An Alternate View on Strong Lexicalization in TAG

Aniello De Santo[1] **Alëna Aksënova**[1] **Thomas Graf** [2]

Department of Linguistics
Stony Brook University
[1]{aniello.desanto,alena.aksenova}@stonybrook.edu
[2]mail@thomasgraf.net

Abstract

TAGs were recently shown not to be closed under strong lexicalization but to be strongly lexicalizable by context-free tree grammars of rank 2. This paper presents an alternative lexicalization procedure that builds on an earlier generalization of TAGs to multi-dimensional trees. A previous theorem that every Tree Substitution grammar is strongly lexicalized by a corresponding TAG is lifted to higher dimensions to show that for every d-dimensional TAG there exists a $(d+1)$-dimensional TAG that strongly lexicalizes it. A similar lifting reveals that d-dimensional TAGs are not closed under strong lexicalization, so for arbitrary TAGs an increase in dimensionality is an unavoidable consequence of strong lexicalization.

1 Introduction

The lexicalization properties of different grammar formalisms have been a topic of interest for a long time. A grammar is lexicalized if the atoms from which compound structures are assembled each contain a pronounced lexical item. In the case of TAGs, this means that no elementary trees may contain only non-terminal symbols or the empty string. Lexicalized grammars have the advantage of being *finitely ambiguous* — no string of finite length can have an infinite number of possible analyses. Not only does this guarantee that recognition is decidable, parsing is also simplified in practice (Schabes et al., 1988): the hypothesis space of the parser at any given point is significantly reduced because elementary trees that cannot be introduced by one of the symbols in the input string never need to be considered. As a result, many parsing algorithms assume that the grammar is lexicalized or at least can be lexicalized in an automatic fashion (cf. Kallmeyer (2010)).

In particular for parsing, though, the issue is not just whether a grammar can be lexicalized but whether the lexicalization procedure preserves essential properties of the grammar. While a recognizer only needs to determine the well-formedness of strings, a parser has to assign them structures licensed by the grammar. One thus has to distinguish between two types of lexicalization: *weak* lexicalization produces a weakly equivalent lexicalized grammar, whereas *strong lexicalization* yields a lexicalized grammars that also generates the same structural descriptions. Recent work showed that TAGs I) can be weakly lexicalized (Fujiyoshi, 2004), II) are not closed under strong lexicalization (Kuhlmann and Satta, 2012) and III) are strongly lexicalized by context-free tree grammars of rank 2 (Maletti and Engelfriet, 2012).

This paper offers a different perspective on strong lexicalization of TAGs that stays close to the basic intuition of TAGs as a mechanism for rewriting nodes by objects that are more complex than strings. The main advantage is that this allows us to preserve the basic insights of previous proofs on TAG lexicalization. We adopt Roger's formalism of multi-dimensional trees (Sec. 2.2 and 2.3), with TAGs as the special case where trees are limited to 3 dimensions (Rogers, 1998a; Rogers, 1998b; Rogers, 2003a; Rogers, 2003b). With TAGs lifted to arbitrary dimensions, we establish three central results:

1. Every d-dimensional TAG is a $(d+1)$-dimensional Tree Substitution Grammar (TSG; Sec. 3.1)

2. Every d-dimensional TSG is strongly lexicalized by some d-dimensional TAG (Sec. 3.2; cf. Schabes (1990)).

Proceedings of the 12th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+12), pages 93–102,
Düsseldorf, Germany, June 29 - July 1, 2016.

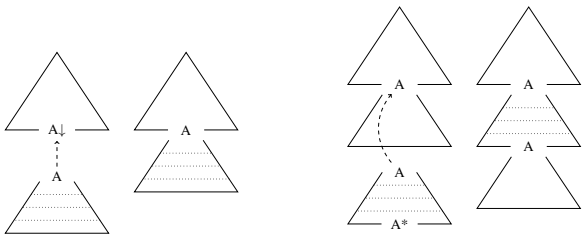

Figure 1: Substitution and adjunction

3. The class of d-dimensional TAGs is not closed under strong lexicalization (Sec. 3.3; cf. Kuhlmann and Satta (2012)).

This entails as a corollary that an increase in dimensionality is unavoidable in the strong lexicalization of (at least some) d-dimensional TAGs.

2 Preliminaries

In order to appreciate the results of Sec. 3.1–3.3, the reader has to be familiar with a few core concepts: the relation between adjunction and substitution (2.1), the view of TAGs as a formalism over 3-dimensional trees (2.2), and their generalization to trees of higher dimensionality (2.3).

2.1 Adjunction and Substitution

We assume familiarity on the reader's part with the TAG formalism as defined in Joshi (1985). A TAG is specified by two finite sets I and A of *initial* and *auxiliary* trees, respectively. Their union is the set E of *elementary* trees. Every auxiliary tree contains exactly one node that is marked as a foot node (indicated by * in our figures). Initial trees must not contain any foot nodes. Elementary trees are then combined by the operations of adjunction and substitution, illustrated in Fig. 1.

The distinction between adjunction and substitution plays an important role in this paper. Both operations replace nodes by trees. *Tree substitution* can only replace leaf nodes, and the tree being substituted may not contain a foot node. *Tree adjunction*, on the other hand, may rewrite any nonterminal node by any tree that contains exactly one foot node (i.e. by an auxiliary tree). Usually, it is also required that the label of the rewritten node is identical to the labels of the root (and the foot node, if it exists) of the substituted tree. Nodes are furthermore annotated with features to indicate whether adjunction and substitution are mandatory or optional as well as which trees may rewrite a given node.

Substitution can be regarded as adjunction of a foot-less tree at a leaf node. We thus use a more general definition of adjunction. Given a tree t, let $t \restriction r$ be the subtree of t rooted in node r. Furthermore, $t[n \leftarrow u]$ replaces a node n of t with tree u. If n does not exist, then $t[n \leftarrow u] = t$. Now adjunction of u into t at node n is defined as $t \overset{n}{\Leftarrow} u := t[n \leftarrow u[f \leftarrow t \restriction n]]$, where f is the foot node of u. Substitution is the special case where f does not exist so that $t[n \leftarrow u[f \leftarrow t \restriction n]] = t[n \leftarrow u]$. As long as nodes are correctly annotated with features to ensure that n is a leaf iff u does not contain a foot node and that labels are correctly matched, this generalized notion of adjunction behaves exactly as the combination of standard adjunction and substitution. We can now define a *tree substitution grammar* as a restricted TAG where all licit instances of adjunction only rewrite leaf nodes — this characterization will play an essential role in Sec. 3.1.

2.2 TAGs as 3-Dimensional Grammar Formalisms

The history of how a TAG G generates a specific tree can be recorded as a derivation tree. Each node in the derivation tree is labeled with the name of an elementary tree and an edge $\langle t, a, u \rangle$ from t to u with label a indicates $t \overset{a}{\Leftarrow} u$. The label of the root node must be the name of an initial tree. Now suppose that each node in the derivation tree is replaced by the tree it denotes, and each edge $\langle t, a, u \rangle$ is instead replaced by a collection of unlabeled edges that go from each node of u to the node in t at address a. The result can be regarded as a 3-dimensional tree such as the one in Fig. 2, with the first dimension corresponding to precedence, the second one to dominance, and the third one to adjunction.

Rogers (1998b) shows that TAGs are equivalent to context-free grammars over such 3-dimensional trees. Each elementary tree is no longer a standard 2-dimensional tree but instead has a 3-dimensional root node that is the mother of all nodes in the 2-dimensional tree. Just like a context-free grammar may combine 2-dimensional trees t and u if t contains a leaf with the same label as the root node of u, a TAG may combine 3-dimensional trees t and u if t contains a leaf in the third dimension whose label matches the 3-dimensional root of u. (Note that the feature annotations regulating adjunction are not considered part of node labels here.)

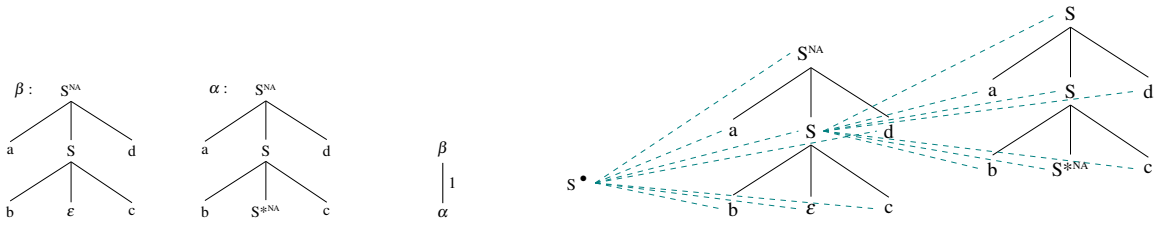

Figure 2: A standard TAG, one of its derivations, and the corresponding 3-dimensional tree

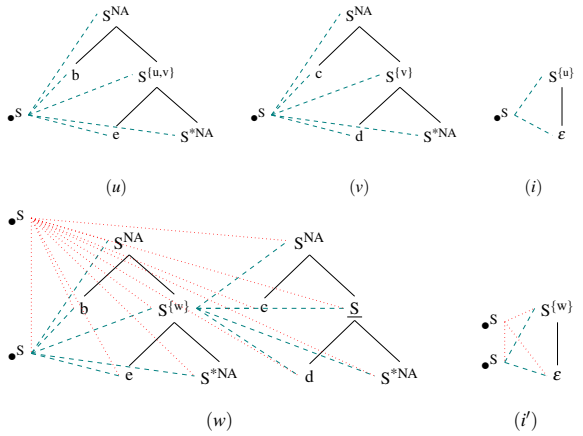

Figure 3: Example of 3- and 4-dimensional TAGs

2.3 Generalization to Multi-Dimensional Trees

Rogers (2003a; 2003b) explores how TAGs can be made more powerful by increasing the dimensionality of elementary trees. Let us illustrate the idea with a simple example first, which is also depicted in Fig. 3.

Consider a TAG with $I := \{i\}$ and $A := \{u, v\}$ such that the only valid adjunction steps are $i \overset{a_i}{\Leftarrow} u$, $u \overset{a_u}{\Leftarrow} u$, $u \overset{a_u}{\Leftarrow} v$, and $v \overset{a_v}{\Leftarrow} v$ for some fixed addresses a_i, a_u, a_v. In Fig. 3, the node at a_i is labeled $S^{\{u\}}$, the one at a_u has label $S^{\{u,v\}}$, and a_v refers to the node with label $S^{\{v\}}$.

This TAG allows derivations of the form iu^*v^*. It is impossible for any TAG to allow only derivations of the form iu^nv^n ($n \geq 0$) so that the number of u-trees matches the number of v-trees. However, a TAG with 4-dimensional trees has sufficient power to generate exactly those derivations.

First we lift i to the 4-dimensional tree i' by adding a single node in the fourth dimension that is the mother of all nodes in the 3-dimensional tree. Then we build a 4-dimensional auxiliary tree w that contains both u and v. Consider the complex 3-dimensional tree t obtained from u and v by identifying the 3-dimensional root of v with the node at address a_u in u. Just as was done for i we add a new root to t in the fourth dimension that is the 4-dimensional mother of all nodes in t, thus creating the 4-dimensional tree w. We also make the node at address a_v the 3-dimensional foot node of w (indicated by underlining) and add features so that w can optionally adjoin into another instance of w at address a_u. The result is a 4-dimensional TAG that only licenses derivations of the form iw^*, which produce standard TAG derivations of the form iu^nv^n, $n \geq 1$ (see Fig. 4). This example can be lifted to arbitrary dimensions to show that each new dimension adds more power to TAGs.

With the general intuition well-established, we now turn to the formal definition of higher-dimensional TAGs. A detailed axiomatization of multi-dimensional trees has already been provided by Rogers (2003a), so we limit the discussion to the bare essentials.

Let Σ be some fixed alphabet. For the sake of simplicity, we assume that Σ directly encodes adjunction constraints. A 2-dimensional tree is an ordered tree as usually defined, with nodes labeled by symbols drawn from Σ. The tree is *footed* iff it contains a foot node in the second dimension. A *d-dimensional local structure* l over Σ ($d \geq 3$) consists of a Σ-labeled *d-root* r and a $(d-1)$-dimensional tree t over Σ such that r immediately dominates every node n of t along the d-th dimension. The notion of $(d-1)$-dimensional tree will be clarified in the next paragraph. We also call $t := \mathrm{yd}^{d-1}(l)$ the *$(d-1)$-yield* of the local structure, and we say that r is a *d-dimensional mother* of a $(d-1)$-dimensional tree. Lower-dimensional yields are obtained by iterated application of the yield operator: $\mathrm{yd}^n(l) = \mathrm{yd}^n(\mathrm{yd}^{n+1}(\cdots \mathrm{yd}^d(l)))$, $2 \leq n < d$. We sometimes omit the dimension of the yield if it is clear from context.

The set of d-dimensional trees ($d \geq 3$) is defined in a recursive fashion. First, every d-dimensional local structure is a d-dimensional tree. Then t is a d-dimensional tree iff i) its root r and all

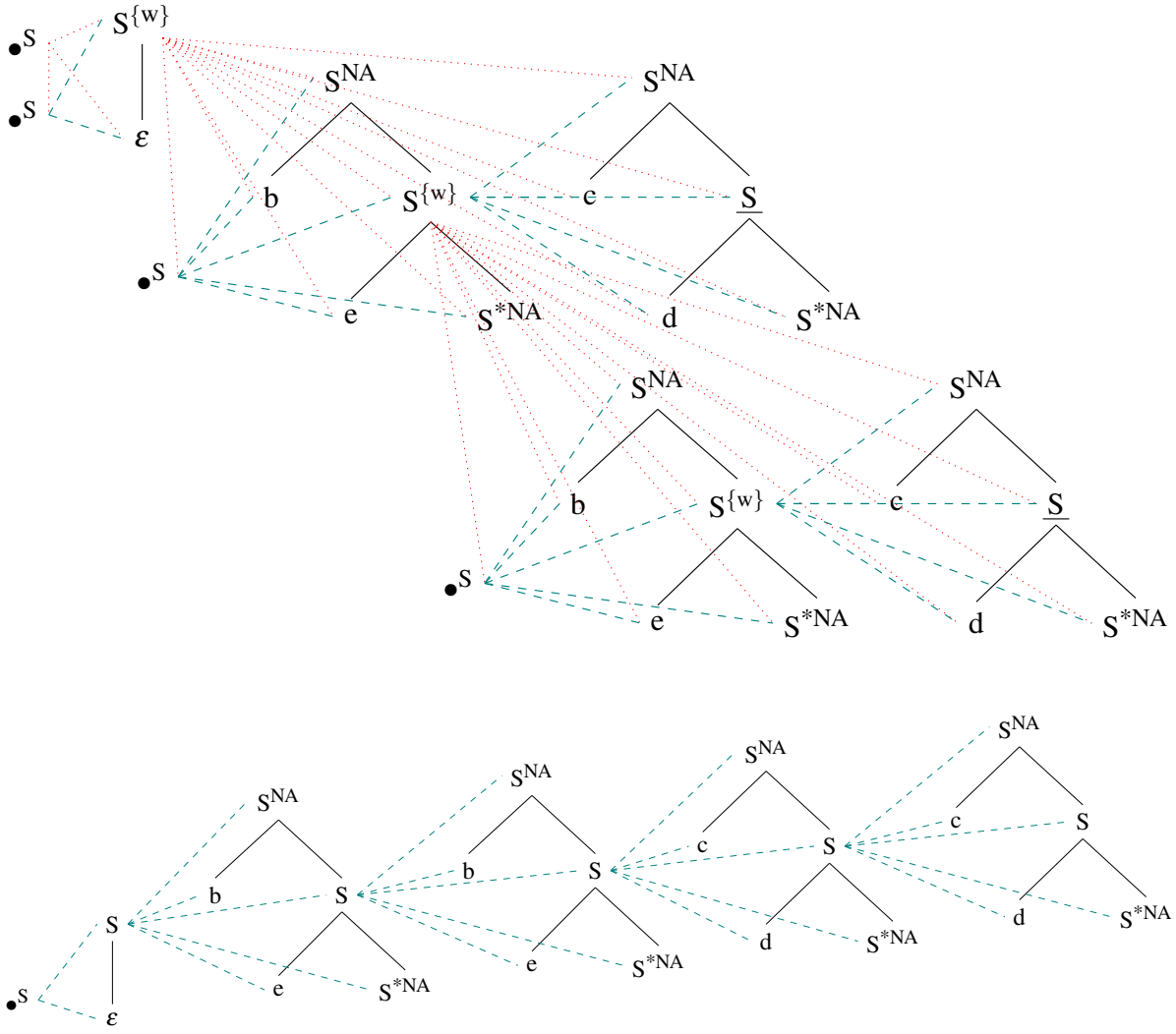

Figure 4: A complex 4-dimensional tree and the 3-dimensional tree obtained from it

the nodes immediately dominated by r along the d-th dimension jointly form a d-dimensional local structure, and II) every node that is immediately dominated by r in the d-th dimension and that is itself a d-dimensional mother must be the root of a d-dimensional tree. A d-dimensional tree is *footed* iff one of its nodes is marked as a $(d-1)$-dimensional foot node. Note that no d-dimensional tree may contain more than one foot node for dimension $(d-1)$. Starting out with the standard definition of 2-dimensional trees it is thus possible to construct 3-dimensional local structures, which can be combined into 3-dimensional trees, from which one can build 4-dimensional local structures, and so on. The examples in Fig. 2 and 4 highlight that d-dimensional trees are indeed trees in the sense that every point is reachable from the d-dimensional root by exactly one path along dimension d.

Whenever the $(d-1)$-dimensional yield of some local d-dimensional structure contains a node that is a mother along the d-th dimension of some $(d-1)$-dimensional tree, this encodes an instance of d-dimensional adjunction (in the generalized sense of Sec. 2.1 with substitution as a special case of adjunction). Consider a d-dimensional tree in which some node n is a d-dimensional mother of a $(d-1)$-dimensional tree u. Then the output of this d-dimensional adjunction is computed as follows. First, n becomes the $(d-1)$-dimensional mother of the $(d-2)$-dimensional v tree whose mother is the root of u. Note that if some nodes of v are $(d-1)$-dimensional mothers, this relation is preserved. Second, if n is already a $(d-1)$-dimensional mother of some $(d-2)$-dimensional tree w and u contains a $(d-1)$-dimensional foot node f, then f becomes the new mother of w. In all other cases, adjunction is undefined. The reader may consult Fig. 4 for a concrete example of 4-dimensional adjunction.

We are now in a position to fully define d-dimensional TAGs. An *elementary* d-dimensional tree (d-tree) is a d-dimensional local structure. It is an *initial* d-tree if it is not footed, and an *auxiliary* d-tree otherwise. Given two elementary d-dimensional trees u and v, we write $u \overset{d\ a}{\Longleftarrow} v$ to indicate that v may adjoin into u at node address a. This is tantamount to stating that the grammar allows for d-dimensional trees where the node at address a in $\mathrm{yd}^{d-1}(u)$ is the d-dimensional mother of $\mathrm{yd}^{d-1}(v)$. A *d-dimensional TAG* (d-TAG) G^d

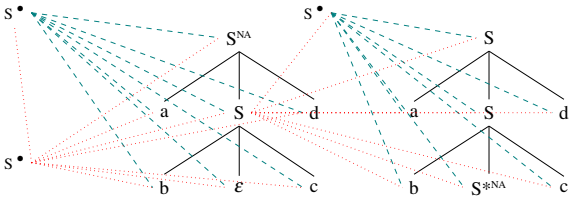

Figure 5: The 3-dimensional adjunction from Fig. 2 is replaced by 4-dimensional substitution

consists of a finite set of *elementary d-trees*. A *d-dimensional TSG* (d-TSG) is the special case of a d-TAG where it holds for every tree that a node is a d-dimensional mother iff it is not a $(d-1)$-dimensional mother.

3 Strong Lexicalization of d-TAGs

3.1 d-TAGs are $(d+1)$-TSGs

The advantage of d-TAGs is that proofs for standard TAGs can be generalized to arbitrary dimension with little modification. The proof that TSGs are strongly lexicalized by TAGs will be suitably modified in the next section, thereby establishing that d-TSGs are strongly lexicalized by d-TAGs. This directly implies that $(d+1)$-TAGs strongly lexicalize d-TAGs thanks to a basic fact we show now: every d-TAG is a $(d+1)$-TSG.

We first observe that a given d-TAG G^d can be easily converted to a $(d+1)$-TAG G^{d+1}. Let e be some elementary d-tree of G^d. We construct an equivalent $(d+1)$-tree e' such that $\mathrm{yd}^d(e') = e$. To this end, let e' be a $(d+1)$-dimensional local structure with root r and yield e such that the label of r is identical to the root label of e. Each node in $\mathrm{yd}^{d-1}(e')$ may (or must) be adjoined to in dimension $d+1$ iff the corresponding node in $\mathrm{yd}^{d-1}(e)$ may (or must) be adjoined to in dimension d.

An example of the construction was already shown in Fig. 3: the 3-dimensional tree i is converted into a 4-dimensional structure i' by adding a new root node S and edges linking the new root to the nodes in i. Since there is a direct link between the 4-dimensional root node and each node in the 3-dimensional tree, every node that was part of i is now a leaf node in i'.

The conversion of interior nodes to leaves is the essential aspect of the construction: all nodes of e' that can be adjoined to are d-dimensional leaves. This holds because e is a d-dimensional local structure, so the only node of e that is not a leaf in dimension d is its root r, which cannot be

adjoined to. Clearly every node of e that can be a mother in dimension $(d+1)$ must be a node that can be adjoined to, and consequently it is necessarily a leaf in dimension d. By definition, then, G^{d+1} is a $(d+1)$-TSG.

A concrete example is given in Fig. 5, where the 3-dimensional trees α and β from Fig. 2 have been lifted to 4-dimensional structures via the addition of 4-dimensional root nodes. As a result, the node of 3-dimensional β that α was adjoined to in Fig. 5 is now a leaf node of 4-dimensional β, and the instance of adjunction in the third dimension is replaced by substitution in the fourth dimension.

Proposition 1. For every d-TAG G^d there exists a $(d+1)$-TSG G^{d+1} such that the d-tree language generated by G is the d-yield of the $(d+1)$-tree language generated by G^{d+1}.

3.2 d-TAGs Strongly Lexicalize d-TSGs

It has been known for a long time that TAGs strongly lexicalize TSGs (Schabes, 1990; Joshi and Schabes, 1997; Kuhlmann and Satta, 2012). In this section we show that the corresponding proof can be lifted to multidimensional structures.

Proposition 2. For each finitely ambiguous d-dimensional TSG that does not generate the empty string and contains only useful trees, there is a strongly equivalent d-dimensional Lexicalized TAG.

Let us first consider the general idea behind the construction from Schabes (1990) for normal TSGs, i.e. 3-TAGs where adjunction is only allowed at leaf nodes. Given such a 3-TSG G, we can build a lexicalized 3-TAG G_l that generates the same tree language as G. The basic idea is that G can be bifurcated into a recursive part and a non-recursive part. The recursive part contains all elementary trees u such that there is at least one 3-dimensional tree generated by G in which a node of u dominates another instance of u in the third dimension. The non-recursive part consists of all other elementary 3-trees, which form the set I_l of

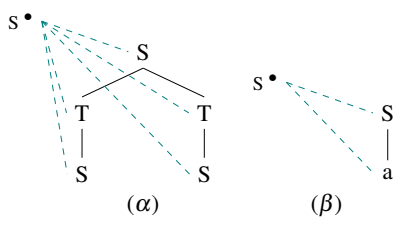

Figure 6: Non-lexicalized 3-dimensional TSG

initial trees of the lexicalized 3-TAG G_l. The set A_l of auxiliary trees of G_l is then created by taking the recursive part of G and computing its closure under 3-dimensional adjunction of trees from I_l (which is only allowed to target leaf nodes since no $t \in I_l$ contains a foot node). Both I_l and A_l are guaranteed to be finite, so G_l is indeed a 3-TAG. Schabes proves, in our generalized terminology, that G and G_l generate 3-dimensional tree languages with the same 2-dimensional yield.

We now show that Schabes' lexicalization algorithm can be directly lifted to work on d-TAGs such that every d-TSG is strongly lexicalized by some d-TAG. As an illustrative example, we refer to the 3-TSG depicted in Fig. 6 and the equivalent, lexicalized 3-TAG in Fig. 7.

We start out with some well-known concepts from graph theory that are needed in the construction of the graph from which the auxiliary trees are computed. Given a Σ-labeled graph $\mathscr{G} := \langle N, B\rangle$ with set N of nodes and set $B \subseteq N \times \Sigma \times N$ of branches, a *path* on $\mathscr{G}$ is a sequence $\langle n_1, \ldots, n_k\rangle \in N^k$ such that for all $1 \leq i < k$, $\langle n_i, \sigma, n_{i+1}\rangle \in B$ for some $\sigma \in \Sigma$ (we use the term branches instead of the more common "edges" to avoid confusion between E as the set of edges and E as the set of elementary trees of some TAG). If furthermore $n_1 = n_k$, then the path is a *cycle*. A subgraph of $\mathscr{G}$ is a graph $\mathscr{H} = (N', B')$ such that $N' \subseteq N$, $B' \subseteq B$, and $\langle n_1, \sigma, n_2\rangle \in B'$ implies $n_1, n_2 \in N'$. The *graph composition* (also, *lexicographic product* (Harary, 1972)) $\mathscr{G}_1 \cdot \mathscr{G}_2$ of graphs $\mathscr{G}_1 := \langle N_1, B_1\rangle$ and $\mathscr{G}_2 := \langle N_2, B_2\rangle$ is a graph such that: I) the node set of $\mathscr{G}_1 \cdot \mathscr{G}_2$ is the cartesian product $N_1 \times N_2$, and II) any two nodes (u, v) and (x, y) are connected by a σ-labeled edge in $\mathscr{G}_1 \cdot \mathscr{G}_2$ iff either $\langle u, \sigma, x\rangle \in B_1$ or $u = x$ and $\langle u, \sigma, y\rangle \in B_2$.

Now let us consider a d-dimensional finitely ambiguous d-TSG G^d with set I of initial trees such that the string yield of the language generated by G^d does not include the empty string. We also assume that for every initial tree $i \in I$ there is at least one well-formed derivation that contains i. Recall that a d-TSG does not have any auxiliary trees by definition, wherefore all elementary trees of G^d are useful initial trees. A lexicalized d-TAG G^d_{lex} is then built from the d-TSG G^d in four steps.

Step 1: Determine Recursion. In order to distinguish between recursive and non-recursive trees, we build a directed graph $\mathscr{G} = (N, B)$ where N is the set of nodes labeled by the names of the

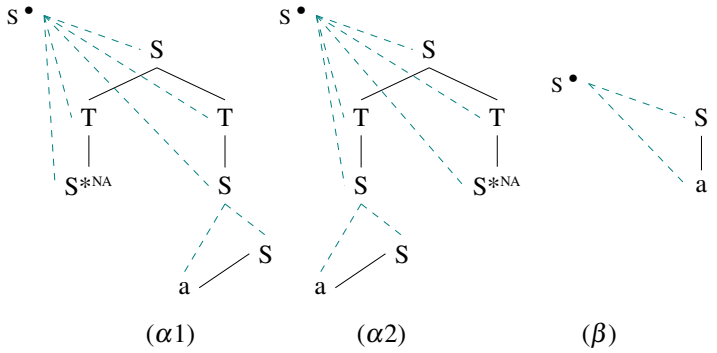

Figure 7: The lexicalized 3-dimensional TAG

trees in I, and $B \subseteq N \times Add \times N$ is a set of branches labeled by tree addresses $a \in Add$. Let us consider $t_1, t_2 \in I$, and let u be the node at address a in $\mathrm{yd}^{d-1}(t_1)$. Then B contains an edge $\langle t_1, a, t_2 \rangle$ iff $t_1 \overset{d\ u}{\Longleftarrow} t_2$ is licensed by the grammar. A tree $t \in I$ is *recursive* iff t in G^d belongs to a cycle of $\mathscr{G}$. We use R to denote the set of all recursive initial trees, whereas $NR := I - R$ is the set of non-recursive trees.

Step 2: Construct the set I_{lex} of initial trees. We use $T(NR)$ to denote the closure of NR under adjunction. Since all members of NR are non-recursive, $T(NR)$ is finite. The set I_{lex} of initial trees for the lexicalized d-TAG G_{lex} is the maximal subset of $T(NR)$ that only contains d-trees whose root is labeled by the start category S. As the empty string is not generated by G^d, the initial trees of G_{lex} have at least one terminal symbol on the frontier. In the example in Fig. 6, β is the only initial tree of the new grammar.

Step 3: Compute Cycles. A cycle of $\mathscr{G}$ is *minimal* iff it contains no proper subgraph that is also a cycle. Let C be a set of minimal cycles of $\mathscr{G}$ such that the closure of C under graph composition is the set of all cycles in $\mathscr{G}$. The members of C are referred to as *base cycles*.

Step 4: Construct the set of auxiliary trees. Each base cycle $c_i \in C$ is decomposed as a tree as follows: I) every node of the cycle is expanded as the tree whose name labels the node; II) substitution of a tree into another is performed at the addresses labeling the edge connecting the corresponding nodes; III)the edge leading to recursion is *cut* from the graph and the tree node whose address labeled that edge is labeled as the foot node. A null adjunction constraint (NA) is also put on the foot nodes to disallow recursive adjunction within the auxiliary trees. The set of auxiliary trees for G^d_{lex} is initialized to the empty set A. Then, substitution is exhaustively applied to the nodes in the yield of any of the base cycle trees — except for the unique node marked as a foot node — through the elements of the set of initial trees. The resulting set of trees is the set of auxiliary trees of the lexicalized d-dimensional TAG G^d_{lex}. This concludes the construction of G^d_{lex}.

Figure 7 shows the final lexicalized 3-dimensional grammar. The set of initial trees contains tree β, whereas $\alpha 1$ and $\alpha 2$ belong to the set of auxiliary trees. The resulting d-TAG G^d_{lex} generates exactly the same set of trees as the original, finitely ambiguous d-TSG G^d.

We can now see how previous results directly imply that d-dimensional TAGs strongly lexicalize d-dimensional TSGs. Furthermore, in the previous section we discussed how d-TAGs are equivalent to $(d+1)$-TSGs. This jointly implies that $(d+1)$-TAGs strongly lexicalize d-TAGs.

Proposition 3. For each finitely ambiguous d-dimensional TAG that does not generate the empty string and contains only useful trees, there is a strongly equivalent $(d+1)$-dimensional Lexicalized TAG.

3.3 d-TAGs are Not Closed Under Strong Lexicalization

In the previous sections we have shown that each d-TSG has a strongly equivalent $d+1$-TAG, and that d-TAGs are strongly lexicalized by $(d+1)$-TAGs. These results have been obtained by lifting previous proofs for TAGs to d-dimensional TAGs. However, one might wonder if the extension of TAGs to multidimensional structures is enough to assure lexicalization. In other words, are d-dimensional TAGs closed under strong lexicalization?

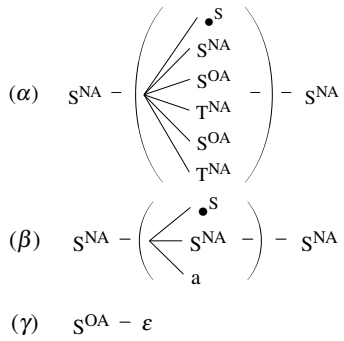

Figure 8: Parenthetical encoding for the 4-dimensional version of the non lexicalized grammar in Fig. 6

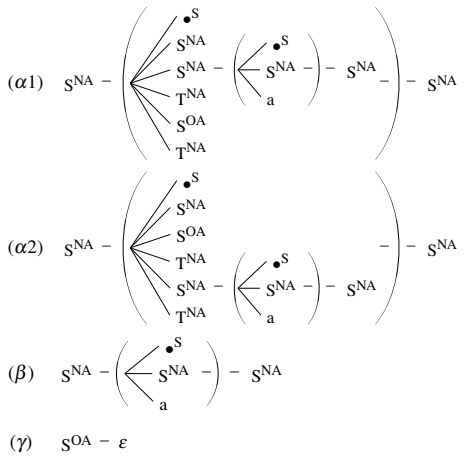

Figure 9: Parenthetical encoding for a 4-dimensional lexicalized grammar

Diligent readers may have already anticipated that the answer is negative. As before, it suffices to lift an existing proof from the literature to higher dimensions, in this case Kuhlmann and Satta's (2012) result that TAGs are not closed under strong lexicalization. This entails that the increase in dimensionality from d to $d+1$ brought about by the strong lexicalization procedure cannot be avoided in the general case.

Kuhlmann and Satta start out with the observation that adjunction may be regarded as context-free rewriting on the path of trees. They build a counterexample grammar to show that a lexicalized TAG cannot generate the same trees as the non-lexicalized, original TAG. They then provide a proof based on the intuition that, while a non-lexicalized grammar can potentially extend the length of the shortest path from the tree root node to any terminal node *ad infinitum*, the length of such paths is finitely bounded for the corresponding lexicalized grammar. This property also holds for d-TAGs, where recursive adjunction of d-trees that do not contain any lexical nodes can increase the length of paths without bounds.

Kuhlmann and Satta's proof rests on two basic concepts. The first one is their *parenthetical* notation, which represents trees as string paths and thus highlights how adjunction operates on the path from the root node to a foot node (the *spine* of a tree). The second is a function called *excess*, which is used to measure the distance between a root node and a terminal node on a tree path.

The *parenthetical* encoding of TAGs can be adopted for multidimensional trees without problems. Consider some arbitrary d-dimensional tree T^d. Every leaf node of T^d is reachable from its d-dimensional root following the branches encoding the d-dimensional dominance relation. This unique path from the root node to a frontier node along the d-th dimension is the *spine* of the d-dimensional structure (cf. Rogers (2003a)). The parenthetical notation simply encodes the d-dimensional spine of d-trees: every internal node n along the d-th dimension is represented by a pair of matching brackets, e.g. "`S−(`" and "`)−S`"). These brackets surround the parenthetical encoding of the d-dimensional subtree rooted in n. A tree encoded in this notation is called a *spinal tree*. Examples are given in Fig. 8 and 9.

Since d-dimensional trees can be described in the same fashion as 2-dimensional ones via the parenthetical notation, the notion of *excess* remains untouched. For every terminal node v, the excess of v is the mismatch of left parentheses over right parentheses in the sequence of nodes in the path from the d-dimensional root node to v. This is done by subtracting the number of all right parenthesis from the number of all left parenthesis that occur along the path from the root to v. For example, the excess of a in $\alpha 2$ of Fig. 9 is 2. The *excess* of the whole spinal tree is always 0.

Let us now return to the 3-TAG given in Fig. 6. It can be lifted to a 4-dimensional TSG as described in the Sec. 2.3. Figure 8 shows a parenthetical rewriting of the 4-dimensional trees in the grammar, while Fig. 9 shows the rewriting of a possible lexicalized d-dimensional grammar, obtained via the procedure in Schabes (1990). Let us call the non-lexicalized parenthetical encoding G_1, and the hypothetical lexicalized version G_1^{lex}. If G_1^{lex} is a correct lexicalization of G_1, it should generate exactly the same 4-dimensional trees. But

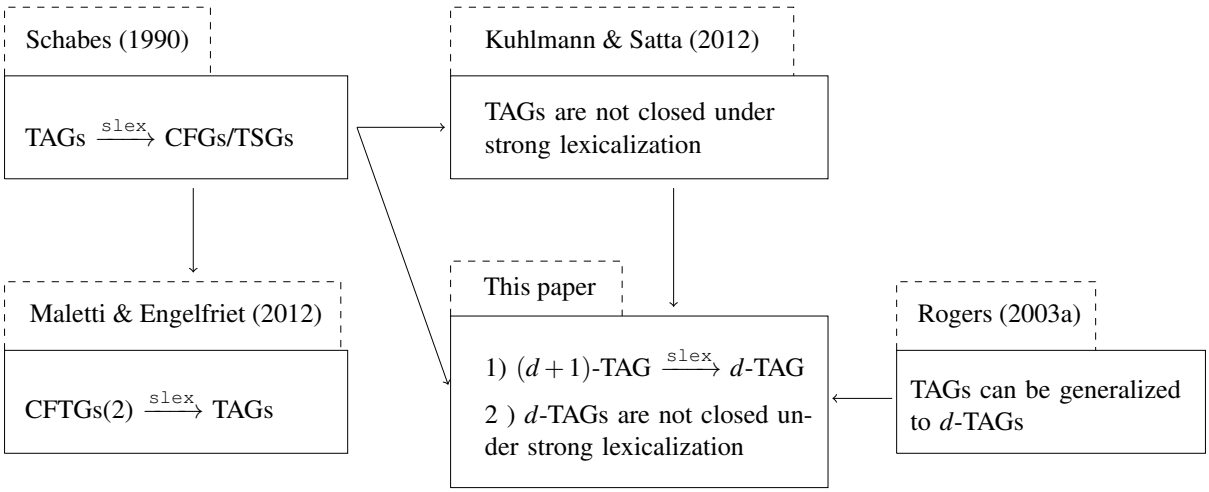

Figure 10: Summary of relevant existing results

the notion of *excess* reveals that this is not the case. In G_1, adjunction of α into another instance of α can take place an arbitrary number of times. As a result, when β finally adjoins into α, an unbounded number of left brackets may occur between the root of the tree and leaf node a of β, wherefore there is no upper bound on the excess of trees generated by G_1. For G_1^{lex}, on the other hand, every tree must contain a lexical node with excess at most 2. Hence G_1 and G_1^{lex} cannot generate the same language of d-trees.

While this simple example does not preclude that some other grammar strongly lexicalizes G_1, this is in fact impossible. Kuhlmann and Satta prove that the potential variation in the excess of a lexical node in an elementary tree is bounded by a constant tied to the grammar, while the excess of a lexical node in a non-lexicalized grammar is potentially infinite. Like so many other facts about 3-TAGs, this one carries over to d-TAGs, too, and consequently d-TAGs turn out not to be closed under strong lexicalization either.

4 Conclusion

The generalization of TAGs to higher-dimensional trees first proposed in Rogers (2003a) is very useful in increasing the power of TAGs while preserving the mechanics of their structure-building operations. This paper has exploited this fact to lift a variety of previously known results about TAGs to higher dimensions, culminating in the realization that TAGs are strongly lexicalized by 4-dimensional TAGs and, more generally, that every d-TAG is strongly lexicalized by some $(d+1)$-TAG. A major advantage of lexicalization is that it simplifies the parsing problem. At the same time, increasing the dimensionality of TAGs makes parsing harder, which can be gleaned from the fact that 4-dimensional TAGs can generate the 8-language $a^n b^n c^n d^n e^n f^n g^n h^n$, whereas standard (3-dimensional) TAGs are restricted to the 4-language $a^n b^n c^n d^n$. An interesting question for future research will be whether the simplifications of lexicalization can offset the parsing disadvantages of higher dimensions.

Acknowledgments

We are very grateful to the three anonymous reviewers, whose comments led to several improvements in the presentation of the material.

References

Akio Fujiyoshi. 2004. Epsilon-free grammars and lexicalized grammars that generate the class of the mildly contextsensitive languages. In *Proceedings of the 7th International Workshop on Tree Adjoining Grammar and Related Formalisms*, pages 16–23.

Frank Harary. 1972. *Graph Theory*. Addison-Wesley, Reading, MA.

Aravind K. Joshi and Y. Schabes. 1997. Tree-adjoining grammars. In Grzegorz Rosenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, pages 69–123. Springer, Berlin.

Aravind K. Joshi. 1985. Tree-adjoining grammars: How much context sensitivity is required to provide reasonable structural descriptions? In David Dowty, Lauri Karttunen, and Arnold Zwicky, editors, *Natural Language Parsing*, pages 206–250. Cambridge University Press, Cambridge.

Laura Kallmeyer. 2010. *Parsing Beyond Context-Free Grammars*. Springer, Berlin.

Marco Kuhlmann and Giorgio Satta. 2012. Tree-adjoining grammars are not closed under strong lexicalization. *Computational Linguistics*, 38:617–629.

Andreas Maletti and Joost Engelfriet. 2012. Strong lexicalization of tree adjoining grammars. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 506–515.

James Rogers. 1998a. A descriptive characterization of tree-adjoining languages. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING'98) and the 36th Annual Meeting of the Association for Computational Linguistics (ACL'98)*, pages 1117–1121.

James Rogers. 1998b. On defining TALs with logical constraints. In Anne Abeillé, Tilman Becker, Owen Rambow, Giorgio Satta, and K. Vijay-Shanker, editors, *Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*, pages 151–154.

James Rogers. 2003a. Syntactic structures as multi-dimensional trees. *Research on Language and Computation*, 1:265–305.

James Rogers. 2003b. wMSO theories as grammar formalisms. *Theoretical Computer Science*, 293:291–320.

Yves Schabes, Anne Abeillé, and Aravind K. Joshi. 1988. Parsing strategies with 'lexicalized' grammars: Application to tree adjoining grammars. Technical Report MS-CIS-88-65, Department of Computer & Information Science, University of Pennsylvania, Philadelphia, PA.

Yves Schabes. 1990. *Mathematical and Computational Aspects of Lexicalized Grammars*. Ph.D. thesis, Philadelphia, PA, USA.