedings of the Fifth Meeting on Mathematics of Language*, pages 153–160, Saarbrücken.

David Schlangen. 2003. *A Coherence-Based Approach to the Interpretation of Non-Sentential Utterances in Dialogue*. Ph.D. thesis, University of Edinburgh.

Djamé Seddah and Benoît Sagot. 2006. Modeling and analysis of elliptic coordination by dynamic exploitation of derivation forests in LTAG parsing. In *Proceedings of TAG+8*, pages 147–152, Syndey.

Djamé Seddah, Benoît Sagot, and Laurence Danlos. 2010. Control verb, argument cluster coordination and multi component TAG. In *Proceedings of TAG+10*, pages 101–109, New Haven, CT.

Djamé Seddah. 2008. The use of MCTAG to process elliptic coordination. In *Proceedings of TAG+9*, pages 81–88, Tübingen.

Stuart M. Shieber and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *Proceedings of COLING-90*, volume 3, pages 253–258, Helsinki, Finland.

Stuart M. Shieber. 1994. Restricting the weak-generative capacity of synchronous Tree-Adjoining Grammar. *Computational Intelligence*, 10(4):371–385.

Robert J. Stainton. 1998. Quantifier phrases, meaningfulness "in isolation", and ellipsis. *Linguistics and Philosophy*, 21:311–340.

Robert J. Stainton. 2006. *Words and Thoughts. Subsentences, Ellipsis, and the Philosophy of Language*. Oxford Univ. Press.

Patrick Sturt and Vicenzo Lombardo. 2005. Processing coordinated structures: Incrementality and connectedness. *Cognitive Science*, 29(2):291–305.

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