LFG and Tree-Adjoining Grammar

Jamie Y. Findlay
University of Oslo

Version of November 30, 2022 – under review; please do not cite

1 Introduction and roadmap

The purpose of this chapter is to give an introduction to some of the properties of Tree-Adjoining Grammar (TAG: Joshi, Levy, and Takahashi, 1975; Joshi and Schabes, 1997; Abeillé and Rambow, 2000; Joshi, 2005; Kallmeyer, 2010, ch. 4) and to draw some comparisons with Lexical Functional Grammar (LFG: Kaplan and Bresnan, 1982; Bresnan, Asudeh, et al., 2016; Dalrymple, Lowe, and Mycock, 2019; chapters/Intro). It is primarily aimed at those familiar with LFG who are looking to learn about TAG and see where the two formalisms differ/overlap, but the comparisons will also be of interest to those coming from a TAG background (although details of the LFG formalism will not be covered – the interested reader is directed to the references above and other chapters in this volume).

A TAG is a mathematical formalism for describing a set of trees, just as a context-free grammar (CFG) is a mathematical formalism for describing a set of strings. Unlike a CFG, which generates or recognises a string by repeatedly rewriting symbols until the target string is produced, a TAG does so by combining members from a starting set of ELEMENTARY TREES using two operations (called SUBSTITUTION and ADJUNCTION) until a tree whose yield is the target string is produced. This gives TAGs a greater generative capacity than CFGs, which is why they are of particular interest to researchers in natural language syntax: since Shieber (1985), we have known that the complexity of natural language syntax exceeds the context-free space which CFGs are capable of describing. In Section 2, I will briefly discuss this finding and the choice it has forced modern linguistic theories to make regarding their formal foundations. Section 4 delves more deeply into the generative power of TAG and compares it to LFG.

When TAG is used in a linguistic capacity, a number of properties are generally added to the basic formalism in order to better align it with certain theoretical assumptions and to enable more natural or transparent analyses of particular grammatical phenomena (e.g. the inclusion of feature structures on nodes to facilitate an analysis of agreement). In Section 3, I introduce some of the details of the TAG formalism along with these linguistically-motivated theoretical assumptions.

One important property generally assumed in linguistic applications of TAG is known as LEXICALISATION, the property whereby each of the basic structures of a grammar is associated with a single lexical item. Lexicalised grammars purportedly have a number of desirable traits, and I discuss lexicalisation in more detail in Section 5. This property sets TAG apart from CFG-based formalisms such as LFG, since the latter cannot in general be lexicalised – a perhaps surprising result given the lexical focus of LFG.

Section 6 briefly compares the TAG and LFG approaches to the factoring out of redundancies from grammars. TAG makes use of a so-called METAGRAMMAR, a formal system used to produce grammars, which can capture high-level generalisations and make grammar engineering easier. LFG uses TEMPLATES, which are part of the grammar proper, and allow pieces of linguistic description to be given names and reused.

Lastly, Section 7 considers the possibility of incorporating a TAG into the LFG architecture, replacing the standard CFG-based description of c-structure. This offers fertile new analytical possibilities, and has pleasing consequences for the descriptive power of LFG more generally, since it allows templates to be extended to the domain of phrase structure, opening the door to a fully constructional LFG.

2 Moving beyond context-free grammars

Context-free grammars have played (and continue to play) a major role in the development of syntactic theory at least since their formal elaboration in the 1950s (Chomsky, 1956), with their conceptual roots going back even further (at least to e.g. Harris, 1946; Wells, 1947). However, there was also always a sneaking suspicion that natural language syntax was formally more complex than CFGs could describe. Nevertheless, by the early '80s there had been no successful proof of this fact (Pullum and Gazdar, 1982). In 1982, Bresnan, Kaplan, et al. (1982) demonstrated that the presence of cross-serial dependencies means that the dependency structure of Dutch requires more than context-free power to describe, although owing to the lack of morphological marking of such dependencies, the *string* language of Dutch remains context free. It turns out, however, that Swiss German exhibits the same cross-serial dependencies, but its nouns are case-marked, and since different verbs can assign different cases to their objects, this means that the dependencies show up in the string language as well. Thus, greater-than-context-free power is definitely needed to describe Swiss German (Shieber, 1985). Since there is no reason to suspect that speakers of Swiss German are biologically distinct from speakers of other languages, or that some people would be intrinsically unable to learn Swiss German, this means that the human language faculty allows for languages which require greater than context-free power to describe, and so CFGs alone are inadequate as the basis of a grammatical formalism.

Given this fact, there are two different kinds of response for the syntactic theorist. We can either

1. replace the CFG with something more powerful; or
2. beef up the CFG with something extra, so that the *combination* is more powerful.

Chomskyan generative grammar had already taken the second approach from the start: the addition of transformations to a CFG base pushes the formalism well beyond context-freeness, into the space of Type-0, unrestricted grammars (Peters and Ritchie, 1973). LFG similarly adds something extra to the CFG component: in this case, a separate level of representation, f-structure; the combination of the two takes the formalism as a whole into the Type-1, context-sensitive space (Berwick, 1982 – although see Section 4).

However, in order to account for cross-serial dependencies, we do not need a full-blown context-sensitive (or even more powerful) grammar. Instead, we only need a more modest MILDLY CONTEXT-SENSITIVE grammar (Joshi, 1985). Such grammars have the useful property, shared with context-free grammars, of being parsable in

polynomial time, unlike the (worst case) exponential parsing time of context-sensitive grammars (Joshi and Yokomori, 1983; Vijay-Shanker and Joshi, 1985), though they still go beyond the expressive power of CFGs in permitting the description of cross-serial dependencies. Responses to the challenge of the non-context-freeness of natural language which take approach number 1 above, and replace the CFG wholesale, tend to do so with a formalism which is explicitly mildly context sensitive, therefore, rather than anything more powerful. One example of this is Combinatory Categorial Grammar (CCG: Steedman, 1987; Steedman, 2000); another is TAG.[1]

3 An introduction to TAG

A TAG is a tree rewriting system which consists of a set of ELEMENTARY TREES and the two operations of SUBSTITUTION and ADJUNCTION for combining them. Substitution simply inserts one tree at the frontier of another (at a non-terminal node), while adjunction inserts a tree *inside* another, attaching it at a non-frontier node (more formal definitions of these processes will be given below).[2]

Most linguistic work in TAG now assumes a LEXICALISED version, LTAG (Schabes, Abeillé, and Joshi, 1988), in which each tree is anchored by, i.e. has as its terminal node(s), a single lexical item (which may still consist of several words, as in the case of phrasal verbs, idioms, etc.). Similarly, while trees in a TAG (*qua* mathematical formalism) can be of any size, in linguistic applications the general principle applied is that trees should correspond to the EXTENDED MAXIMAL PROJECTION (Grimshaw, 2000; Grimshaw, 2005) of a lexical item, i.e. the syntactic projection which includes all functional heads and the full argument structure of the item. Some examples of elementary trees matching these restrictions are given in Table 1. In this chapter, I will use "TAG" to refer specifically to this sub-class of lexicalised, linguistically-constrained TAG rather than merely to the mathematical formalism, except where otherwise indicated.

Elementary trees come in two types. An INITIAL TREE is one where all of the frontier nodes are either terminals or non-terminals marked as SUBSTITUTION SITES using the down arrow (↓).[3] Substitution sites indicate arguments of a predicate. An AUXILIARY TREE is like an initial tree except that one of the frontier nodes, called the FOOT node, shares the same label as the root, and is marked with an asterisk (*). Auxiliary trees are combined with other trees via adjunction, to be described below.

[1] In fact, things are a little more complex than this. In some definitions of mild context sensitivity (e.g. Kallmeyer, 2010, pp. 23–24), "permutation-complete" languages like MIX (the language consisting of the subset of $\{a, b, c\}^*$ with an equal number of as, bs, and cs; see Bach 1981) are included (Salvati, 2015; Nederhof, 2016); in others (e.g. Joshi, Vijay-Shanker, and Weir, 1991), they are not (Kanazawa and Salvati, 2012). TAG and CCG are in the class which does not contain MIX, and so Steedman (2019, p. 415) suggests they should be called SLIGHTLY NON-CONTEXT-FREE to distinguish them from the larger class which does contain MIX. Nothing in this chapter hinges on this distinction, so I continue to use the more traditional "mildly context sensitive", without taking a position on whether or not this refers to the class of languages containing MIX or not.

[2] In the original formulation (Joshi, Levy, and Takahashi, 1975), TAG only has one combining operation – adjunction. The addition of substitution, however, improves the descriptive capabilities of the framework, making it easier to use for linguistic purposes, while leaving its formal expressive power the same, since adjunction can be used to simulate substitution (Abeillé, 1988, p. 7). In this chapter I will therefore continue to assume that both operations are used.

[3] As can be seen from this definition, every non-terminal frontier node is a substitution site, so the ↓-annotation is formally redundant. Nevertheless, it is often included in expository text (if not in computational implementation) to make it clear at a glance that a tree does not represent a fully completed derivation.

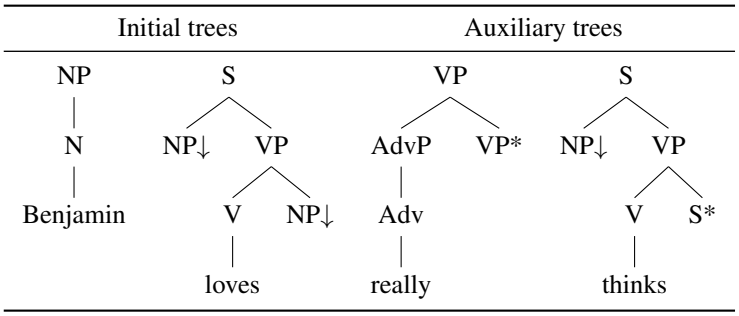


Table 1: Some elementary trees

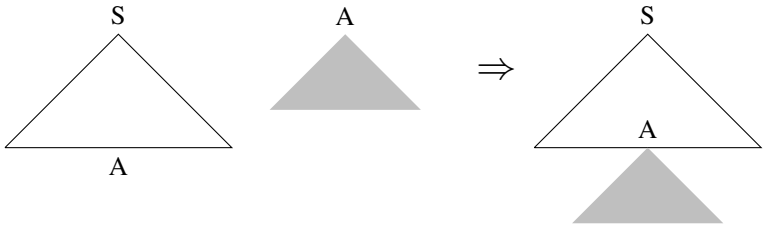
When two elementary trees have been combined, we have a DERIVED TREE, which can then be further manipulated just like an elementary tree.

In the next two sections, I introduce the two operations used to combine trees in TAG: substitution and adjunction.

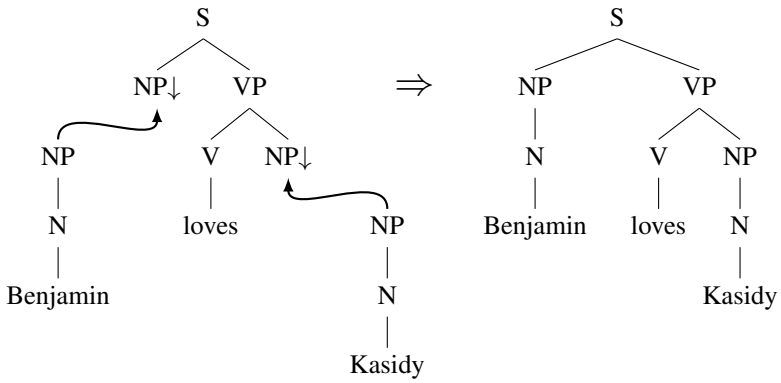
3.1 Substitution

When the root of a tree has a label which matches that of a non-terminal frontier node in another tree, the first tree can be inserted at that frontier node in the second; this is called substitution, and is the process normally used to combine a predicate and its arguments. Example (1) shows this process schematically, while (2) gives a linguistic example involving two cases of substitution: the derivation for *Benjamin loves Kasidy*.

(1) **Substitution** (after Abeillé and Rambow, 2000, p. 5)



(2)

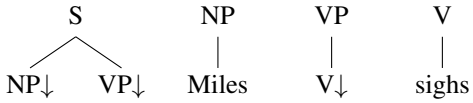


A tree rewriting grammar which makes use of substitution alone is called a TREE SUBSTITUTION GRAMMAR (TSG), and is at least weakly equivalent to a CFG – that is,

such grammars describe the same set of string languages (weak equivalence), although there are some tree languages which which can be described by a TSG for which an equivalent CFG does not exist (so strong equivalence is not guaranteed).[4] This is easy to see if we imagine converting a CFG into a TSG: all we do is replace each phrase-structure rule with the equivalent tree which it describes (cf. McCawley's 1968 conception of phrase-structure rules as node admissibility conditions, i.e. descriptions of trees). For example, the CFG in (3) corresponds to the TSG in (4):

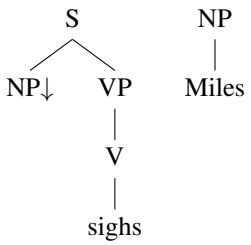
(3) S $\longrightarrow$ NP VP
NP $\longrightarrow$ Miles
VP $\longrightarrow$ V
V $\longrightarrow$ sighs

(4)



Although TSGs and CFGs are formally very close (at least weakly equivalent), there is an important theoretical difference between them: TSGs have an EXTENDED DOMAIN OF LOCALITY with respect to CFGs. Every rule in a CFG describes a tree of depth 1, but trees in a TSG can be of arbitrarily large size, which means that certain grammatical dependencies, like agreement or extraction, can be described locally in a TSG (i.e. in a single elementary tree) when they cannot be in a CFG (i.e. they cannot be described in a single rule). For example, the TSG in (5) is equivalent to that in (4), except that now the verb and its subject are in the same elementary tree, and so the agreement relationship between the two could be described locally.

(5)



Many possibilities exist as to *how* to describe such a dependency – for example, by using complex categories in the style of GPSG (on which see Gazdar et al. 1985), or by using feature structures (on which see Section 3.3); but the point is that however one chooses to represent this relationship, it can be described locally in a TSG when it can only be described indirectly in a CFG, e.g. by percolating features up from V to VP, thus making them visible to the S $\rightarrow$ NP VP rule which introduces the subject.

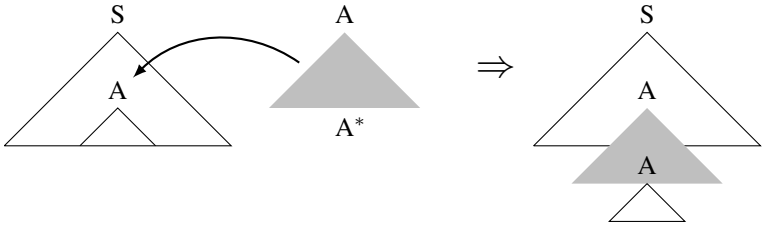
3.2 Adjunction

If we add adjunction, a second type of combining operation, to a TSG, we obtain a TAG. While substitution allows a tree to be inserted at a frontier node of another tree,

[4] Any CFG can easily be converted into a TSG by simply turning each phrase-structure rule into a tree rooted in the left-hand symbol with the right-hand symbols as daughters, as will be illustrated in the text. But to convert a TSG into a CFG, it may be necessary to relabel some nodes, since the dependency between a mother and its daughters may be tree-specific, and so not hold generally (for example, it might be the case that only trees anchored by transitive verbs have a VP dominating both a V and an NP node) – so the CFG might have to have more non-terminal symbols than the TSG, e.g. VP_{trans} and $\text{VP}_{\text{intrans}}$ instead of just VP.

adjunction allows insertion at a *non*-frontier node: the adjoining tree expands the target node around itself. A tree which adjoins into another tree, called an auxiliary tree, must therefore have at least one frontier node with the same label as its root – this is called the foot node. The process of adjunction is represented schematically in (6):

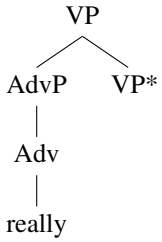
(6) **Adjunction** (after Abeillé and Rambow, 2000, p. 9)



Because of the requirement that the root and foot of an auxiliary tree have the same label, such trees can be seen as factoring recursion out of the grammar. Rather than having a cyclic path through the rewrite rules (as in a CFG), we have a tree which directly encodes such a cycle (in (6), an A contained within an A), which can then be added into a structure via adjunction. For this reason, such auxiliary trees are used to model the recursive aspects of natural language syntax – most notably modification and sentential embedding.

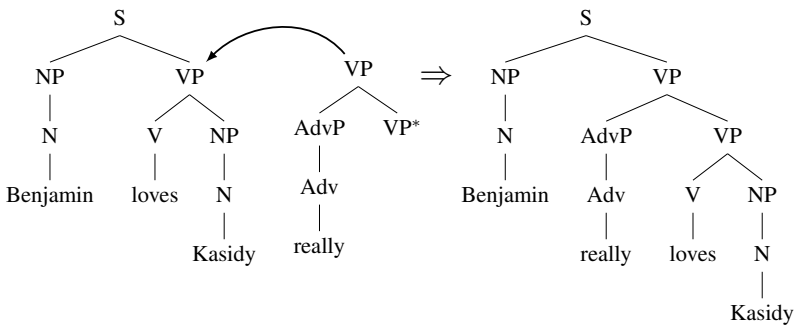
Modifiers such as adjectives and adverbs, but also e.g. relative clauses, are represented as auxiliary trees. For example, *really* is a VP-adverb which appears to the left of the VP it modifies, and so is represented by the following tree:

(7)



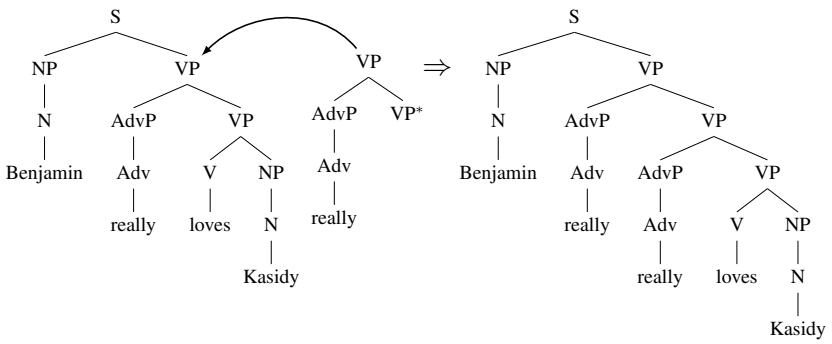
To see this in use, consider the derivation for the sentence *Benjamin really loves Kasidy*: after the substitutions shown above in (2) to generate *Benjamin loves Kassidy*, we can then adjoin the tree from (7) at the VP node in the clause:

(8)



Of course, such a process can be repeated indefinitely many times, since there is always still a tree-internal VP node available to be adjoined to:

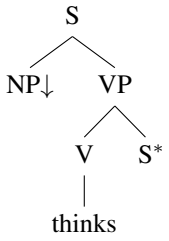
(9)



This accounts for the iterability of modifiers like *really*.

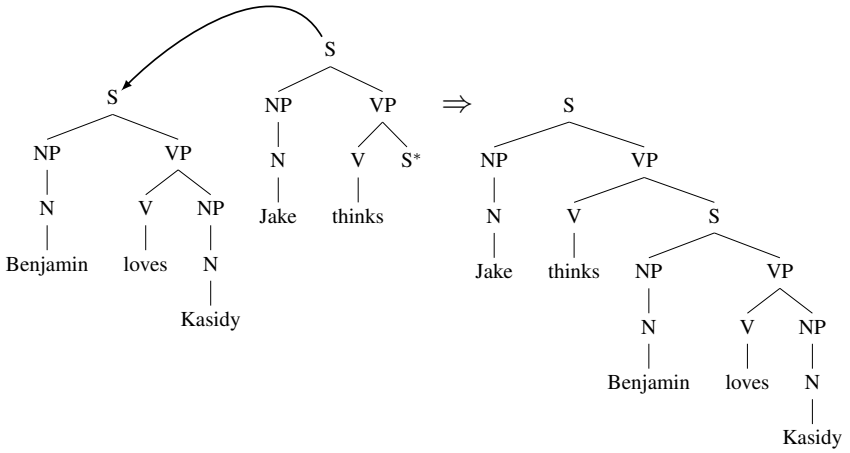
Another, perhaps more theoretically interesting, area of recursion in the grammar is in the domain of subordination, i.e. sentential embedding. Verbs which take sentential complements are represented as auxiliary trees, as in (10), for example:

(10)



Notice that this means that sentential arguments are treated rather differently from other arguments in TAG: while arguments are normally combined with their governors by means of the former being substituted into the latter, sentential arguments combine with their governors by means of the latter being adjoined into the former – this is shown in (11).

(11)



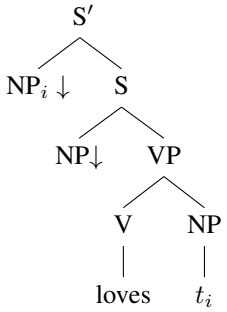
For simple declarative sentences this is a rather unnecessary complication, since the same effect could be achieved by making the foot node of the sentential-embedding

verb a substitution site instead. However, the factoring of recursion into auxiliary trees interacts with another TAG principle – the local representation of syntactic dependencies. Owing to their extended domain of locality, it is possible to represnt many kinds of syntactic dependencies locally (i.e. in a single elementary structure) in TAG that would require some additional mechanism in other frameworks. This principle extends to filler-gap relations as well, such as that between a fronted focus phrase and its verbal governor, as in (12):

(12) *Kassidy* Benjamin loves (whereas Kira he merely likes).

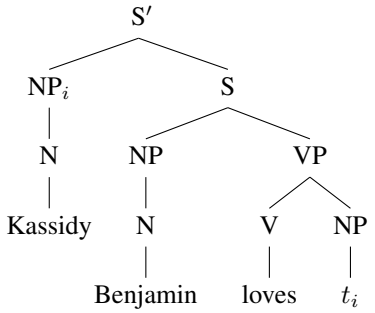
The tree in (13) represents the appropriate form of the verb *loves*, with its object extracted:[5]

(13)



Through substitution alone, this can be used to derive (14):

(14)



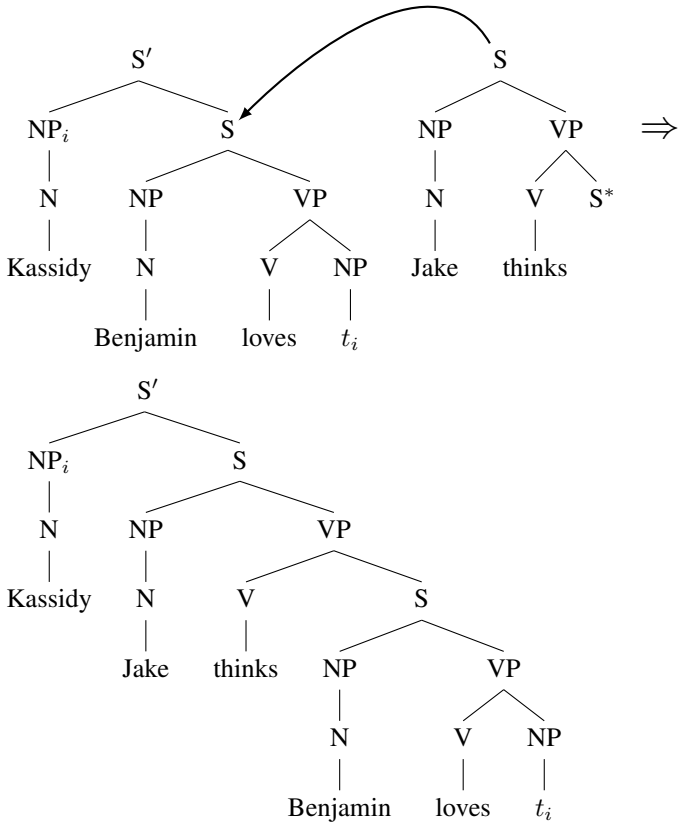
But of course the distance between the fronted phrase and the gap can span multiple clauses, and can be arbitrarily large:

(15) *Kassidy* [Jadzia knows [Jake thinks ... [Benjamin loves]]].

Since sentential embedding verbs are treated as auxiliary trees, this poses no problem – they are adjoined to the internal S node, and thus extend the distance between the gap and the filler:

[5]The use of a trace in object position here is not an essential part of the TAG analysis, though in practice it is common. One reason for this, as argued for by Kroch, 1987 and Kroch and Joshi (1985), is that empty elements allow for easier specification of some constraints on extraction in terms of the topology of trees, rather than necessitating additional mechanisms, like functional uncertainty and off-path constraints in LFG. A reviewer points out that traces are also useful in the metagrammar (see Section 6 on this concept), since they allow tree fragments to be reused more easily.

(16)



What is more, this can clearly be repeated: other trees can be adjoined at the topmost S node, further increasing the distance between the filler and the gap. Thus, a potentially quite radically non-local dependency, between the fronted expression and its governing verb, can be expressed locally in the grammar, in a single elementary tree, because the operation of adjunction allows for the distance between nodes in a tree to grow over the course of a derivation. This same process can be applied to other kinds of filler-gap dependencies, such as *wh*-questions and relative clauses, though for ease of exposition I have chosen not to illustrate those here (since there we must also account for things like subject-auxiliary inversion and *do*-support).[6] The TAG approach is in contrast to that of many other syntactic theories which instead derive or infer the relation between filler and gap via some additional syntactic mechanism, be that movement or, in the case of LFG, a functional uncertainty path.

3.3 Expressing constraints

Adjunction of an auxiliary tree, which has a root and foot node with the same label, captures the effects of recursion in other formalisms. However, once an auxiliary tree has been adjoined in, there will be two nodes with the same label where there was previously just one. This means that if we adjoin the same tree again (e.g. as in (9),

[6]The interested reader should consult the detailed analyses of these and other constructions in English provided by the XTAG project (XTAG Research Group, 2001).

where we adjoin *really* twice), there are two distinct possible targets (and after that adjunction there will be three, etc.). This has the potential to dramatically complicate parsing, since there will be multiple distinct possible derivations for the same tree (without there also being a genuine ambiguity of interpretation), and so we would like a means of controlling where adjunction takes place. TAG originally did this by using local constraints on adjunction (Joshi, 1987, 100ff.), annotations added to the nodes of elementary trees indicating which auxiliary trees can be adjoined there. If the list of adjoiners is empty, we have a NULL ADJUNCTION (NA) constraint, which prohibits adjunction at the node. If the list is non-empty, then we have a SELETIVE ADJUNCTION (SA) constraint, which limits the trees which can adjoin. There are also OBLIGATORY ADJUNCTION (OA) constraints, which are like SA constraints except that one of the listed trees *must* be adjoined at the annotated node. In classic TAG, this is achieved simply by a diacritic indicating that the constraint is an OA one rather than an SA one. We will see below how this can be achieved in a less stipulative way by making use of feature structures.

Using these constraints, we can avoid having multiple possible parses for sentences by marking the foot node of auxiliary trees with an NA constraint (as is done in the XTAG grammar of English, for example – XTAG Research Group, 2001). This means we do not add an extra potential target for adjunction each time such a tree is adjoined in, since only the root of the auxiliary tree is available for further adjunction.

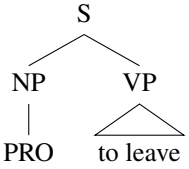
In addition to this practical motivation, adjunction constraints play a crucial theoretical role: they are a vital part of what makes TAG mildly context sensitive. Without adjunction constraints, the formalism is still more powerful than a CFG, but there are several mildly context-sensitive languages which it cannot express, such as the copy language $\{ww \mid w \in \Sigma^*\}$, or the language $\{a^n b^n c^n \mid n \geq 0\}$, also called COUNT-3 (Kallmeyer 2010, pp. 27, 58; we return to COUNT-3 in Section 4).

Let us now consider an example illustrating the other two kinds of constraints, which shows their use in linguistic theorising. Vijay-Shanker (1987, pp. 134–135) considers non-finite sentential complements such as (17):

(17) John tried [PRO to leave].

Assuming the subordinate clause has the tree in (18), our analysis needs to do two things: ensure that such clauses cannot appear on their own as full sentences – cf. (19) – and ensure that they can only be embedded under verbs that select for infinitival forms – cf. (20).

(18)



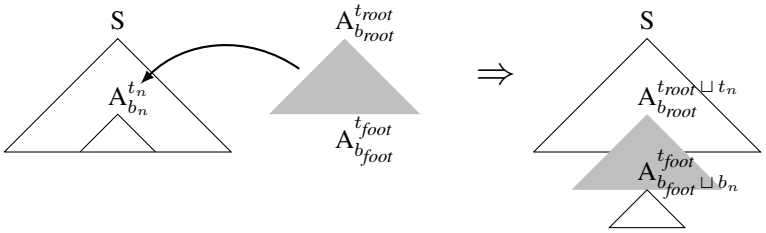
(19) *To leave.

(20) a. John tried to leave.
b. *John imagined to leave.

In other words, *something* must be adjoined into the root node S (an OA constraint), and that something must only be a sentential embedding verb that selects for a non-finite complement clause (an SA constraint).

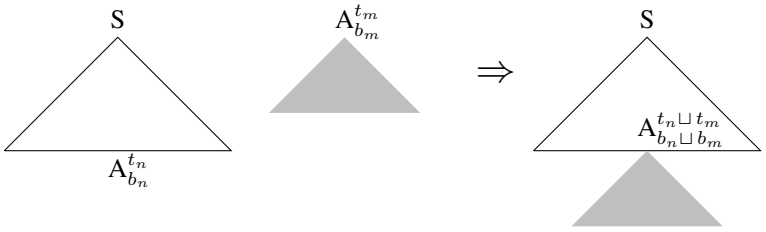
Originally, these constraints were simply seen as listings of (permitted/required) auxiliary trees, but this is not particularly linguistically illuminating, and also difficult to maintain for a grammar writer. This is remedied in later TAG work through the use of feature structures. It is common to associate nodes with feature structures in CFG-based grammars (e.g. in GPSG – Gazdar et al. 1985) in order to represent grammatical features such as case, number, tense, etc. Indeed, this is what LFG's f-structures do too (although there multiple structures from different nodes are merged into one). However, in a TAG, we cannot guarantee the integrity of each node in the tree: through adjunction, it may be split up into two nodes, corresponding to the root and foot nodes of an auxiliary tree. For this reason, in feature structure-based TAG (FTAG: Vijay-Shanker, 1987, ch. 5; Vijay-Shanker and Joshi, 1988), each node is associated with a *pair* of feature structures, called the TOP and BOTTOM feature structures. The top features refer to the relation of the node to its siblings and its ancestors, i.e. the view from above the node in a tree. The bottom features refer to its relation to its descendants, i.e. the view from below (Vijay-Shanker, 1987, p. 129). Ultimately, the top and bottom features of a node must unify, to give a single description of the properties of that node. However, during the course of a derivation, adjunction may split up the node so that it is now two nodes instead; in that case, its top feature will be unified with the top features of the root of the auxiliary tree involved, and its bottom features will be unified with the bottom features of the auxiliary tree's foot node. This is shown schematically in (21):

(21) **Adjunction in FTAG** (after Vijay-Shanker 1987, p. 130)



Substitution is simpler, since the root node of the substituted tree is simply identified with the substitution site, and so both top and bottom feature structures unify:[7]

(22) **Substitution in FTAG**

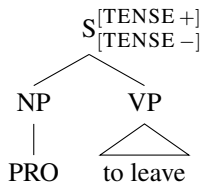


These feature structures can be used to enforce various linguistic constraints. For example, we can enforce subject agreement in initial trees for verbs by specifying number and person features on the subject NP position. More interestingly, we can use

[7]The diagram in (22) follows Vijay-Shanker's (1987) original formulation, where substitution sites also contain bottom features. In much other work using FTAG, this is not the case, so b_n in (22) would be absent, and the final bottom features of A in the derived tree would simply be b_m (XTAG Research Group, 2001, p. 13). This is of course equivalent to (22) with b_n instantiated as the empty set.

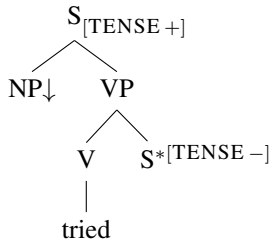
features to account for the constraints on adjunction discussed above. Because the features associated with whatever tree is adjoined at a node have to unify appropriately with its top and bottom features, we can control which trees are compatible by giving them (mis)matching features which make unification possible or not. What is more, we can give a more principled account of obligatory adjunction constraints by making the top and bottom features of a particular node incompatible with one another. This means that unless adjunction takes place and the node is split up, unification will be impossible, and the derivation will fail. Returning to our example from above, here is the tree from (18) with two added feature annotations:

(23)

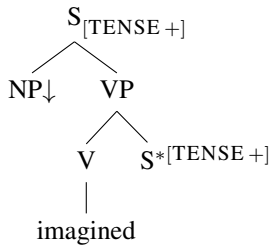


Since these features are incompatible and cannot unify, we have achieved the first of our goals, which is to ensure that this tree cannot appear on its own – i.e., to implement an OA constraint. Owing to the feature mismatch, this tree is illicit unless something adjoins to the root node. To achieve the SA constraint, we need to consider the elementary trees of verbs like *tried* and *imagined*. Here we present them with just the relevant features added:

(24)

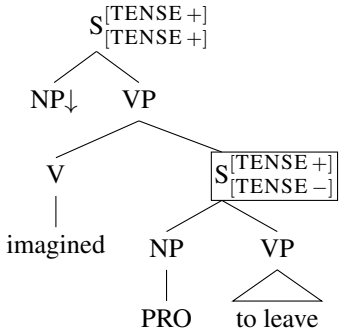


(25)



The difference between the two verbs is that *tried* selects for a non-finite, untensed, complement clause, whereas *imagined* selects for a tensed one – this is indicated by the top features on their foot nodes. If we attempt to adjoin *imagined* into the tree for the subordinate clause in (23), then we end up with mismatching features on the foot node, which means they cannot unify, and the tree remains illicit:

(26)



If we adjoin the tree for *tried*, however, then there is no mismatch, and the derivation succeeds.

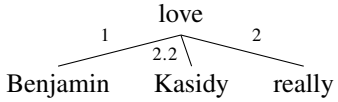
If we allow a fully-fledged unification-based feature system in FTAG, with recursive feature structures of potentially unbounded size, then FTAG becomes undecidable (Vijay-Shanker, 1987, 155f.). This is a very bad result given the emphasis that TAG places on tractable, polynomial parsing. For this reason, the feature structures in FTAG are more restricted, and do not permit recursion/re-entrancy, which makes them quite unlike LFG's f-structures.

3.4 Derivation trees and dependencies

In a CFG as classically conceived, the familiar phrase-structure tree is in fact a representation of the derivation, i.e. of the process by which the output, namely the string, was produced. TAGs also have these DERIVATION TREES, representing the way in which trees were combined during a derivation – but, of course, in a TAG, the output of the derivation is already a tree, called the "derived tree" to set it apart. The derived tree represents word order, constituency, and category information, like LFG's c-structure. So what linguistic information does a TAG derivation tree encode? Since each elementary tree in a (lexicalised) TAG corresponds to a lexical item, the derivation tree actually represents relations between lexical items, and so has much in common with a dependency grammar representation of the sort illustrated by Meaning-Text Theory (Mel'čuk, 1988), the more contemporary Universal Dependencies project (UD: Nivre et al., 2016), or, indeed, an LFG f-structure (on the relationship between dependency grammars and LFG, see also chapters/Dependency).

A derivation tree for *Benjamin really loves Kasidy*, the derivation for which was shown in (2) and (8), is given in (27) (cf. Joshi and Schabes, 1997, 74ff.):

(27)

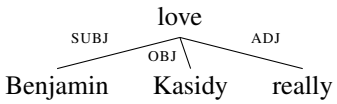


Here nodes are labelled with the lexeme corresponding to the elementary tree in question. Whenever a tree is substituted or adjoined into another tree, it becomes its daughter in the derivation tree. The derivation tree in (27) shows that three trees, corresponding to *Benjamin*, *Kasidy*, and *really*, were combined with the tree for *love*. Each edge is also labelled, standardly with a node address which indicates where the tree was substituted or adjoined.[8] However, we can equally well use different labels, such as

[8]These are so-called GORN ADDRESSES (Gorn, 1967). The root has the address 0 (or sometimes ϵ, i.e. the empty string), the kth child of the root (reading left-to-right) has the address k, and for all other nodes, the qth child of a node with address p has the address $p.q$ (Joshi and Schabes, 1997, p. 75).

assigning grammatical function names to argument positions and then treating other positions as ADJ (cf. Rambow and Joshi, 1997, p. 175). This would give us the following derivation tree instead of (27), making the parallel with dependency structures quite explicit:

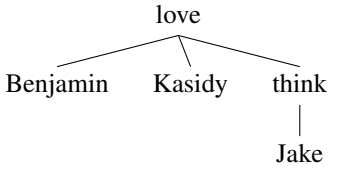
(28)



Rambow and Joshi (1997) discuss the relationship between TAG and dependency grammars in more detail.

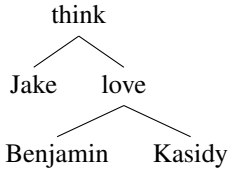
Unfortunately, the TAG treatment of sentential embedding somewhat undermines the neat parallel between derivation trees and dependency structures (Rambow, Vijay-Shanker, and Weir, 1995; Rambow, Vijay-Shanker, and Weir, 2001). Recall that arguments are normally substituted into their governors, but that clausal complements have their governors adjoined into them. This reverses the normal dependency relations, and means that "(standard) LTAG derivation trees do *not* provide a direct representation of the dependencies between the words of the sentence, i.e., of the predicate-argument and modification structure" (Rambow, Vijay-Shanker, and Weir, 2001, 117, emphasis in original). To see why this is so, consider the derivation tree for (11), *Jake thinks Benjamin loves Kasidy*:

(29)



Here *think* is a dependent of *love*, because the *think* tree is adjoined into the *love* tree, when of course in any real dependency grammar the relation would be reversed:

(30)



There are technical means of handling this unhappy result (see e.g. Joshi and Vijay-Shanker, 2001; Kallmeyer and Kuhlmann, 2012), but still it makes the parallel with dependency structures rather less direct. Nevertheless, we might be tempted to see the division of labour between derived trees and derivation trees in TAG as analogous to that between c-structure and f-structure in LFG, where the former represents constituency, word order, and category information, and the latter encodes a sentence's dependency structure. This is certainly true to a point, but the parallel is imperfect, because f-structure also represents other information beyond the dependency structure of a sentence – syntatic features like person, number, tense, aspect, etc., which in TAG are encoded in the feature structures associated with each node instead. Still, one thing that derivation trees and f-structures have in common is that they are both seen as the appropriate level of representation to serve as input to the semantic component of the grammar.

3.5 Semantics

There have been a variety of different proposals for interfacing TAG with a semantic theory, and space precludes a full presentation here. Nonetheless, I give a superficial overview here so that the interested reader can investigate further.

An early proposal for doing semantics with TAG makes use of SYNCHRONOUS TAG (STAG: Shieber and Schabes, 1990). In STAG, elementary trees from one grammar are paired with those from another, and links are established between individual nodes in those trees. Then, when adjunction or substitution applies in one grammar, it must also take place in the other, at the linked node, and using the equivalent, paired tree. By pairing a TAG grammar with a tree-based semantic representation, we can therefore implement a "rule-to-rule" approach to semantic derivation (Bach, 1976).[9] Nothing requires the paired trees to be isomorphic, so on the syntactic side it is the structure of the derivation tree, not the derived tree, which determines the meaning. Although this approach has largely fallen out of favour in TAG circles, see Nesson and Shieber (2006), Nesson and Shieber (2007), and Nesson and Shieber (2008) for a modern revival.

Another approach which uses the derivation tree as the basis for semantic interpretation is that of Joshi and Vijay-Shanker (2001). Here elementary trees are associated with triples of semantic expressions, the first of which specifies the main variable of the predication, the second of which gives the predicate with its arguments, and the third of which specifies which argument variables are associated with which nodes in the tree. When a tree is substituted into another tree, its main variable is identified with the corresponding argument variable in the target tree's semantics (special consideration has to be made for adjunction, as discussed above: the order of dominance in the derivation tree will be different for sentential vs. non-sentential complements – see Joshi and Vijay-Shanker 2001, 152f.). Since this makes use of a unification-based semantics, the order of combination of the elementary trees is irrelevant, and the derivation tree thus offers an appropriate level of representation, since it abstracts away from order, and simply says how and where trees were combined. This unification-based approach has been developed more recently by Laura Kallmeyer and colleagues, introducing a new focus on underspefication (Gardent and Kallmeyer, 2003; Kallmeyer and Joshi, 2003; Kallmeyer and Romero, 2004; Kallmeyer and Romero, 2008). This has also been integrated with Frame Semantics (Kallmeyer and Osswald, 2013).

In LFG the *de facto* standard approach to the syntax-semantics interface is GLUE SEMANTICS (chapters/Glue). Observing that, for example, the operation of function application as used in natural language semantics is order insensitive, and that quantifier scope ambiguities show that semantic interpretation does not (always) respect the constituent structure of a sentence (see Asudeh, 2012, ch. 5), Glue rejects c-structure as the appropriate level of input to semantics, and uses (a projection of) f-structure instead (where order is irrelevant and many c-structure hierarchies are collapsed). Thus, as in TAG, it is the dependency structure, not the phrasal structure, which is taken as relevant for semantic interpretation.

Interestingly, however, one of the only examples of TAG theorists criticising the derivation-tree-based approach to semantic interpretation is when Glue Semantics has been combined with TAG (Frank and van Genabith, 2001). Frank and van Genabith argue, mostly on the basis that, as discussed above, it provides the wrong dependency structure, that the derivation tree is not suitable as the input to semantic interpretation,

[9] Alternatively, by pairing two TAG grammars from different languages, we can implement a machine translation system – see Abeillé, Schabes, and Joshi (1990).

and they instead make use of the derived tree in their Glue-based framework.

3.6 The big picture

Linguistic theories based on TAG have two key properties (Joshi and Schabes, 1997, 95f.):

1. EXTENDED DOMAIN OF LOCALITY: Since TAG elementary trees encompass the whole extended projection of a lexical item, dependencies which in a simple CFG-based grammar would be spread across multiple rules, e.g. agreement, can be expressed "locally" in a TAG (i.e. in the same elementary structure). This is what enables a TAG to lexicalise a CFG (see Section 5).

2. FACTORING RECURSION FROM THE DOMAIN OF DEPENDENCIES: Relatedly, the elementary trees are the structures over which the vast majority of dependencies are stated, and that includes filler-gap relations. Such dependencies are therefore local in nature, but can become long distance via the adjunction operation. Recursion is thereby factored out of the domain over which these dependencies are initially stated.

This approach is summed up by Bangalore and Joshi (2010, p. 2) in the slogan "complicate locally, simplify globally". That is, local, elementary representations are where almost all linguistic constraints are stated, meaning that they can become quite complex, but the payoff is that the composition of elementary trees can be achieved by just two, very general operations, substitution and adjunction. This also means that cross-linguistic variation is entirely a matter of what elementary trees a grammar contains, a position very much in keeping what Baker (2008, p. 353) calls the BORER-CHOMSKY CONJECTURE, after Borer's (1984) proposal and Chomsky's (1995) later adoption of it, whereby parametric variation is restricted to the lexicon.

How does this compare with LFG? The second property certainly divides the frameworks: LFG grammars include recursive c-structure rules, and filler-gap dependencies are expressed syntactically, not lexically. This means, moreover, that the lexicon is not the only source of complexity in LFG grammars; many constructions are analysed as instantiating complex annotated phrase structure rules (see e.g. the analysis of long-distance dependencies in Dalrymple 2001, ch. 14). There is more overlap between the two frameworks when it comes to the first property. Via the parallel projection architecture ([see][sec. 5]chapters/Intro), LFG does obtain an extended domain of locality: for example, agreement can be encoded locally in the agreement controller's lexical entry via the use of paths through f-structure ([see][]chapters/Agreement). However, since c-structure is generated by a CFG, any non-local dependencies between c-structure nodes (i.e. those spanning more than one "generation" in the tree) can only be expressed indirectly via other levels of representation. That is, we have no extended domain of locality at c-structure *per se*, only parasitically via other levels. To the extent that phrasal constructions larger than a tree of depth 1 are objects we want to be able to represent in the grammar, this is a shortcoming. We return to this point in Section 7.2.

CONSTRUCTION GRAMMAR (CxG: Fillmore, Kay, and O'Connor 1988; Goldberg 1995; Goldberg 2006; Kay and Fillmore 1999; Boas and Sag 2012; Hoffmann and Trousdale 2013, *i.a.*) of course considers such objects as basic to linguistic theorising, and for this reason it has been argued that TAG is a natural means of formalising CxG (Lichte and Kallmeyer, 2017). For example, among the properties of constructions listed by Fillmore, Kay, and O'Connor (1988, p. 501), one is that they "need not be

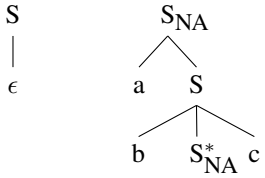
limited to a mother and her daughters, but may span wider ranges of the sentential tree" – precisely the enlarged definition of locality which a TAG provides, and which LFG denies (at least directly). TAG has a natural means of representing both "formal" and "substantive" idioms, to use Fillmore, Kay, and O'Connor's classification: formal idioms can be included in the set of trees associated with each lexical item of the appropriate class (Lichte and Kallmeyer, 2017, 208f.), and substantive idioms can be represented as elementary trees in their own right (Abeillé, 1995).

4 Generative capacity

TAG was designed specifically as a formalism with only mildly context-sensitive power (in the technical sense of Joshi 1985). This means there are languages out of the reach of context-free grammars that TAGs can describe, but also that there are languages properly considered context sensitive which TAG cannot. Such a constrained expansion into the context-sensitive space enables parsing algorithms for TAG to preserve the computationally appealing property of a polynomial run time.

As a simple demonstration of the increased generative capacity of a TAG when compared to a CFG, consider the artificial formal language $\{a^n b^n c^n \mid n \geq 0\}$, also known as COUNT-3 – that is, the language which contains all strings consisting of some number of as followed by the same number of bs, then the same number of cs. Partee, ter Meulen, and Wall (1990, p. 497) demonstrate through application of the pumping lemma for context-free languages that COUNT-3 is not context free. By contrast, there is a quite straightforward TAG grammar for COUNT-3, shown in (31):[10]

(31)



So we can see that TAGs are more powerful than CFGs. They are not, however, very much more powerful. There are many kinds of language which they cannot describe, including those which it has been shown can be described by similarly modest extensions to context-free grammars. One example of this is the language MIX (Bach, 1981), mentioned in footnote 1, which consists of all permutations of each string in the set $\{a^n b^n c^n \mid n \geq 0\}$, i.e. any number of as, bs, and cs, in any order, provided there is the same number of each. Salvati (2015 – originally circulated as a technical report in 2011) showed that MIX is in the class of multiple context-free languages, where a multiple context-free grammar (MCFG) is itself a mildly context-sensitive grammar formalism, for which the parsing problem is also decidable in polynomial time. However, MIX is *not* a tree-adjoining language, as conjectured by Joshi, Vijay-Shanker, and Weir (1991) and proved by Kanazawa and Salvati (2012): so there are languages which are only slightly within the context-sensitive space and which are still not describable by a TAG. More generally, although COUNT-3 and COUNT-4 (i.e. $\{a^n b^n c^n d^n \mid n \geq 0\}$) are tree-adjoining languages, COUNT-5 is not (Joshi, 1985, 223f.).

[10] Recall that a node annotated with "NA" bears a null adjunction constraint – see Section 3.3. As mentioned above, without adjunction constraints, TAG becomes less expressive, and cannot describe COUNT-3: see Kallmeyer (2010, p. 222) for a proof.

The carefully constrained computational complexity of TAG is in marked contrast to the situation in LFG (although see below for attempts to constrain the power of the LFG formalism). Whereas the class of tree-adjoining languages is equivalent to that of the mildly context-sensitive languages (or, perhaps, the slightly non-context-free languages: see fn. 1), the languages described by LFGs are equivalent to the class of recursively enumerable languages (Nakanishi, Seki, and Kasami, 1992). This has the expected deleterious effect on computational complexity: the parsing problem for LFGs is NP-complete (Berwick, 1982), and so, in the worst case scenario, computationally intractable (assuming $\mathsf{P} \neq \mathsf{NP}$).

There have been attempts to remedy this situation, however. While the LFG formalism as a whole may be computationally very complex, some of the properties responsible for this are not relevant for the description of natural languages – this opens the possibility that the formalism could be constrained to allow tractable parsing (i.e. in polynomial time) while still preserving its usefulness as a tool for describing natural languages. Seki et al. (1993) propose one such restriction, which limits the kinds of functional annotations permitted on c-structure nodes, and the number of nodes which can correspond to a single f-structure. This successfully buys tractability for the resulting formalism, but at a heavy theoretical cost: many staple aspects of LFG analyses are no longer available, including the very common $\uparrow = \downarrow$ head-sharing annotation, or functional control equations like $(\uparrow \textsc{xcomp subj}) = (\uparrow \textsc{subj})$. More recently, Wedekind and Kaplan (2020) have addressed this limitation, describing a more expressive but still tractable version of the LFG formalism, which is provably equivalent to a Linear Context-Free Rewriting System (LCFRS), and therefore in the mildly context-sensitive space. (See also chapters/Computational and references therein for discussion of the formal and computational properties of LFG.) This approach only covers the original LFG formalism of Kaplan and Bresnan (1982), however, and it remains to be seen whether certain extensions to this basic formalism, such as functional uncertainty (Kaplan, Maxwell, and Zaenen, 1987; Kaplan and Zaenen, 1989), can be accommodated as straightforwardly in this new approach.

One point worth noting is that even in the absence of a tractable version of LFG, this contrast between TAG and LFG should not automatically be viewed as a failing on the part of the latter. In fact, it reflects a rather deep meta-theoretical question: do we want the formalism *itself* to say something interesting about the class of natural languages? The view embodied by TAG (and CCG, MCFG, etc.) is that we should be interested in "finding a grammar formalism that, by itself, gives already a close characterization of the class of natural languages" (Kallmeyer, 2010, p. 7). By contrast, the view embodied by LFG (and HPSG, Minimalism, etc.) is that "it is the theory that imposes the constraints, not the language in which the theory is expressed" (Pollard, 1997, p. 9). In theoretical terms, at least, it does not seem obvious that one approach is *better* than the other – they are merely different perspectives on the problem.[11]

[11] Of course, from a more practical point of view, it matters very much whether a formalism is tractable if it is to be used in some natural language processing task. However, there is already a very successful computational implementation of LFG in the form of the Xerox Linguistic Environment (XLE: Kaplan and Newman, 1997; Crouch et al., 2008), which employs various "packed computation" (Lev, 2007) heuristics to ensure efficient parsing (Maxwell and Kaplan, 1989; Maxwell and Kaplan, 1993; Maxwell and Kaplan, 1996). So whatever limitations may exist in principle, they do not necessarily apply in practice.

5 Lexicalisation

I mentioned at the start of Section 3 that linguistic applications of TAG assume that the grammar is "lexicalised". Abeillé and Rambow (2000, p. 7) give the following definition of this term (emphasis in original):

> We will call a grammar *lexicalised* if every elementary structure is associated with exactly one lexical item (which can consist of several words), and if every lexical item of the language is associated with a finite set of elementary structures in the grammar.

In contrast to (L)TAG, LFG grammars are not in general lexicalised, which is perhaps somewhat surprising given what the "L" in "LFG" stands for – although there is a focus on the lexicon as a richly structured respository of grammatical information, there is no requirement that this information *cannot* be expressed through non-lexical means. This section begins by sketching the potential for lexicalising CFG-based formalisms, like LFG, and then explores what the potential advantages of lexicalised grammars are.

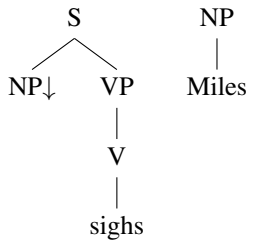
In general, CFGs are not lexicalised. For example, the toy grammar in (3), repeated below, is not lexicalised, since the first and third rules are not associated with any lexical item – they consist purely of non-terminals.

(3)
$$\begin{array}{lcl} \text{S} & \longrightarrow & \text{NP VP} \\ \text{NP} & \longrightarrow & \text{Miles} \\ \text{VP} & \longrightarrow & \text{V} \\ \text{V} & \longrightarrow & \text{sighs} \end{array}$$

Since LFG is based on a CFG, via c-structure, LFG grammars standardly make use of many non-lexicalised rules like these, which means that LFG grammars are generally not lexicalised.

It is possible to convert a non-lexicalised grammar into a lexicalised one, but this can require a change to the formalism used. We can speak of one grammar LEXICALISING another if the former is (weakly or strongly) equivalent to the latter, except that the former is lexicalised whereas the latter is not.[12] For example, the Tree Substitution Grammar shown above in (5), and repeated below, STRONGLY LEXICALISES the grammar in (3), since each elementary object in (5) is associated with a lexical item, and the grammar describes the same string and tree language as (3).

(5)



Sometimes it is possible to use a CFG to strongly lexicalise another CFG, but it turns out that this cannot be guaranteed in principle. For, although there is a way of converting any CFG into so-called GREIBACH NORMAL FORM (Greibach, 1965),

[12] Two (classes of) grammars are weakly equivalent if they describe the same (sets) of string languages (though the corresponding (sets of) tree languages may differ). They are strongly equivalent if they also describe the same (sets) of tree languages.

where the right-hand side of each rule begins with a terminal symbol – thereby lexicalising the grammar – such grammars do not in general generate the same set of trees as the grammars they normalise, since they will include different (and many more) rules. That is, converting a CFG into Greibach normal form only weakly lexicalises it. The extended domain of locality of a TSG/TAG allows us to avoid this problem, however, and makes tree grammars like this "naturally" lexicalised (Schabes, Abeillé, and Joshi, 1988, p. 579). In fact, to strongly lexicalise an arbitrary CFG, we require a TAG, not simply a TSG (see Kallmeyer 2010, pp. 22–23 for a proof). And although a TSG may be sufficient to lexicalise many linguistically relevant CFGs, it places syntactically undesirable restrictions on the resulting grammar, and so a TAG is preferable here too (Schabes, Abeillé, and Joshi, 1988, p. 579; Schabes 1990, ch. 1). But why should we care whether a grammar is lexicalised or not?

One early advantage touted for lexicalised grammars was based on parsing. In a lexicalised grammar, a given sentence can contain at most as many elementary structures as there are words in the sentence. Since each lexical item is associated with a finite number of elementary structures, this also means that the number of analyses for the sentence is finite, thus guaranteeing that the recognition problem is decidable (Schabes, Abeillé, and Joshi, 1988, pp. 581–582). As Kallmeyer (2010, 21, emphasis in original) puts it, "[l]exicalized grammars are *finitely ambiguous*, i.e., no sequence of finite length can be analyzed in an infinite number of ways". However, in practice, the dangers of non-terminating parses are virtually non-existent in sensibly-written natural-language grammars, and so this advantage is not so great as it may seem.[13]

A related claim is that lexicalised grammars assist parsing because "parsing need consider only those trees of the grammar that are associated with the lexical symbols in the input string" (Eisner and Satta, 2000, pp. 79–80), rather than searching the whole grammar, and so the specific words used in a sentence "help to restrict the search space during parsing" (Kallmeyer, 2010, p. 20). Once again, however, this argument carries less practical weight than it might seem, since parsing times for TAG grammars are actually rather slow: the best parsing algorithms for TAGs have a time complexity of $\mathcal{O}(n^6)$, as opposed to $\mathcal{O}(n^3)$ in the case of CFGs, for example (Kallmeyer, 2010, ch. 5).[14]

There are, however, more theoretical reasons to be interested in lexicalised grammars. Firstly, it is by virtue of lexicalisation that the derivation tree of a sentence corresponds to its dependency structure (Kuhlmann, 2010, 4ff.), as discussed in Section 3.4. Because each elementary object in a lexicalised grammar corresponds to a lexical item, by tracking the combination of those objects we are in fact tracking the combination of lexical items. Especially given the recent interest in dependency grammars prompted by the Universal Dependencies project (Nivre et al., 2016), it is clearly advantageous if our formalism has a transparent connection to dependency structures (see also chapters/Dependency on the relationship between LFG and dependency grammars).

Secondly, a lexicalised grammar fits very well with a lexicalist view of syntactic theory. Since the 1970s (at least since the publication of Chomsky 1970), there has been a trend in linguistic theory towards giving lexical analyses of many phenomena which were previously treated as purely syntactic. Indeed, driven by this trend, a plethora of linguistic frameworks have emerged which very deliberately place the lexicon front and centre, treating it as a "richly structured" object, and assuming "an articulated theory

[13] My thanks to Adam Przepiórkowski and Timm Lichte for discussion of this point.

[14] Although this is true of TAGs in general, if our only concern is lexicalising an existing CFG-based grammar, then we could likely devise a parser specialised for TAG grammars that lexicalise CFGs which would have a complexity below $\mathcal{O}(n^6)$. My thanks to a reviewer for this observation.

of complex lexical structure" (Dalrymple, 2001, p. 3) – this includes LFG, as well as (to a greater or lesser extent) Generalized Phrase Structure Grammar (GPSG: Gazdar et al., 1985), Head-Driven Phrase Structure Grammar (HPSG: Pollard and Sag, 1994), Combinatory Categorial Grammar (CCG: Steedman, 2000), Minimalism (Chomsky, 1995; see also chapters/Minimalism), and others.[15] Such a focus on the richness of the lexicon is in stark contrast to the historically more prominent view of it as a mere "collection of the lawless", to use Di Sciullo and Williams's (1987, p. 4) term, where it is simply a repository of exceptions, "incredibly boring by its very nature", about which "there neither can nor should be a theory" (*ibid.*: 3–4). Given that a lexicalist syntactic theory assumes a richly detailed lexicon, in its most parsimonious form this is *all* it would require, the syntactic component being encoded in the lexical entries themselves. In fact, this is just what lexicalisation provides: in TAG, for example, aside from the basic operations of adjunction and substitution, any other grammatical constraints are described in the elementary trees of lexical items; that is, in the lexicon. In a lexicalised grammar, the lexicon essentially *is* the grammar.[16] This means that every language shares the same computational component, and the only differences between languages are in the lexicon (cf. the Borer-Chomsky Conjecture, mentioned above). This is unlike LFG, for example, where languages differ both in their lexica and in the set of c-structure rules they employ.

6 Factoring out redundancies

Natural language grammars involve a large amount of redundancy: for example, the TAG elementary trees for *loves* and *thinks* shown in Table 1 are identical except for their lexical anchors and for the fact that *loves* takes an NP complement where *thinks* takes an S complement. Similarly, all proper nouns will have elementary trees like *Benjamin*, and all VP adverbs will have elementary trees like *really*, except they may follow rather than precede the VP they modify (i.e. the order of the foot node VP* and the AdvP node may be reversed). There is less redundancy when it comes to trees in an LFG grammar, because elementary trees are broken down into smaller-scale phrase-structure rules, but there is plenty of repetition in functional descriptions, where, for example, all 3SG verbs in English will bear the same annotations describing the person and number of their subjects.

Such redundancy or repetition is unavoidable, but it brings with it two undesirable properties: firstly, from a theoretical perspective, it means that certain generalisations may not be expressed; e.g. there is something that *thinks* and *loves* have in common, namely that they require a 3SG subject, and so it is not a mere coincidence that there is overlap in their TAG elementary trees or in their LFG functional descriptions. But nowhere in either grammar is this generalisation expressed *qua* generalisation. Secondly, from a grammar engineering perspective, this kind of redundancy makes updat-

[15] To the extent that Minimalism is truly minimal, i.e. the only syntactic operation is Merge, with constraints being imposed by lexical features, then it is not only lexicalist but also lexicalised (though the use of various "empty" heads arguably still disqualifies it). However, it is also common to assume a number of other operations, such as Agree, which can be sensitive to emergent structural properties not specified in lexical entries, which then undermines the framework's lexicalised status.

[16] Note that it is possible to collapse the lexicon/grammar distinction without also collapsing the word/phrase (or, equivalently, morphology/syntax) distinction: the processes which build word forms, i.e. the leaf nodes of elementary trees in TAG, need not be the same as those which build derived trees in the syntax. Thus, the formal language theory objections to Construction Grammar presented by Asudeh, Dalrymple, and Toivonen (2013, pp. 4–5) are only objections to the most radical version of the theory, and need not be taken as objections to constructional approaches generally.

ing and extending grammars very difficult: if we change how we analyse a particular phenomenon, we have to make sure we change every instance of it in the grammar (e.g. change every transitive elementary tree); and if we introduce a new feature to handle some new phenomenon, we have to make sure it is handled correctly in all the existing structures, by manually adapting them one by one. This is clearly likely to lead to inconsistencies and inaccuracies due to human error.

It is therefore desirable to find a means of factoring out redundancies from a grammar, and expressing the generalisations they capture in a single place. Both LFG and TAG have means of achieving this. In TAG, it is common practice to make use of a METAGRAMMAR, essentially a grammar responsible for generating grammars, where such redundancies can be described just once. Candito (1996,1999) was the first to develop such a metagrammar; her version describes elementary trees along three dimensions: 1) subcategorisation (i.e. how many arguments a verb selects for), including the canonical syntactic functions of the subcategorised arguments; 2) valency alternations/redistribution of syntactic functions; i.e. the actual syntactic function of the arguments; 3) the surface syntactic manifestation of these functions. Each of these dimensions is described by an inheritance hierarchy, and the classes of the metagrammar, corresponding to specific linguistic constructions, such as the English *by*-passive, inherit from one of the terminal classes in the first dimension, one of the terminal classes in the second dimension, and as many of the terminal classes in the third dimension as there are arguments to realise. Constructions can therefore be described by listing the terminal classes they inherit from each of the three dimensions, a label which Kinyon (2000) calls a HYPERTAG (following from the notion of "supertag" introduced by Bangalore 1997 – see also Bangalore and Joshi 2010).

The most recent implementation of the concept of metagrammar is the EXTENSIBLE METAGRAMMAR (XMG) of Crabbé et al. (2013). This does away with Candito's three explicit dimensions, and instead employs a highly expressive description language that enables linguistic structures to be given a single, complex description, including multiple levels of representation (e.g. syntax and semantics). It is also designed so that it can be extended to cover new phenomena or linguistic formalisms, and so fewer theoretical assumptions are baked into the formalism. XMG makes use of an inheritance hierarchy, but a single hierarchy instead of Candito's three: rather than taking the approach of describing default syntactic function assignments (dimension 1) and then overwriting them with specific valency frames (dimension 2), which might vary, e.g. in the case of diathesis alternations, XMG makes heavy use of disjunctions between alternating descriptions, which enables such alternations to be described fully declaratively, and in just one place. For example, we can express the familiar active-passive diathesis of English as in (32), where each term in italics refers to a class in the metagrammar's inheritance hierarchy that gives a partial description of a (sub-)tree (Crabbé et al., 2013, p. 616).

(32) $$\begin{aligned} \textit{TransitiveDiathesis} \rightarrow \quad & (\textit{Subject} \wedge \textit{ActiveVerbForm} \wedge \textit{Object}) \\ & \vee (\textit{Subject} \wedge \textit{PassiveVerbForm} \wedge \textit{ByObject}) \\ & \vee (\textit{Subject} \wedge \textit{PassiveVerbForm}) \end{aligned}$$

Each of the disjuncts in (32) combines these descriptions to give a partial description of a full elementary tree schema (an elementary tree minus its lexical anchor). For now I leave aside the details of how these classes actually describe trees; a simplified version of the logical description language employed will be introduced in the next section. The crucial observation here, and the move which sets XMG apart from earlier approaches to metagrammatical analysis, is that the description in (32) does not privilege

one elementary tree/realisation of arguments as basic, but simply describes all possible realisations simultaneously.

The terminal classes of the metagrammar are families of trees which are then associated with lemmas, and represent all the different ways of realising that lemma's arguments (e.g. active vs. passive, *wh*-extraction, clefting, etc.). The *TransitiveFamily* associated with a lemma like LOVE might just consist of (32), while the *DitransitiveFamily* of a verb like GIVE might inherit from the *TransitiveFamily* but add an additional object argument:

(33) $\begin{array}{lcl} \textit{TransitiveFamily} & \rightarrow & \textit{TransitiveDiathesis} \\ \textit{DitransitiveFamily} & \rightarrow & \textit{TransitiveDiathesis} \wedge \textit{IndirectObject} \end{array}$

This modular and structured approach to the metagrammar means that, for instance, if the analysis of a particular phenomenon changes, we just need to modify the relevant class(es): when the grammar is compiled anew, all of the implicated elementary trees will be altered accordingly. The choice of classes can also have theoretical implications, and may shed light on important linguistic generalisations.

Although the metagrammatical approach has been used to generate LFG grammars as well as TAGs (Clément and Kinyon, 2003a; Clément and Kinyon, 2003b), this is not common practice in LFG work. Rather, since redundancies in an LFG grammar are far more abundant in the functional descriptions associated with lexical entries than in phrase structure, the standard solution employed here is to make use of TEMPLATES, a type of macro which can be used to abbreviate pieces of functional description that are re-used across lexical entries (Dalrymple, Kaplan, and King, 2004; Crouch et al., 2008; see also [sec. 5.1]chapters/CoreConcepts). These templates can take arguments, and can also call other templates, creating a hierarchical organisation – though it should be noted that this is an *inclusion* hierarchy rather than an inheritance hierarchy, since template calls can be negated (Asudeh, Dalrymple, and Toivonen, 2013, 18f.). The semantics of template invocation (represented by prefixing the template name with a '@' symbol) is substitution: the template name is replaced by its contents. This means that a grammar without templates is extensionally equivalent to one with them, but in the latter it will be possible to express generalisations that cannot be expressed in the former.

By way of illustration, (34–35) present some templates which capture some of the same information present in the XMG classes shown above. The TRANSITIVEDIATHESIS template takes a predicate name as its argument, and consists of a disjunction of three other templates; it will be called by the lexical entry of any transitive verb which participates in the active/passive alternation in English. Each of the three templates it invokes provides a PRED value for the verb in question, associating it with the correct set of grammatical functions, and also provides mapping equations which link the GFs to argument positions at semantic structure, or express the fact that the argument is not syntactically realised, in the case of the short passive (this approach to mapping is described in Asudeh and Giorgolo 2012 and Findlay 2016; see also chapters/Mapping).

(34) $\begin{array}{l} \textsc{TransitiveDiathesis}(P) \equiv \\ @\textsc{ActiveTransitive}(P) \vee @\textsc{ByPassive}(P) \vee @\textsc{ShortPassive}(P) \end{array}$

(35) a. $\textsc{ActiveTransitive}(P) \equiv \begin{array}{l} (\uparrow \textsc{pred}) = \text{`}P\langle \textsc{subj}, \textsc{obj} \rangle\text{'} \\ (\uparrow_\sigma \textsc{arg}1) = (\uparrow \textsc{subj})_\sigma \\ (\uparrow_\sigma \textsc{arg}2) = (\uparrow \textsc{obj})_\sigma \end{array}$

b. $\textsc{ByPassive}(P) \equiv (\uparrow \textsc{pred}) = \text{`}P\langle \textsc{subj}, \textsc{obl}_{\textsc{by}} \rangle\text{'}$
$(\uparrow_\sigma \textsc{arg}1) = (\uparrow \textsc{obl}_{\textsc{by}})_\sigma$
$(\uparrow_\sigma \textsc{arg}2) = (\uparrow \textsc{subj})_\sigma$

c. $\textsc{ShortPassive}(P) \equiv (\uparrow \textsc{pred}) = \text{`}P\langle \textsc{subj} \rangle\text{'}$
$(\uparrow_\sigma \textsc{arg}1)_{\sigma^{-1}} = \varnothing$
$(\uparrow_\sigma \textsc{arg}2) = (\uparrow \textsc{subj})_\sigma$

One noteworthy difference between the use of a metagrammar and the use of templates is that the latter but not the former are part of a grammar itself. A metagrammar, as the name suggests, sits outside the grammar proper: it outputs grammars, where the elementary objects do not (necessarily) contain information about which metagrammar classes they instantiate. Templates, on the other hand, are part of the description language of the grammar, although of course they are merely names for pieces of functional description, and so have no special formal status themselves.

7 Combining LFG and TAG

Now that we have seen some of the key concepts of TAG, along with their motivations and apparent benefits, we might wonder whether LFG could also benefit from some of these boons if we were to combine the two approaches – most naturally, by using a TAG instead of a CFG to describe LFG's c-structure. Joshi (2005, p. 496) described this idea as being "of great interest", and it was previously explored by Kameyama (1986) and Burheim (1996) – but unfortunately only in unpublished work, which has proved impossible to track down. More recently, the idea has been revived by Findlay (2017a), Findlay (2017b), and Findlay (2019). In this section, I outline two different approaches to achieving the goal of combining LFG and TAG, and discuss some of the consequences of adopting such a merger.[17]

7.1 Two approaches

The most straightforward way of combining TAG and LFG is simply to take a TAG grammar and add appropriate LFG annotations to the elementary trees. Of course, once we have access to the whole tree, we gain a greater degree of flexibility in how we express functional annotations. Most notably, we can refer to any node in the tree directly, rather than being limited to the current node or its mother – a consequence of TAG's extended domain of locality. For example, instead of relying on a sequence of $\uparrow = \downarrow$ annotations to pass information from a lexical item to the top of its extended projection, we can refer to the top directly. For the sake of simplicity, let us use node labels as shorthand for the nodes themselves.[18] Then the $(\uparrow \textsc{pred}) =$ 'love' annotation on the verb *loves*, for example, could be rewritten as $(\mathrm{S}_\phi \textsc{pred}) =$ 'love', using S_ϕ to refer to the f-structure of the clause directly, rather than indirectly via V_ϕ (the instantiation of $\uparrow$), which is equated with both VP_ϕ and S_ϕ. Indeed, since we can use absolute rather than relative labels for the nodes in the tree, there is no need to mark annotations

[17] One concern about replacing the CFG component of LFG with a more powerful TAG might be that it makes the formalism as a whole more computationally complex. However, since TAGs are strictly less powerful than LFGs (see Section 4), such a concern is ultimately baseless.

[18] Of course, in reality nodes and their labels are distinct: several nodes can bear the same label, for example (e.g. there can be more than one NP in a tree). When this happens, I follow the TAG convention of suffixing node labels with numbers (e.g. NP1 and NP2), but it should be borne in mind that this is just a representational choice, and that in reality such nodes have identical labels.

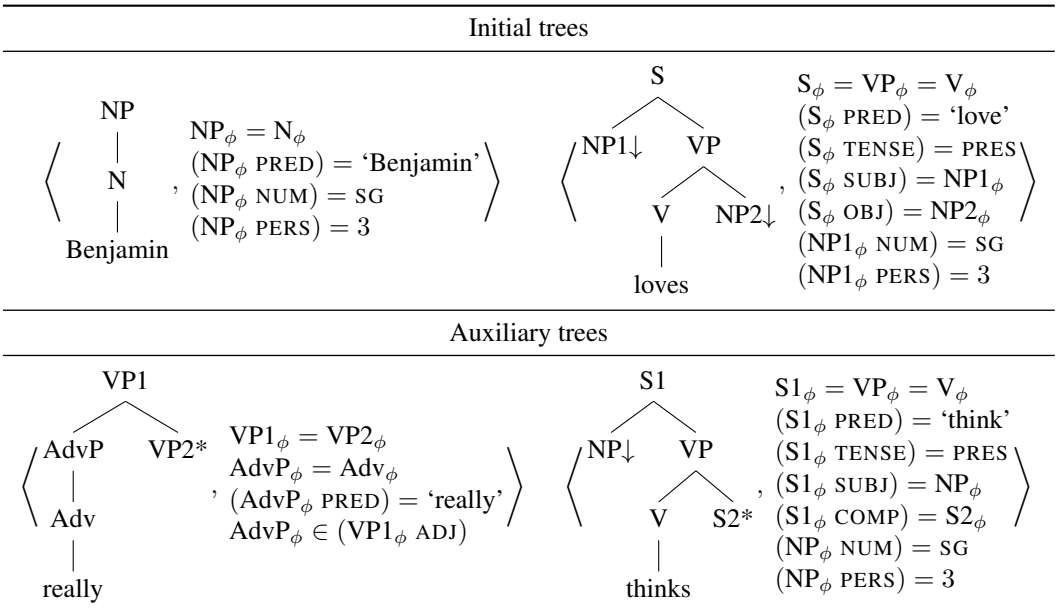


Table 2: Some elementary trees with associated LFG annotations

actually on the tree at all; instead, we can treat lexical entries as pairs consisting of the tree on the one hand and the annotations on the other, which refer to nodes in the tree. This arguably simplifies the process of determining an f-structure from an annotated c-structure, since many identities which would normally have to be computed are instead already given in the descriptions. Table 2 shows the elementary trees from Table 1 augmented in this fashion. The trees then combine as usual for a TAG, using the operations of substitution and adjunction, albeit understood in a particular fashion. Substitution involves identifying two nodes, so that, e.g. if the tree for *Benjamin* were substituted into the subject position of the tree for *loves*, NP and NP1 would be identified (and therefore so would their f-structures, requiring the nodes to bear compatible annotations – and thereby accounting for the agreement facts, for example). Adjunction involves three steps: first we excise a sub-tree rooted at the adjunction site; next, we *replace* it with the adjoining auxiliary tree; finally, we unify the foot node of the auxiliary tree with the root node of the excised sub-tree it replaced. This way, we identify the target of adjunction with the foot of the auxiliary tree, and correctly distribute the annotations between the two "parts" of the expanded node without the need for top and bottom feature structures.[19]

[19]Findlay (2017a, 222, fn. 12) claims that we are forced to adopt the second proposal to be discussed below, using descriptions of trees, because adjunction means that the ↑ and ↓ chains in annotations will be disrupted. This would be true if we were forced to refer to f-structures only indirectly, via mother-daughter links, but fails to appreciate the additional freedom afforded by being able to refer to nodes absolutely, as discussed above. There is, however, a small wrinkle when it comes to verbal trees for extraction constructions (e.g. *wh*-questions): if nothing is adjoined to them, we want to unify the f-structures of the root S and the S node it immediately dominates; but if a sentential embedding verb is adjoined there, we cannot identify the two f-structures, or else we will end up with a cyclic f-structure which is its own COMP. All this shows us though is that we have to take care when writing the functional annotations. Here, for example, we can solve the problem by actually reintroducing an element of relativity: we identify the root node's f-structure with the f-structure of its S daughter (e.g. by defining a predicate CATDAUGHTER(n, C) which identifies the unique daughter of node n which bears label C, and is undefined if there is none or more than one),

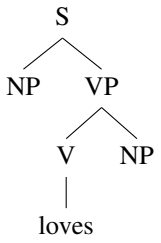
This first approach is much more in the spirit of TAG than of LFG, since the c-structure component is DERIVATIONAL, making use of the combining operations of substitution and adjunction. Let us therefore call it LFG-TAG. But as Kaplan (1995, p. 11) points out, this PROCEDURAL, or CONSTRUCTIVE, approach to grammatical analysis is in contrast to the DESCRIPTIVE (a.k.a. DECLARATIVE or MODEL-BASED) approach which is the "hallmark of LFG" (*ibid.*). Findlay (2019, ch. 5) therefore explores another way of combining the two frameworks which is more in keeping with the LFG spirit. In brief, we associate lexical entries with *descriptions* of trees, rather than with the trees directly, as is standard practice in metagrammars, for instance.[20] In the simplest cases there is a one-to-one correspondence between a description and the (minimal) tree it describes, and so we could straightforwardly translate LFG-TAG into a more LFG-like format. However, descriptions can also make use of negation, disjunction, or other operations that go beyond simple conjunction of propositions, and in this case the relation between descriptions and trees is no longer isomorphic (Kaplan, 1995, p. 14).

In order to add descriptions of trees to LFG lexical entries, we need a suitable language to write the descriptions in. There are a variety of different possibilities, but here we will assume a fairly simple language based on that used in XMG (Crabbé et al., 2013, p. 599), which will consist of the following:[21]

(36)
1. a set N of node variables
2. a set P of unary labelling predicates, including all terminal and non-terminal labels
3. the following binary predicates:
   - $\rightarrow$, immediate dominance (the *mother-of* relation)
   - $\rightarrow^*$, dominance (the transitive, reflexive closure of $\rightarrow$)
   - $\prec$, linear precedence[22]

The tree in (37) can then be described by the set of constraints in (38):[23]

(37)



regardless of which node that actually ends up being. I omit the formal details of how this can be achieved, since ultimately we will settle on the second approach to integrating TAG and LFG described below, but it is important to note that this first approach is not unworkable.

[20]The use of tree descriptions has been discussed extensively in the context of TAG – see, for instance, Vijay-Shanker (1992), Rogers and Vijay-Shanker (1994), Rambow, Vijay-Shanker, and Weir (1995), Rambow, Vijay-Shanker, and Weir (2001), and Kallmeyer (2001).

[21]We will also assume that sufficient axioms are in place to ensure the usual well-formedness conditions on trees, e.g. that they are singularly rooted, that branches cannot cross, etc. Rogers (1998, pp. 15–16) gives one such set of axioms.

[22]Here this is to be understood as the transitive closure of immediate linear precedence, i.e. what Crabbé et al. (2013, p. 599) represent as $\prec^+$. In other words, a node linearly precedes everything to its right, but does not linearly precede itself.

[23]In descriptions, we will assume that all node variables are ultimately existentially bound.

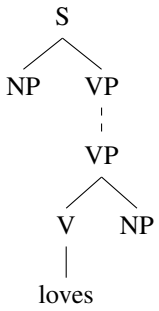
(38)

$\text{S}(n_1)$	$n_1 \rightarrow n_2$	$n_2 \prec n_3$
$\text{NP}(n_2)$	$n_1 \rightarrow n_3$	$n_4 \prec n_5$
$\text{VP}(n_3)$	$n_3 \rightarrow n_4$	
$\text{V}(n_4)$	$n_3 \rightarrow n_5$	
$\text{NP}(n_5)$	$n_4 \rightarrow n_6$	
$\text{loves}(n_6)$		

However, as it stands, the constraints in (38) are too rigid. Specifically, they will not allow adjunction at the VP node, since then at least one statement in the description will no longer be true: if we identify the target n_3 with the root of the adjoining tree, then $n_3 \rightarrow n_4$ will no longer hold (the foot node of the auxiliary tree will dominate n_4 instead), and if we identify it with the foot node of the adjoining tree, then $n_1 \rightarrow n_3$ will not be true instead. The basic problem is that "[t]he composition operation of adjoining creates a new structure that does not maintain all of the properties that held in the original (fully specified) structures of which it is composed" (Vijay-Shanker, 1992, p. 486). What this means is that we cannot operate with fully specified descriptions, but must make use of partial descriptions instead.

For each node where adjunction can apply, we instead describe a pair of QUASI-NODES which stand in the dominance relation (Vijay-Shanker, 1992, 486ff.). That is, instead of (38), we have (40), which is represented schematically in (39) (where a dashed line represents dominance rather than immediate dominance):

(39)



(40)

$\text{S}(n_1)$	$n_1 \rightarrow n_2$	$n_2 \prec n_3$
$\text{NP}(n_2)$	$n_1 \rightarrow n_3$	$n_5 \prec n_6$
$\text{VP}(n_3)$	$n_3 \rightarrow^* n_4$	
$\text{VP}(n_4)$	$n_4 \rightarrow n_5$	
$\text{V}(n_5)$	$n_4 \rightarrow n_6$	
$\text{NP}(n_6)$	$n_5 \rightarrow n_7$	
$\text{loves}(n_7)$		

As elsewhere in LFG, we take the solution to a set of constraints to be the *minimal* structure (or structures) which satisfies all the constraints. Since the dominance relation is reflexive, the minimal tree which satisfies (40) remains (37), i.e. one where we equate n_3 and n_4. But, crucially, if something is adjoined here, the nodes can come apart, with the result that n_3 is identified with the root of the auxiliary tree and n_4 with its foot node.

Now that we have a description of this tree, we can combine it with functional descriptions to form a full LFG lexical entry:[24]

[24]Here I have kept to a more conservative annotation scheme than above, whereby e.g. lexical contribu-

(41)

$\text{S}(n_1)$	$n_1 \rightarrow n_2$	$n_2 \prec n_3$	$n_{1\phi} = n_{3\phi}$
$\text{NP}(n_2)$	$n_1 \rightarrow n_3$	$n_5 \prec n_6$	$n_{4\phi} = n_{5\phi}$
$\text{VP}(n_3)$	$n_3 \rightarrow^* n_4$		$(n_{5\phi}\ \text{PRED}) =$ 'love'
$\text{VP}(n_4)$	$n_4 \rightarrow n_5$		$(n_{5\phi}\ \text{TENSE}) = \text{PRES}$
$\text{V}(n_5)$	$n_4 \rightarrow n_6$		$(n_{1\phi}\ \text{SUBJ}) = n_{2\phi}$
$\text{NP}(n_6)$	$n_5 \rightarrow n_7$		$(n_{4\phi}\ \text{OBJ}) = n_{6\phi}$
$\text{loves}(n_7)$			$(n_{2\phi}\ \text{NUM}) = \text{SG}$
			$(n_{2\phi}\ \text{PERS}) = 3$

To parse a sentence, we just collect up all of the constraints associated with each lexical item and find the minimal structures – both c-structure and f-structure – which satisfy them.

Of course, (41) is not particularly readable, so we might prefer to collect some parts of the description in various templates. For example, the tree for any transitive verb will share most of the description in (41), so we can factor out this part of the description, parametrising the only variable, namely the lexical anchor:

(42) $\text{TRANSITIVETREE}(s, np_1, vp_1, vp_2, v, np_2, a, \textit{anchor}) \equiv$

$\text{S}(s)$	$s \rightarrow np_1$	$np_1 \prec vp_1$	$s_\phi = vp_{1\phi}$
$\text{NP}(np_1)$	$s \rightarrow vp_1$	$v \prec np_2$	$vp_{2\phi} = v_\phi$
$\text{VP}(vp_1)$	$vp_1 \rightarrow^* vp_2$		$(s_\phi\ \text{SUBJ}) = np_{1\phi}$
$\text{VP}(vp_2)$	$vp_2 \rightarrow v$		$(vp_{2\phi}\ \text{OBJ}) = np_{2\phi}$
$\text{V}(v)$	$vp_2 \rightarrow np_2$		
$\text{NP}(np_2)$	$v \rightarrow a$		
$\textit{anchor}(a)$			

In fact, we have to "expose" all nodes as parameters of the template, so that they can be referred to by other constraints in the same lexical entry, thereby taking advantage of the extended domain of locality afforded by having the description of the whole tree in one place. However, since all of the parameters in (42) except the lexical anchor will simply be node variables, I propose a shorthand: when calling the template, all but the last parameter will be omitted (though, to repeat, when defining it all the parameters must be specified); if we wish to refer to any of the other parameters, we can do so by using the template name and suffixing it with the appropriate parameter.[25] For example, TRANSITIVETREE.s refers to the first parameter, a node variable which corresponds to the root node s in (42). With these conventions in place, we can write a more readable lexical entry for *loves* as follows (using a LOCAL NAME, %UP, to refer to the verb's f-structure – see [sec. 3.2.5]chapters/CoreConcepts for the details on local names):[26]

tions are associated with the f-structure of n_5, i.e. the V node, rather than with that of the root S. This is because adjunction may in principle alter the structure of the tree so that it is no longer the case that the f-structure of the S node is the same as the f-structure of the V node. In fact, with verbal trees like this, that will not be the case, because the only auxiliary trees which target VPs in a TAG grammar will be auxiliary verbs or adverbial modifiers, neither of which will break the link between V and S in terms of f-structure-identity. But it will be relevant for trees containing extraction sites, for example, which can be targetted by sentential embedding verbs, and where the root's f-structure will thereby be separated from the head verb's.

[25]This is based on the conventions of XMG for exported variables (Crabbé et al., 2013, pp. 602–604).

[26]Here I have reverted to describing the agreement constraints on the subject via the clause's f-structure rather than via the NP's, to make this lexical entry closer to the LFG standard. But of course the option is still open to us to describe it via the tree directly, by associating TRANSITIVETREE.$np_{1\phi}$ with a name, e.g. %SUBJ-NP, and then declaring that (%SUBJ-NP NUM) = SG. Although these options are extensionally equivalent, there can be theoretical/descriptive reasons to prefer one over the other. Cross-linguistically, for example, we might want to treat subject agreement as the same kind of phenomenon both in languages where phrase-structure position is a clear guide to grammatical function (like English) and in languages where it is

(43) @TRANSITIVETREE(loves)
$\%$UP $=$ TRANSITIVETREE.v_ϕ
($\%$UP PRED) $=$ 'love'
($\%$UP TENSE) $=$ PRES
($\%$UP SUBJ NUM) $=$ SG
($\%$UP SUBJ PERS) $= 3$

One theoretical advantage of this approach is that we can build up trees from smaller parts by making use of nested template calls. This allows us to capture connections between phrasal configurations in a way which CFG rules do not. For example, there is no relationship between the two rules in (44), even though the latter is obviously partially described by the former:[27]

(44) a. VP $\longrightarrow$ V NP
$\uparrow = \downarrow$ $(\uparrow \text{OBJ}) = \downarrow$

b. VP $\longrightarrow$ V NP NP
$\uparrow = \downarrow$ $(\uparrow \text{OBJ}) = \downarrow$ $(\uparrow \text{OBJ}_\theta) = \downarrow$

On the other hand, if we have a template DITRANSITIVETREE which calls the TRANSITIVETREE template (as well as another template which adds a secondary object), then this containment relationship is made explicit:

(45) DITRANSITIVETREE(*anchor*) $\equiv$ @TRANSITIVETREE(*anchor*)
@SECONDARYOBJECT

Of course the TRANSITIVETREE template can also be decomposed into a call of an INTRANSITIVETREE template plus a PRIMARYOBJECT one, and so on. By continuing along these lines, we can capture all the various generalities across trees in a template inclusion hierarchy, recreating the class hierarchies of a TAG metagrammar inside an LFG grammar itself.

7.2 Implications

Having now seen how LFG and TAG can be combined, let us consider the consequences of such a merger. There are several potential gains which such a move could bring, along with several unanswered questions which require further research.

Firstly, the second approach described above offers a pleasing harmonisation of LFG lexical entries. Standard LFG lexical entries contain descriptions of all levels of the projection architecture, but since such lexical entries are really just context-free phrase-structure rules, the description of c-structure is limited to information about the

not (like Warlpiri); so it would make sense to retain the standard LFG approach of describing agreement via f-structure. But in other cases it might make more sense to refer to a particular phrase-structure position, and the integration of an extended tree description into the lexical entry means we now have that choice.

[27] Of course, we can use the convention of surrounding optional nodes in parentheses, and then we can express the relationship between the two rules within a single phrase-structure rule as follows:

(i) VP $\longrightarrow$ V NP $\left(\text{NP}\right)$
$\uparrow = \downarrow$ $(\uparrow \text{OBJ}) = \downarrow$ $(\uparrow \text{OBJ}_\theta) = \downarrow$

However, once we move beyond simple examples like this, such an approach becomes unwieldy, with multiply nested parentheses and very complex disjunctions. Unlike the templatic approach, which provides a readable front-end to the formal complexity, and allows us to represent the relationship(s) between sub-trees in an inclusion hierarchy, this approach forces us to create fewer but more complex rules, which does nothing to aid human-readability.

word itself and its mother. In contrast, descriptions of all other levels of structure can refer to arbitrarily distant elements (via functional uncertainty). The inclusion of tree descriptions removes this irregularity from the grammar, since now non-local elements of c-structure can also be included.

Secondly, we now have the opportunity to lexicalise an LFG grammar (indeed, Findlay 2019, ch. 5 calls the description-based approach described above "Lexicalised LFG"). As outlined above, especially in Section 5, the extended domain of locality of a TAG means that all dependencies, including long-distance ones, can be encoded locally in a lexical entry. Lexicalisation seems a natural goal for a lexicalist theory like LFG, and it is perhaps lamentable that it was not possible before.

Thirdly, we can now straightforwardly account for idioms (Findlay, 2019, ch. 6). These are problematic for the current leading account of constructions in LFG (Asudeh, Dalrymple, and Toivonen, 2013), since they do not simply add additional constructional meaning to existing lexical meaning, but rather *replace* the lexical meaning with another, different meaning (that is, *shooting the breeze* involves neither shooting nor a uniquely contextually salient breeze). This forces lexicalist theories like LFG to adopt an approach which treats idioms as conspiracies of independent lexical items that select for one another. Such approaches face a host of problems, not least of which is that they singularly fail to capture our intuitions about idioms – *viz.* that they are "things" (as Williams 2007 puts it), and not mere epiphenomena of the grammar (see Findlay 2019, 58ff. for discussion of various other problems). But now that we can have lexical entries containing multiple, separable word forms, something which is not possible in vanilla LFG, there is no obstacle to encoding multiword expressions straigthforwardly in a single place, thus enabling a much more satisfying analysis. Findlay (2019, ch. 3) provides detailed discussion of the need for this kind of constructional approach to idioms, as well as arguments against other types of analysis.

Fourthly, as well as idioms, we have a straightforward account of constructional phenomena more broadly. Similar arguments can be made here about the need for constructions to have some ontological status in the theory – to be "things".[28] Admittedly, Asudeh, Dalrymple, and Toivonen (2013) demonstrate that we do not need to admit constructions as first-class entities in our theory in order to explain *some* kinds of constructional effects, but they only consider constructions which can be described by a single lexical entry or a single context-free phrase-structure rule, and so the constructions in question can be described in a single place. Other constructions require reference to wider spans of phrase structure, or require the presence of specific words in quite distant parts of the phrase, and here the approach will once again have to rely on multiple interacting lexical entries and phrase-structure rules which conspire to give the correct constructional effects. Even if this gives the right results, one might, again, object that it does so for the wrong reasons, since it fails to account for the unitary nature of constructions as grammatical objects. By contrast, in the description-based approach to merging TAG and LFG, although constructions are still not added as new objects in the ontology of the theory, they nevertheless have a kind of first-class status, since they can either be entire (complex) lexical entries or be a part of a lexical entry in the form of a tree template which can be called by all the different words which can

[28]There is suggestive psycho- and neurolinguistic evidence that the way we process language makes heavy use of prefabricated chunks ('prefabs') (Pawley and Syder, 1983; Wray, 2002) and of constructions more generally (Bencini and Goldberg, 2000; Kaschak and Glenberg, 2000; Goldwater and Markman, 2009; Pulvermüller, 2010; Allen et al., 2012; Johnson and Goldberg, 2012). Obviously grammatical theory need not have anything to say about how language is processed in the mind, but it might still be seen as an advantage if it at least makes available the kinds of objects the mind seems to work with – e.g. constructions.

fill the empty slots in the construction.

As well as these advantages, there remain some unexplored implications which are ripe for future work. Firstly, one of the parade examples of LFG's utility is in describing languages with highly flexible word orders, such as Warlpiri (see e.g. Bresnan, Asudeh, et al., 2016, ch. 1). With TAG's focus on configurational properties, we need to ensure that incorporating a TAG into LFG does not undo its ability to describe these non-configurational languages. Given the flatter tree structures generally assumed for such languages (Simpson, 1991; Austin and Bresnan, 1996), a first pass solution in the present framework would be to simply make use of looser tree descriptions, which, for example, lack linear precedence relations between a verb and its arguments, so that the the entry for a verb does not describe a unique minimal tree, but rather several minimal trees which represent the different orderings of arguments. Of course, these different orderings are not just random, and actually correspond to different information structures, so simply allowing free choice between them is inadequate. Instead, we should once again make use of disjunction, this time between the descriptions corresponding to the different orderings of verb and arguments, where each of the different word orders is also accompanied by the correct information-structural annotations.

Such languages also often permit discontinuous constituents, and these will require their own solution. For example, some adjuncts might be represented not as auxiliary trees that induce a more articulated structure, but rather as simpler trees whose root merely unifies with another node, such as the root S, adding the adjunct as a sister to the existing daughters. Obviously this sketch needs to be developed into a fully fleshed-out proposal before we can be confident that no analytical clout has been lost.

Another open question arises from the fact that using a TAG as the basis of the c-structure component means that we can employ adjunction to account for long-distance dependencies. This then removes a foundational motivation for functional uncertainty (Kaplan and Zaenen, 1989), one of the major sources of formal complexity in LFG. Unfortunately, this does not mean we can simply remove functional uncertainty from the formalism, since it has been employed by researchers in various other domains beyond filler-gap dependencies – most notably in LFG's binding theory (e.g. Dalrymple, 1993; Dalrymple, Haug, and Lowe, 2018). Determining whether these analyses can be reformulated so that functional uncertainty could be done away with altogether remains a task for future work, perhaps drawing on existing TAG analyses of binding (e.g. Ryant and Scheffler, 2006; Champollion, 2008; Storoshenko, Han, and Potter, 2008; Storoshenko and Han, 2013).

Lastly, including a description of a tree which incorporates the full extended projection of a predicate in its lexical entry means that we can take a rather different view of argument structure. A predicate's arguments and the possibilities for their realisation can be encoded directly in its lexical entry, rather than relying on a separate level of representation like a-structure ([on which see][]chapters/Mapping). And alternative argument realisations, e.g. diathesis alternations, can be expressed through disjunctive templates, as discussed above in parallel with XMG, rather than through a separate mechanism of mapping between a-structure and f-structure. Work on developing templatic approaches to argument structure include Asudeh and Giorgolo (2012), Findlay (2016), Findlay (2020), and Przepiórkowski (2017), but these do not take c-structure into account: with the new TAG perspective, the phrase-structural effects of argument structure/mapping phenomena can also be directly expressed.

8 Conclusion

Tree-Adjoining Grammar offers a rather different perspective on some grammatical phenomena from that of CFG-based formalisms.[29] For instance, it allows us to describe constraints on filler-gap relationships via the structure of the elementary trees in the grammar rather than via an independent principle like Subjacency. It also provides a natural account of the fact that many "lexical" items in fact incorporate several distinct word forms (e.g. phrasal verbs, compounds, idioms), and of constructional meaning, by virtue of its expanded concept of locality. And, computationally speaking, it possesses just the right degree of context-sensitivity to account for natural languages while remaining parsable in polynomial time. Nonetheless, its representation of dependency structures is imperfect, and its focus on the primacy of phrase structure leaves it somewhat impoverished when compared to the richly expressive parallel projection architecture of LFG, which facilitates a much fuller view of the grammar as a whole. Combining the two approaches might therefore offer a tempting opportunity to acquire the best of both worlds. In Section 7 we saw how this could be achieved, and the possibilities this affords for LFG, both in terms of descriptive power and in terms of potentially further-reaching formal or architectural changes.

Acknowledgements

Much of this work stems from my D.Phil. thesis (Findlay, 2019), which was funded by a UK Arts and Humanities Research Council studentship (grant reference AH/L503885/1), and for which I am indebted to my fantastic supervisors, Ash Asudeh and Mary Dalrymple. The writing of this chapter was completed while I was employed under a Norwegian Research Council grant (number 300495, "Universal Natural Language Understanding"), which I gratefully acknowledge.

References

[1] Anne Abeillé. "Parsing French with Tree Adjoining Grammar: Some Linguistic Accounts". In: *COLING '88: Proceedings of the 12th Conference on Computational Linguistics*. Budapest, 1988, pp. 7–12. DOI: 10.3115/991635.991637.

[2] Anne Abeillé. "The flexibility of French idioms: a representation with Lexicalized Tree Adjoining Grammar". In: *Idioms: structural and psychological perspectives*. Ed. by Martin Everaert et al. Hove: Lawrence Erlbaum, 1995.

[3] Anne Abeillé and Owen Rambow. "Tree Adjoining Grammar: An overview". In: *Tree Adjoining Grammars: Formalisms, linguistic analysis and processing*. Ed. by Anne Abeillé and Owen Rambow. Stanford, CA: CSLI Publications, 2000, pp. 1–68.

[4] Anne Abeillé, Yves Schabes, and Aravind K. Joshi. "Using Lexicalized TAGs for machine translation". In: *Proceedings of the 13th conference on Computational linguistics (COLING '90)*. Vol. 3. 1990, pp. 1–6. DOI: 10.3115/991146.991147.

[29] TAG has also played an important role outside of theoretical linguistics – specifically in both computational linguistics (see e.g. Kallmeyer, Lichte, et al., 2008; Kasai et al., 2017; Koller, 2017) and psycholinguistics (see e.g. Ferreira, 2000; Ferreira, Lau, and Bailey, 2004).

[5] Kachina Allen et al. “Distinguishing grammatical constructions with fMRI pattern analysis”. In: *Brain and Language* 123.3 (2012), pp. 174–182. DOI: 10.1016/j.bandl.2012.08.005.

[6] Ash Asudeh. *The Logic of Pronominal Resumption*. Oxford Studies in Theoretical Linguistics. Oxford: Oxford University Press, 2012. DOI: 10.1093/acprof:oso/9780199206421.001.0001.

[7] Ash Asudeh, Mary Dalrymple, and Ida Toivonen. “Constructions with Lexical Integrity”. In: *Journal of Language Modelling* 1.1 (2013), pp. 1–54. DOI: 10.15398/jlm.v1i1.56.

[8] Ash Asudeh and Gianluca Giorgolo. “Flexible Composition for Optional and Derived Arguments”. In: *Proceedings of the LFG ’12 Conference*. Ed. by Miriam Butt and Tracy Holloway King. Stanford, CA: CSLI Publications, 2012, pp. 64–84.

[9] Peter K. Austin and Joan Bresnan. “Non-configurationality in Australian Aboriginal Languages”. In: *Natural Language & Linguistic Theory* 14 (1996), pp. 215–268. DOI: 10.1007/bf00133684.

[10] Emmon W. Bach. “An extension of classical transformational grammar”. In: *Problems in linguistic metatheory: proceedings of the 1976 conference*. Ed. by Jerrold M. Sadock et al. East Lansing, MI: Michigan State University, 1976, pp. 183–224.

[11] Emmon W. Bach. “Discontinuous constituents in generalized categorial grammars”. In: *Proceedings of the 11th Annual Meeting of the North Eastern Linguistics Society*. Ed. by V. A. Burke and James Pustejovsky. Amherst, MA: Graduate Linguistics Student Association of the University of Massachusetts, Amherst, 1981, pp. 1–12.

[12] Mark C. Baker. “The Macroparameter in a Microparametric World”. In: *The Limits of Syntactic Variation*. Ed. by Theresa Biberauer. Philadelphia: John Benjamins, 2008, pp. 351–374. DOI: 10.1075/la.132.16bak.

[13] Srinivas Bangalore. “Complexity of lexical descriptions and its relevance for partial parsing”. PhD thesis. University of Pennsylvania, 1997.

[14] Srinivas Bangalore and Aravind K. Joshi. “Introduction”. In: *Supertagging: using complex lexical descriptions in natural language processing*. Ed. by Srinivas Bangalore and Aravind K. Joshi. Cambridge, MA: MIT Press, 2010, pp. 1–31. DOI: 10.7551/mitpress/8370.003.0004.

[15] Giulia M. L. Bencini and Adele E. Goldberg. “The contribution of argument structure constructions to sentence meaning”. In: *Journal of Memory and Language* 43.4 (2000), pp. 640–651. DOI: 10.1006/jmla.2000.2757.

[16] Robert C. Berwick. “Computational Complexity and Lexical-Functional Grammar”. In: *American Journal of Computational Linguistics* 8.3–4 (1982), pp. 97–109. DOI: 10.3115/981923.981926.

[17] Hans C. Boas and Ivan A. Sag, eds. *Sign-Based Construction Grammar*. Stanford, CA: CSLI Publications, 2012.

[18] Hagit Borer. *Parametric syntax: case studies in Semitic and Romance languages*. Studies in Generative Grammar 13. Dordrech: Foris, 1984.

[19] Joan Bresnan, Ash Asudeh, et al. *Lexical-Functional Syntax*. 2nd ed. Blackwell Textbooks in Linguistics 16. Malden, MA: Wiley-Blackwell, 2016.

[20] Joan Bresnan, Ronald M. Kaplan, et al. “Cross-serial Dependencies in Dutch”. In: *Linguistic Inquiry* 13.4 (1982), pp. 613–635. URL: https://www.jstor.org/stable/4178298. Reprinted in Savitch et al. (1987, pp. 286–319).

[21] Tore Burheim. “Aspects of merging Lexical-Functional Grammar with Lexicalized Tree-Adjoining Grammar”. Unpublished ms., University of Bergen. 1996.

[22] Marie-Hélène Candito. “A principle-based hierarchical representation of LTAGs”. In: *Proceedings of the 16th conference on Computational Linguistics (COLING)*. Association for Computational Linguistics. 1996, pp. 194–199. DOI: 10.3115/992628.992664.

[23] Marie-Hélène Candito. “Représentation modulaire et paramétrable de grammaires électroniques lexicalisées : application au français et à l’italien”. PhD thesis. Université Paris 7, 1999.

[24] Lucas Champollion. “Binding Theory in LTAG”. In: *Proceedings of the Ninth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+9)*. Tübingen, Germany: Association for Computational Linguistics, 2008, pp. 1–8. URL: https://aclanthology.org/W08-2301.

[25] Noam Chomsky. “Remarks on Nominalization”. In: *Readings in English Transformational Grammar*. Ed. by Roderick A. Jacobs and Peter S. Rosenbaum. Waltham, MA: Ginn, 1970, pp. 184–221.

[26] Noam Chomsky. *The Minimalist Program*. Cambridge, MA: The MIT Press, 1995. DOI: 10.7551/mitpress/9780262527347.001.0001.

[27] Noam Chomsky. “Three Models for the Description of Language”. In: *IRE Transactions on Information Theory* 2.3 (1956), pp. 113–124. DOI: 10.1109/tit.1956.1056813.

[28] Lionel Clément and Alexandra Kinyon. “Generating LFGs with a MetaGrammar”. In: *Proceedings of the LFG ’03 Conference*. Ed. by Miriam Butt and Tracy Holloway King. Stanford, CA: CSLI Publications, 2003, pp. 105–125.

[29] Lionel Clément and Alexandra Kinyon. “Generating parallel multilingual LFG-TAG grammars from a metagrammar”. In: *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. 2003, pp. 184–191.

[30] Benoît Crabbé et al. “XMG: eXtensible MetaGrammar”. In: *Computational Linguistics* 39.3 (2013), pp. 591–629. DOI: 10.1162/COLI_a_00144.

[31] Richard Crouch et al. *XLE Documentation*. Xerox Palo Alto Research Center. Palo Alto, CA, 2008. URL: https://ling.sprachwiss.uni-konstanz.de/pages/xle/doc/xle_toc.html.

[32] Mary Dalrymple. *Lexical Functional Grammar*. Syntax and Semantics 34. New York: Academic Press, 2001. DOI: 10.1163/9781849500104.

[33] Mary Dalrymple. *The Syntax of Anaphoric Binding*. Stanford, CA: CSLI Publications, 1993.

[34] Mary Dalrymple, Dag T. Haug, and John J. Lowe. “Integrating LFG’s binding theory with PCDRT”. In: *Journal of Language Modelling* 6.1 (2018), pp. 87–129. DOI: 10.15398/jlm.v6i1.204.

[35] Mary Dalrymple, Ronald M. Kaplan, and Tracy Holloway King. “Linguistic Generalizations Over Descriptions”. In: *Proceedings of the LFG ’04 Conference*. Ed. by Miriam Butt and Tracy Holloway King. Stanford, CA: CSLI Publications, 2004, pp. 199–208.

[36] Mary Dalrymple, Ronald M. Kaplan, John T. Maxwell III, et al., eds. *Formal Issues in Lexical-Functional Grammar*. Stanford, CA: CSLI Publications, 1995.

[37] Mary Dalrymple, John J. Lowe, and Louise Mycock. *The Oxford Reference Guide to Lexical Functional Grammar*. Oxford: Oxford University Press, 2019. DOI: 10.1093/oso/9780198733300.001.0001.

[38] Anna Maria Di Sciullo and Edwin Williams. *On the definition of word*. Linguistic Inquiry Monographs 14. Cambridge, MA: MIT Press, 1987.

[39] Jason Eisner and Giorgio Satta. “A faster parsing algorithm for Lexicalized Tree-Adjoining Grammars”. In: *Proceedings of the Fifth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+5)*. Université Paris 7, 2000, pp. 79–84. URL: http://aclweb.org/anthology/W00-2011.

[40] Fernanda Ferreira. “Syntax in language production: an approach using Tree-Adjoining Grammars”. In: *Aspects of language production*. Ed. by Linda Wheeldon. Hove: Psychology Press, 2000, pp. 291–330.

[41] Fernanda Ferreira, Ellen F. Lau, and Karl G. D. Bailey. “Disfluencies, language comprehension, and Tree Adjoining Grammars”. In: *Cognitive Science* 28.5 (2004), pp. 721–749. DOI: 10.1207/s15516709cog2805_5.

[42] Charles J. Fillmore, Paul Kay, and Mary Catherine O’Connor. “Regularity and idiomaticity in grammatical constructions: The case of *let alone*”. In: *Language* 64 (1988), pp. 501–538.

[43] Jamie Y. Findlay. “Mapping Theory and the anatomy of a verbal lexical entry”. In: *Proceedings of the LFG ’20 Conference*. Ed. by Miriam Butt and Ida Toivonen. Stanford, CA: CSLI Publications, 2020, pp. 127–147.

[44] Jamie Y. Findlay. “Mapping Theory Without Argument Structure”. In: *Journal of Language Modelling* 4.2 (2016), pp. 293–338. DOI: 10.15398/jlm.v4i2.171.

[45] Jamie Y. Findlay. “Multiword expressions and lexicalism”. In: *Proceedings of the LFG ’17 Conference*. Ed. by Miriam Butt and Tracy Holloway King. Stanford, CA: CSLI Publications, 2017, pp. 200–229.

[46] Jamie Y. Findlay. “Multiword expressions and lexicalism: the view from LFG”. In: *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*. Valencia, Spain: Association for Computational Linguistics, 2017, pp. 73–79. URL: http://aclweb.org/anthology/W17-1709.

[47] Jamie Y. Findlay. “Multiword expressions and the lexicon”. D.Phil. Thesis. Oxford: University of Oxford, 2019.

[48] Anette Frank and Josef van Genabith. “GlueTag: Linear logic based semantics for LTAG – and what it teaches us about LFG and LTAG”. In: *Proceedings of the LFG ’01 Conference*. Ed. by Miriam Butt and Tracy Holloway King. Stanford, CA: CSLI Publications, 2001, pp. 104–126.

[49] Claire Gardent and Laura Kallmeyer. “Semantic construction in feature-based TAG”. In: *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics (EACL ’03)*. Vol. 1. 2003, pp. 123–130. DOI: 10.3115/1067807.1067825.

[50] Gerald Gazdar et al. *Generalized Phrase Structure Grammar*. Cambridge, MA: Harvard University Press, 1985.

[51] Adele E. Goldberg. *Constructions at Work: The nature of generalization in language*. Oxford: Oxford University Press, 2006.

[52] Adele E. Goldberg. *Constructions: A Construction Grammar Approach to Argument Structure*. Chicago: University of Chicago Press, 1995.

[53] Micah B. Goldwater and Arthur B. Markman. “Constructional sources of implicit agents in sentence comprehension”. In: *Cognitive Linguistics* 20.4 (2009), pp. 675–702. DOI: 10.1515/COGL.2009.029.

[54] Saul Gorn. “Explicit definitions and linguistic dominoes”. In: *Systems and Computer Science*. Ed. by John Francis Hart and S. Takasu. University of Toronto Press, 1967, pp. 77–115.

[55] Sheila A. Greibach. “A new normal-form theorem for context-free phrase structure grammars”. In: *Journal of the Association for Computing Machinery* 12.1 (1965), pp. 42–52. DOI: 10.1145/321250.321254.

[56] Jane Grimshaw. “Extended projection”. In: *Words and structure*. [Published version of 1991 manuscript by the same name]. Stanford, CA: CSLI Publications, 2005, pp. 1–74.

[57] Jane Grimshaw. “Locality and Extended Projection”. In: *Lexical Specification and Insertion*. Ed. by Peter Coopmans, Martin Everaert, and Jane Grimshaw. Amsterdam/Philadelphia: John Benjamins, 2000, pp. 115–133. DOI: 10.1075/cilt.197.07gri.

[58] Zellig Harris. “From Morpheme to Utterance”. In: *Language* 22.3 (1946), pp. 161–183. DOI: 10.2307/410205. Reprinted in Joos (ed) (1957), *Readings In Structural Linguistics*, University of Chicago Press, 142–153.

[59] Thomas Hoffmann and Graeme Trousdale, eds. *The Oxford Handbook of Construction Grammar*. Oxford: Oxford University Press, 2013. DOI: 10.1093/oxfordhb/9780195396683.001.0001.

[60] Matthew A. Johnson and Adele E. Goldberg. “Evidence for automatic accessing of constructional meaning: Jabberwocky sentences prime associated verbs”. In: *Language and Cognitive Processes* 28.10 (2012), pp. 1–14. DOI: 10.1080/01690965.2012.717632.

[61] Aravind K. Joshi. “An introduction to Tree Adjoining Grammars”. In: *Mathematics of language: proceedings of a conference held at the University of Michigan, Ann Arbor, October 1984*. Ed. by Alexis Manaster-Ramer. Amsterdam: John Benjamins, 1987, pp. 87–114. DOI: 10.1075/z.35.07jos.

[62] Aravind K. Joshi. “Tree adjoining grammars: how much context-sensitivity is required to provide reasonable structural descriptions?” In: *Natural language parsing: Psychological, computational, and theoretical perspectives*. Ed. by David R. Dowty, Lauri Karttunen, and Arnold M. Zwicky. Cambridge: Cambridge University Press, 1985, pp. 206–250.

[63] Aravind K. Joshi. “Tree-adjoining grammars”. In: *The Oxford handbook of computational linguistics*. Ed. by Ruslan Mitkov. 1st ed. Oxford: Oxford University Press, 2005, pp. 483–498. DOI: 10.1093/oxfordhb/9780199276349.013.0026.

[64] Aravind K. Joshi, Leon S. Levy, and Masako Takahashi. “Tree adjunct grammars”. In: *Journal of Computer and System Sciences* 10.1 (1975), pp. 136–163. DOI: 10.1016/S0022-0000(75)80019-5.

[65] Aravind K. Joshi and Yves Schabes. “Tree-Adjoining Grammars”. In: *Handbook of formal languages, vol. 3: beyond words*. Ed. by Grzegorz Rozenberg and Arto Salomaa. Berlin: Springer, 1997, pp. 69–123.

[66] Aravind K. Joshi and K. Vijay-Shanker. “Compositional semantics with Lexicalized Tree-Adjoining Grammar (LTAG): how much underspecification is necessary?” In: *Computing meaning*. Ed. by Harry Bunt, Reinhard Muskens, and Elias Thijsse. Vol. 2. Studies in Linguistics and Philosophy 77. Dordrecht: Springer, 2001, pp. 147–163. DOI: 10.1007/978-94-010-0572-2_9.

[67] Aravind K. Joshi, K. Vijay-Shanker, and David Weir. “The convergence of mildly context-sensitive formalisms”. In: *Foundational issues in natural language processing*. Ed. by Peter Sells, Stuart M. Shieber, and Thomas Wasow. Cambridge, MA: MIT Press, 1991.

[68] Aravind K. Joshi and Takashi Yokomori. *Parsing of tree adjoining grammars*. Tech. rep. Department of Computer and Information Science, University of Pennsylvania, 1983.

[69] Laura Kallmeyer. “Local Tree Description Grammars”. In: *Grammars* 4 (2001), pp. 85–137.

[70] Laura Kallmeyer. *Parsing Beyond Context-Free Grammars*. Berlin: Springer, 2010. DOI: 10.1007/978-3-642-14846-0.

[71] Laura Kallmeyer and Aravind K. Joshi. “Factoring predicate argument and scope semantics: underspecified semantics with LTAG”. In: *Research on Language and Computation* 1.1–2 (2003), pp. 3–58. DOI: 10.1023/A:1024564228892.

[72] Laura Kallmeyer and Marco Kuhlmann. “A Formal Model for Plausible Dependencies in Lexicalized Tree Adjoining Grammar”. In: *Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11)*. Paris, France, Sept. 2012, pp. 108–116. URL: https://aclanthology.org/W12-4613.

[73] Laura Kallmeyer, Timm Lichte, et al. “TuLiPA: towards a multi-formalism parsing environment for grammar engineering”. In: *Proceedings of the workshop on grammar engineering across frameworks (GEAF08)*. Ed. by Stephen Clark and Tracy Holloway King. Association for Computational Linguistics, 2008, pp. 1–8. URL: https://www.aclweb.org/anthology/W08-1701.pdf.

[74] Laura Kallmeyer and Rainer Osswald. “Syntax-driven semantic frame composition in Lexicalized Tree Adjoining Grammars”. In: *Journal of Language Modelling* 1.2 (2013), pp. 267–330. DOI: 10.15398/jlm.v1i2.61.

[75] Laura Kallmeyer and Maribel Romero. “LTAG semantics with semantic unification”. In: *TAG+7: Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms*. Vancouver, BC, 2004, pp. 155–162. URL: https://aclanthology.org/W04-3321.

[76] Laura Kallmeyer and Maribel Romero. “Scope and situation binding in LTAG using semantic unification”. In: *Research on Language and Computation* 6.1 (2008), pp. 3–52. DOI: 10.1007/s11168-008-9046-6.

[77] Megumi Kameyama. “Characterising Lexical Functional Grammar (LFG) in terms of Tree Adjoining Grammar (TAG)”. Unpublished ms., Department of Computer and Information Science, University of Pennsylvania. 1986.

[78] Makoto Kanazawa and Sylvain Salvati. “MIX is not a tree-adjoining language”. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2012, pp. 666–674. URL: https://www.aclweb.org/anthology/P12-1070.

[79] Ronald M. Kaplan. “The Formal Architecture of Lexical-Functional Grammar”. In: *Journal of Information Science and Engineering* 5 (1989), pp. 305–322. Revised version published in Dalrymple, Kaplan, Maxwell, et al. (1995, pp. 7–27).

[80] Ronald M. Kaplan. “The Formal Architecture of Lexical-Functional Grammar”. In: *Formal Issues in Lexical-Functional Grammar*. Ed. by Mary Dalrymple et al. Stanford, CA: CSLI Publications, 1995, pp. 7–27. Earlier version published as Kaplan (1989).

[81] Ronald M. Kaplan and Joan Bresnan. “Lexical-Functional Grammar: A Formal System for Grammatical Representation”. In: *The Mental Representation of Grammatical Relations*. Ed. by Joan Bresnan. Cambridge, MA: The MIT Press, 1982, pp. 173–281. Reprinted in Dalrymple, Kaplan, Maxwell, et al. (1995, pp. 29–130).

[82] Ronald M. Kaplan, John T. Maxwell III, and Annie Zaenen. “Functional uncertainty”. In: *CSLI Publications Monthly Newsletter*. Stanford, CA: Stanford University, 1987.

[83] Ronald M. Kaplan and Paula S. Newman. “Lexical resource reconciliation in the Xerox Linguistic Environment”. In: *Proceedings of the ACL Workshop on Computational Environments for Grammar Development and Engineering*. Association for Computational Linguistics, 1997.

[84] Ronald M. Kaplan and Annie Zaenen. “Long-Distance Dependencies, Constituent Structure, and Functional Uncertainty”. In: *Alternative Conceptions of Phrase Structure*. Ed. by Mark Baltin and Anthony Kroch. Chicago: University of Chicago Press, 1989, pp. 17–42. Reprinted in Dalrymple, Kaplan, Maxwell, et al. (1995, pp. 137–165).

[85] Jungo Kasai et al. “TAG parsing with neural networks and vector representations of supertags”. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, 2017, pp. 1712–1722. DOI: 10.18653/v1/D17-1180. URL: http://aclweb.org/anthology/D17-1180.

[86] Michael P. Kaschak and Arthur M. Glenberg. “Constructing meaning: the role of affordances and grammatical constructions in sentence comprehension”. In: *Journal of Memory and Language* 43.3 (2000), pp. 508–529. DOI: 10.1006/jmla.2000.2705.

[87] Paul Kay and Charles J. Fillmore. “Grammatical constructions and linguistic generalizations: The *What’s X doing Y?* construction”. In: *Language* 75 (1999), pp. 1–33.

[88] Alexandra Kinyon. "Hypertags". In: *The 18th International Conference on Computational Linguistics (COLING 2000, vol. 1)*. 2000. URL: https://aclanthology.org/C00-1065.

[89] Alexander Koller. "A feature structure algebra for FTAG". In: *Proceedings of the 13th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+13)*. 2017. URL: https://www.aclweb.org/anthology/W17-6201.

[90] Anthony Kroch. "Unbounded dependencies and subjacency in a Tree Adjoining Grammar". In: *Mathematics of language: proceedings of a conference held at the University of Michigan, Ann Arbor, October 1984*. Ed. by Alexis Manaster-Ramer. Amsterdam: John Benjamins, 1987, pp. 143–172. DOI: 10.1075/z.35.09kro.

[91] Anthony Kroch and Aravind K. Joshi. *The linguistic relevance of Tree Adjoining Grammar*. Tech. rep. MC-CS-85-16. Department of Computer and Information Sciences, University of Pennsylvania, 1985.

[92] Marco Kuhlmann. *Dependency Structures and Lexicalized Grammars: An Algebraic Approach*. Berlin: Springer, 2010. DOI: 10.1007/978-3-642-14568-1.

[93] Iddo Lev. "Packed computation of exact meaning representations". PhD thesis. Stanford, CA: Stanford University, 2007.

[94] Timm Lichte and Laura Kallmeyer. "Tree-Adjoining Grammar: a tree-based constructionist grammar framework for natural language understanding". In: *Proceedings of the AAAI 2017 Spring Symposium on Computational Construction Grammar and Natural Language Understanding*. Technical Report SS-17-02. Association for the Advancement of Artificial Intelligence, 2017, pp. 205–212. URL: https://www.aaai.org/ocs/index.php/SSS/SSS17/paper/viewFile/15330/14536.

[95] John T. Maxwell III and Ronald M. Kaplan. "An Overview of Disjunctive Constraint Satisfaction". In: *Proceedings of the 4th International Workshop on Parsing Technologies (IWPT 1995)*. 1989, pp. 18–27. Also published in Tomita (1991) as 'A Method for Disjunctive Constraint Satisfaction', and reprinted in Dalrymple, Kaplan, Maxwell, et al. (1995, pp. 381–402).

[96] John T. Maxwell III and Ronald M. Kaplan. "The interface between phrasal and functional constraints". In: *Computational Linguistics* 19 (1993), pp. 571–590.

[97] John T. Maxwell III and Ronald M. Kaplan. "Unification-based Parsers that Automatically Take Advantage of Context Freeness". In: *Proceedings of the LFG '96 Conference*. Ed. by Miriam Butt and Tracy Holloway King. Stanford, CA: CSLI Publications, 1996, pp. 1–31.

[98] James D. McCawley. "Concerning the Base Component of a Transformational Grammar". In: *Foundations of Language* 4.3 (1968), pp. 243–269.

[99] Igor A. Mel'čuk. *Dependency syntax: Theory and practice*. Albany: State University of New York Press, 1988.

[100] Ryuichi Nakanishi, Hiroyuki Seki, and Tadao Kasami. "On the generative capacity of lexical-functional grammars". In: *IEICE Transactions on Information and Systems* E75-D.4 (1992), pp. 509–516. URL: https://search.ieice.org/bin/summary.php?id=e75-d_4_509.

[101] Mark-Jan Nederhof. "A short proof that O_2 is an MCFL". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2016, pp. 1117–1126. URL: https://www.aclweb.org/anthology/P16-1106.

[102] Rebecca Nesson and Stuart M. Shieber. "Extraction Phenomena in Synchronous TAG Syntax and Semantics". In: *Proceedings of the Workshop on Syntax and Structure in Statistical Translation*. Ed. by Dekai Wu and David Chiang. Rochester, New York, 26 April 2007. 2007.

[103] Rebecca Nesson and Stuart M. Shieber. "Simpler TAG semantics through synchronization". In: *Proceedings of the 11th Conference on Formal Grammar (FG 2006)*. Ed. by Shuly Wintner. Stanford, CA: CSLI Publications, 2006, pp. 129–142. URL: https://web.stanford.edu/group/cslipublications/cslipublications/FG/2006/nesson.pdf.

[104] Rebecca Nesson and Stuart M. Shieber. "Synchronous Vector-TAG for Natural Language Syntax and Semantics". In: *Proceedings of the Ninth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+ 9)*. Tübingen, Germany, July 2008. URL: http://tagplus9.cs.sfu.ca/papers/NessonShieber.pdf.

[105] Joakim Nivre et al. "Universal Dependencies v1: A Multilingual Treebank Collection". In: *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC'16)*. Portorož, Slovenia: European Language Resources Association (ELRA), 2016, pp. 1659–1666. URL: https://aclanthology.org/L16-1262.

[106] Barbara Partee, Alice ter Meulen, and Robert E. Wall. *Mathematical methods in linguistics*. Dordrecht: Kluwer, 1990.

[107] Andrew Pawley and Frances Hodgetts Syder. "Two puzzles for linguistic theory: nativelike selection and nativelike fluency". In: *Language and communication*. Ed. by Jack C. Richards and Richard W. Schmidt. London: Longman, 1983, pp. 191–226.

[108] P. Stanley Peters and R. W. Ritchie. "On the generative power of transformational grammars". In: *Information Sciences* 6 (1973), pp. 49–83. DOI: 10.1016/0020-0255(73)90027-3.

[109] Carl Pollard. "The Nature of Constraint-Based Grammar". In: *Linguistic Research* 15 (1997), pp. 1–18. URL: http://isli.khu.ac.kr/journal/content/data/15/1.pdf.

[110] Carl Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*. Chicago: University of Chicago Press and CSLI Publications, 1994.

[111] Adam Przepiórkowski. "A full-fledged hierarchical lexicon in LFG: the FrameNet Approach". In: *The Very Model of a Modern Linguist – In Honor of Helge Dyvik*. Ed. by Victoria Rosén and Koenraad De Smedt. 8. Bergen: Bergen Language and Linguistics Studies (BeLLS), 2017, pp. 202–219. DOI: 10.15845/bells.v8i1.1336.

[112] Geoffrey K. Pullum and Gerald Gazdar. "Natural languages and context-free languages". In: *Linguistics and Philosophy* (1982), pp. 471–504. DOI: 10.1007/bf00360802.

[113] Friedmann Pulvermüller. “Brain embodiment of syntax and grammar: discrete combinatorial mechanisms spelt out in neuronal circuits”. In: *Brain and Language* 112.3 (2010), pp. 167–179. DOI: 10.1016/j.bandl.2009.08.002.

[114] Owen Rambow and Aravind K. Joshi. “A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena”. In: *Recent trends in Meaning-Text Theory*. Ed. by Leo Wanner. Studies in Language Companion Series 39. Amsterdam: John Benjamins, 1997, pp. 167–190.

[115] Owen Rambow, K. Vijay-Shanker, and David Weir. “D-Tree Grammars”. In: *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL ’95)*. 1995, pp. 151–158. DOI: 10.3115/981658.981679.

[116] Owen Rambow, K. Vijay-Shanker, and David Weir. “D-tree substitution grammars”. In: *Computational Linguistics* 27.1 (2001), pp. 87–121. DOI: 10.1162/089120101300346813.

[117] James Rogers. *A Descriptive Approach to Language-Theoretic Complexity*. Stanford, CA: CSLI Publications, 1998.

[118] James Rogers and K. Vijay-Shanker. “Obtaining Trees from Their Descriptions: An Application to Tree-adjoining Grammars”. In: *Computational Intelligence* 10.4 (1994), pp. 401–421. DOI: 10.1111/j.1467-8640.1994.tb00005.x.

[119] Neville Ryant and Tatjana Scheffler. “Binding of Anaphors in LTAG”. In: *Proceedings of the Eighth International Workshop on Tree Adjoining Grammar and Related Formalisms*. Sydney, Australia, 2006, pp. 65–72. URL: http://www.aclweb.org/anthology/W/W06/W06-1509.

[120] Sylvain Salvati. “MIX is a 2-MCFL and the word problem in $\mathbb{Z}^2$ is captured by the IO and the OI hierarchies”. In: *Journal of Computer and System Sciences* 18.7 (2015), pp. 1252–1277. DOI: 10.1016/j.jcss.2015.03.004.

[121] Walter J. Savitch et al., eds. *The Formal Complexity of Natural Language*. Studies in Linguistics and Philosophy. Dordrecht: Springer, 1987.

[122] Yves Schabes. “Mathematical and computational aspects of lexicalized grammars”. PhD thesis. University of Pennsylvania, 1990.

[123] Yves Schabes, Anne Abeillé, and Aravind K. Joshi. “Parsing strategies with ‘lexicalized’ grammars: application to Tree Adjoining Grammars”. In: *COLING ’88: Proceedings of the 12th Conference on Computational Linguistics*. Stroudsburg, PA: Association for Computational Linguistics, 1988, pp. 578–583. DOI: 10.3115/991719.991757.

[124] Hiroyuki Seki et al. “Parallel Multiple Context-Free Grammars, Finite-State Translation Systems, and Polynomial-Time Recognizable Subclasses of Lexical-Functional Grammars”. In: *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*. Columbus, OH: Association for Computational Linguistics, June 1993, pp. 130–139. DOI: 10.3115/981574.981592.

[125] Stuart M. Shieber. “Evidence against the context-freeness of natural language”. In: *Linguistics and Philosophy* 8.3 (1985), pp. 333–343. DOI: 10.1007/978-94-009-3401-6_12.

[126] Stuart M. Shieber and Yves Schabes. “Synchronous Tree Adjoining Grammars”. In: *Proceedings of the 13th International Conference on Computational Linguistics (COLING ’90)*. Helsinki, 1990, pp. 253–258.

[127] Jane Simpson. *Warlpiri Morpho-Syntax: A Lexicalist Approach*. Dordrecht: Kluwer Academic Publishers, 1991.

[128] Mark Steedman. “Combinatory Categorial Grammar”. In: *Current approaches to syntax: a comparative handbook*. Ed. by András Kertész, Edith Moravcsik, and Csilla Rákosi. Berlin: De Gruyter Mouton, 2019, pp. 389–420.

[129] Mark Steedman. “Combinatory grammars and parasitic gaps”. In: *Natural Language & Linguistic Theory* 5.3 (1987), pp. 403–439. DOI: 10.1007/BF00134555.

[130] Mark Steedman. *The syntactic process*. Cambridge, MA: MIT Press, 2000.

[131] Dennis R. Storoshenko and Chung-hye Han. “Using synchronous tree adjoining grammar to model the typology of bound variable pronouns”. In: *Journal of Logic and Computation* 25.2 (2013), pp. 371–403. DOI: 10.1093/logcom/exs064.

[132] Dennis R. Storoshenko, Chung-hye Han, and David Potter. “Reflexivity in English: An STAG analysis”. In: *Proceedings of the Ninth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+9)*. Tübingen, Germany, June 2008, pp. 149–156. URL: http://www.aclweb.org/anthology/W08-2320.

[133] Masaru Tomita, ed. *Current Issues in Parsing Technology*. Dordrecht: Kluwer Academic Publishers, 1991. DOI: 10.1007/978-1-4615-3986-5.

[134] K. Vijay-Shanker. “A Study of Tree Adjoining Grammars”. PhD thesis. University of Pennsylvania, 1987.

[135] K. Vijay-Shanker. “Using Descriptions of Trees in a Tree Adjoining Grammar”. In: *Computational Linguistics* 18.4 (Dec. 1992), pp. 481–517. URL: http://dl.acm.org/citation.cfm?id=176313.176317.

[136] K. Vijay-Shanker and Aravind K. Joshi. “Feature Structure Based Tree Adjoining Grammars”. In: *Proceedings of the 12th Conference on Computational Linguistics (COLING ’88)*. Budapest, Hungry: Association for Computational Linguistics, 1988, pp. 714–719. DOI: 10.3115/991719.991783.

[137] K. Vijay-Shanker and Aravind K. Joshi. “Some computational properties of Tree Adjoining Grammars”. In: *Proceedings of the 23rd annual meeting on Association for Computational Linguistics (ACL ’85)*. Association for Computational Linguistics, 1985, pp. 82–93. DOI: 10.3115/981210.981221.

[138] Jürgen Wedekind and Ronald M. Kaplan. “Tractable Lexical-Functional Grammar”. In: *Computational Linguistics* 46.2 (2020), pp. 515–569. DOI: 10.1162/coli_a_00384.

[139] Rulon S. Wells. “Immediate constituents”. In: *Language* 23.2 (1947), pp. 81–117. URL: http://www.jstor.org/stable/410382.

[140] Edwin Williams. “Dumping lexicalism”. In: *The Oxford handbook of linguistic interfaces*. Ed. by Gillian Ramchand and Charles Reiss. Oxford: Oxford University Press, 2007, pp. 353–382. URL: https://doi.org/10.1093/oxfordhb/9780199247455.013.0012.

[141] Alison Wray. *Formulaic language and the lexicon*. Cambridge: Cambridge University Press, 2002.

[142] XTAG Research Group. *A Lexicalized Tree Adjoining Grammar for English*. Tech. rep. IRCS-01-03. Philadelphia, PA: Institute for Research in Cognitive Science, University of Pennsylvania, 2001. URL: https://www.cis.upenn.edu/~xtag/gramrelease.html.