20

Formalization of RRG syntax

Laura Kallmeyer & Rainer Osswald

20.1 Introduction

Role and Reference Grammar (RRG) has been developed as a theory of grammar which covers typologically distinct languages and which is able to capture the interaction between syntax, semantics, and pragmatics. The design of RRG was not driven by specific formal considerations. In particular, there is no formal core that plays a crucial role in RRG, as, for example, the theory of feature structures does in Head-Driven Phrase Structure Grammar (HPSG; Pollard and Sag 1994). The goal of this chapter is to present RRG as a *formalized* grammatical theory that puts emphasis on mathematical and logical rigor. In particular, it will be shown how the possible universal and language-specific syntactic templates of RRG can be formally specified, and by which formal operations they can be combined to larger syntactic structures.

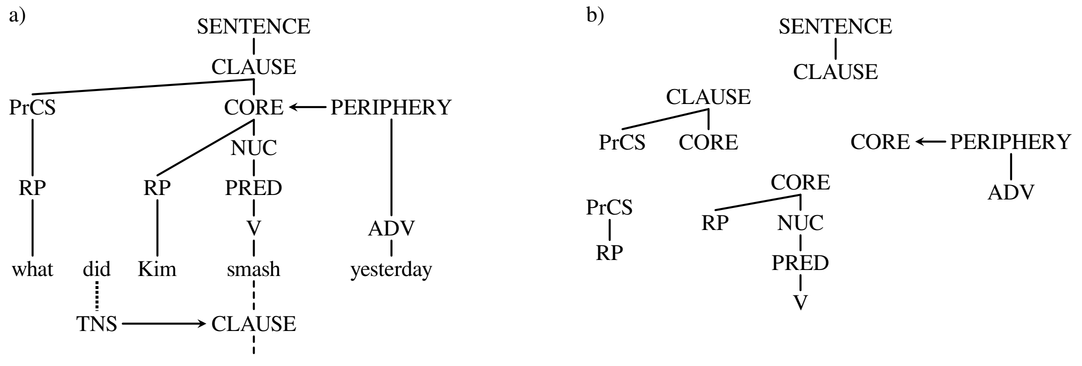
The working typologist who uses RRG as her or his framework for linguistic analysis may not regard a thorough formalization as particularly important. In fact, one of the appeals RRG has for field linguistics is that it does not come with an overly heavy theoretical load but keeps a good balance between a rich and elaborate set of notions and explanatory mechanisms, and a semi-formal, intuitive presentation. A formalization can however help to identify and eliminate possible gaps and inconsistencies of the theory and, thereby, to improve the theory. Moreover, a formalization can serve as a basis for computational implementations of RRG. While a thorough formalization may not be absolutely necessary for a computational treatment from an engineering perspective (cf. Chapter 21 on Computational implementation and applications of RRG), it can contribute to implementations that give full consideration to the overall architecture of RRG as a theory of grammar.

20.2 The task of formalizing RRG syntax

In RRG, syntactic representations are composed of syntactic templates stored in the syntactic inventory. Figure 20.1a shows a simple example of a syntactic representation; possible candidates of syntactic templates are shown in Figure 20.1b. A formalization first needs to decide on what kind of formal structures to use. In line with the tradition of RRG, tree structures will be employed for this purpose, where nodes can carry additional features besides the category labels. The second task is to define the modes by which the syntactic representations are composed from the members of the syntactic inventory (Section 20.3). Next the templates available in the inventory must be specified, and this should be done in a way that allows us to capture generalizations among them within and across languages (Section 20.4). A further point that needs special treatment is the formalization of the operator projection (Section 20.5).

A key component of RRG's approach to syntactic analysis is the layered structure of the clause: sentences are assumed to have an internal structural layering consisting of clause, core,

Figure 20.1. *Examples of syntactic representation and syntactic templates in RRG*



and nucleus. The different layers serve as attachment sites for different types of operators: tense operators attach to the clause, modality to the core, aspect to the nucleus, etc. The core level is also the default attachment site for arguments. In the following, we will refer to the subtree of a syntactic representation consisting of the root and its non-peripheral clause, core, nucleus, and predicating descendants as the *clausal skeleton* of the representation. The syntactic structures in RRG are basically labeled trees, and there are good reasons to use tree structures in a formalization as well. Trees provide the most natural way to analyze syntactic structures since they build on the basic relations *immediate dominance* and *linear precedence*.

20.2.1 Approaches to formalizing RRG syntax

The formal specification of syntactic structures in RRG is briefly addressed by Van Valin and LaPolla (1997, Sect. 2.5), where two approaches are discussed: (i) the specification by immediate dominance and linear precedence rules and (ii) the specification by syntactic tree templates and their combination. As to approach (i), Van Valin and LaPolla (1997, pp. 69f) propose the following universal immediate dominance rules for the constituent structure of simple sentences:[1]

(1)	SENTENCE	→	{(DP)}, CLAUSE
	DP	→	XP | ADV
	CLAUSE	→	{(ECS)}, CORE, (PERIPHERY), {NP*}
	ECS	→	XP | ADV
	PERIPHERY	→	XP | ADV
	CORE	→	ARG*, NUC
	NUC	→	PRED
	PRED	→	V | XP
	ARG	→	PRO | XP
	XP	→	NP | PP

The commas on the right-hand side of these rules do not indicate any ordering of the subconstituents. The ordering is specified by additional linear precedence rules, which are partly universal and partly language-specific. For example, English, a verb-medial language, obeys the following linear precedence rules (Van Valin and LaPolla, 1997, p. 71):

(2) NP > NUC
NUC > NP* > PP*

As Van Valin and LaPolla point out, the range of possible syntactic constellations specified by the above rules needs to be further constrained by the linking of syntax to semantics, which includes constraints on the syntactic realization of arguments depending on semantic structures in the lexicon. In addition to the constituent structure rules in (1) and (2), a separate set of immediate dominance and linear precedence rules is needed for specifying the structure of the operator projection.

The idea of specifying the constituent structure and the operator structure separately by different context free grammars was originally proposed by Johnson (1987), based on the observation that the ordering among the operators is systematically correlated with their scope given by their attachment site at the clausal skeleton, whereas the surface order of the operators relative to arguments and adjuncts is much less transparent and often requires crossing branches. The two grammars taken together then constitute a *projection grammar*, giving rise to a *constituent projection* and an *operator projection*. Johnson formally defines a *projection grammar* as a quadruple $P = \langle T, I, (A_i)_{i \in I}, (G_i)_{i \in I} \rangle$, where T is a set of terminal symbols, I is a finite set of "projection" indices and, for every $i \in I$, A_i is a subset of T and G_i is a formal grammar with terminal symbols in A_i. A string s over T then belongs to the language generated by P if and only if its ith projection, that is, the concatenation of the elements of s belonging to A_i, is in the language generated by G_i.

While it seems reasonable to distinguish between constituent structure and operator structure, the proposal of Johnson has the problem of being purely surface oriented. As a consequence, it does not enforce matching clausal skeletons in the two projections. However, corresponding clausal skeletons in both projections are taken for granted in the syntactic representations of RRG. A further problem arises from the assumption that the operator projection can be represented as a tree, i.e., that each operator contributes only to one layer (cf. Section 20.5 for counterexamples).

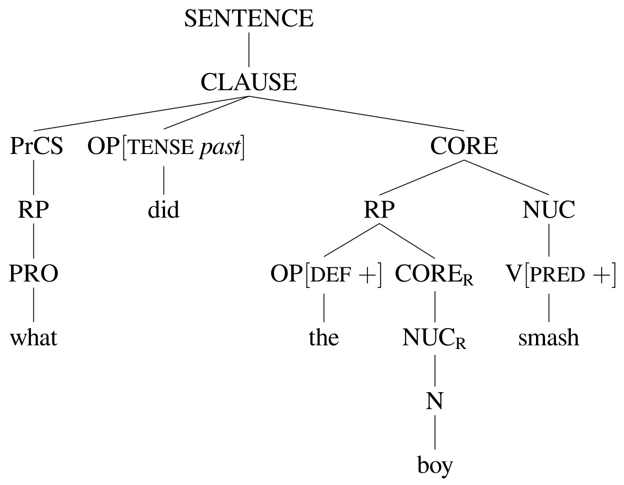
Approach (ii), the second approach discussed by Van Valin and LaPolla, postulates an inventory of elementary syntactic trees that can be combined into more complex syntactic structures. Van Valin and LaPolla (1997, p.654, note 34) point out that *Tree Adjoining Grammars* (TAG, Joshi and Schabes, 1997) may provide a way to formalize such tree templates and their composition. Building on this idea, Kallmeyer et al. (2013) (see also Kallmeyer and Osswald 2017; Osswald and Kallmeyer 2018) propose a formalization of RRG as a grammar based on so-called elementary trees and TAG-inspired tree composition operations. We will detail this approach in Section 20.3.

A slightly different proposal has been suggested by Nolan (2004), who argues for a formalization of RRG that systematically exploits *feature-based* representations, similar in style to HPSG (Pollard and Sag, 1994) and, more recently, Sign-Based Construction Grammar (Sag, 2012). Representing constituent structures in this way calls for features, or attributes, by which the subconstituents can be addressed. This can either be done by reconstructing tree structures as feature structure based on formal features such as FIRST and REST, or by employing functional notions like SUBJECT, DIRECT-OBJECT, etc. However, configurational syntactic notions are usually considered not as basic but as derived concepts in RRG. While the representation of the constituent projection proposed in Nolan (2004, Sect. 5.5), which builds on the immediate dominance rules given in (1), is not fully explicit about the attributes involved, it seems that either pure list-oriented attributes are used (FIRST and REST, or even 1ST, 2ND, 3RD, etc.) or attributes specific to the categories of the subconstituents.

20.2.2 The formal treatment of operators and peripheral elements

The formalization presented in the following assumes a single syntactic tree structure in which operator components are distinguished by a special category OP and a feature structure that characterizes their contribution. For example, the operator *did* in Figure 20.2 contributes [TENSE *past*] to the CLAUSE and the definiteness operator *the* contributes [DEF +] to the RP layer of *boy*.[2] This representation, together with the approach to operator adjunction presented in Section 20.5, turns out to be sufficient for capturing the scope-related ordering among the operators. The operator projection can then be defined as the subtree consisting of the clausal skeleton plus the

Figure 20.2. *Operator marking by features*



components with category OP.[3]

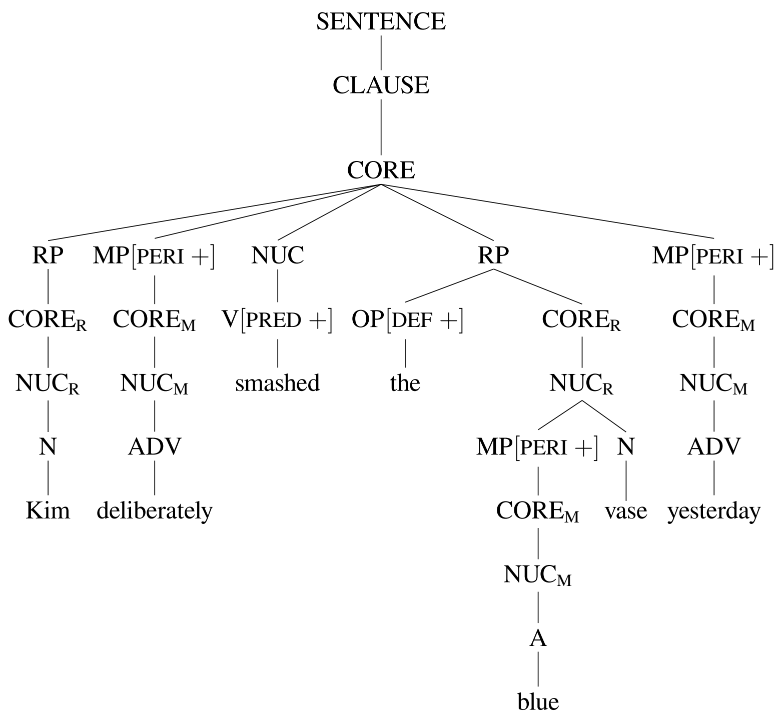
In RRG's graphical presentations of syntactic structures, peripheral structures and linkage markers are usually attached to the clausal skeleton by arrows (cf. Figure 20.1a). The formalization proposed here does not make use of separate PERIPHERY nodes but marks peripheral structures by a feature [PERI +] as indicated in Figure 20.3. As with operators, peripheral elements are subject to the iconicity principle that their relative ordering respects the layering of their attachment sites. This aspect will be taken care of in the formalization of the operator projection and the periphery structure described in Sections 20.5 and 20.7. The proposed modes of combining peripheral structures with the clausal skeleton are similar to those used for operators (Section 20.3.2). Linkage markers can be treated similarly.

To sum up, the syntactic structures of RRG can be formalized as labeled trees, where node labels can carry additional features. Since the labels can be regarded as feature values, too, e.g., category labels as values of the feature CAT, we may assume without restriction of generality that node labels are sets of attribute-value specifications. From this perspective, $\mathrm{MP}_{[\mathrm{PERI}\ +]}$ is short for [CAT *mp*, PERI +]. Introducing features allowed us to get rid of the PERIPHERY nodes, whose only purpose is to mark their daughters as peripheral. By the same line of reasoning we can eliminate the PRED nodes, whose purpose is to mark their daughters as predicating (cf. Van Valin 2005, p. 13), by adding [PRED +] to the respective daughter nodes, as illustrated by the V nodes in Figures 20.2 and 20.3.

20.3 A tree rewriting formalism for syntactic composition in RRG

The standard presentation of RRG gives only an informal description of how syntactic templates are combined to more complex syntactic structures. As mentioned in Section 20.2.1, Van Valin and LaPolla (1997, p. 654, note 34) suggest that a formal account of the modes of composition may show some similarity to Tree Adjoining Grammars (TAG, Joshi and Schabes, 1997). They rightly point out that TAG is a *grammar formalism* and not a *linguistic theory* in its own right. TAG *per se* does not make any commitments about what kind of categories and what kind of syntactic configurations are appropriate for linguistic analysis. There is one caveat, however: the standard adjunction operation of TAG aims at modifying binary branching structures. Since flat syntactic structures are prevalent in RRG, the formalization proposed in the following employs slightly different modes of composition.

Figure 20.3. *Periphery marking by features*



20.3.1 Elementary trees

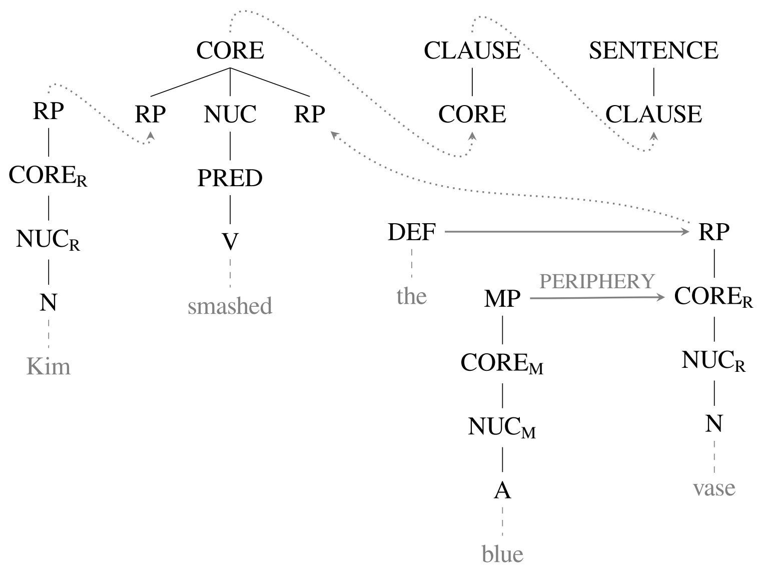
The architecture of RRG assumes an inventory of syntactic templates as elementary building blocks for syntactic representations (cf. Figure 20.1b). These templates often have a more complex structure than just consisting of a root node together with a series of immediate daughters. They can thus capture a wider range of dependencies than standard phrase structure rules like those listed in (1). In TAG, this property is called the *extended domain of locality (EDL)* represented in elementary trees (Joshi and Schabes, 1997, p. 95f). In particular, templates can contain a predicative node and slots for all arguments of that predicate. An example is given in Figure 20.4, which shows the templates that could be used for generating the syntactic structure for (3).

(3) Kim smashed the blue vase.

In this example, the transitive verb template associated with the verbal predicate *smash* contains not only the NUC node but also the CORE and the two RP argument slots. Note that the lexical elements (marked gray in the figure) are not part of the templates but are added in a separate step. The availability of larger syntactic units allows one to *lexicalize* parts of the grammar, up to the point of a fully *Lexicalized* Tree Adjoining Grammars (LTAG), in which every elementary tree is required to have a lexical anchor.

The similarities between RRG's tree templates and (L)TAG's elementary trees suggest a formalization of RRG syntax along the following lines: A language is syntactically described by a tree rewriting grammar comprising a set of tree templates, lexical elements filling the anchor nodes of these templates, and certain operations for combining them to syntactic representations of phrases and sentences. More specifically, a grammar consists of a finite set of *elementary trees* that can be composed inductively via three basic tree composition operations, namely *sister adjunction*, *substitution* and *wrapping substitution*. Elementary trees are defined as labeled ordered trees whose internal nodes are labeled with categories such as CLAUSE, CORE, RP, N, V etc., and whose leaf nodes are labeled either with lexical items or with syntactic categories. The assumption that

Figure 20.4. *RRG templates*



elementary trees are *ordered* trees means: (i) every two nodes that do not stand in a dominance relation are ordered by linear precedence; (ii) if a node n_1 precedes a node n_2 then every node dominated by n_1 precedes every node dominated by n_2. Note that these requirements exclude structures with crossing branches.

Figure 20.5 shows possible elementary trees (including lexical elements) for the syntactic analysis of sentence (3) and illustrates how they combine. The choice of the elementary trees in the example reflects the following general principles: Each lexical predicate comes with its entire layered structure, including argument slots for all its arguments (see the trees for *Kim*, *smashed*, *blue* and *vase* in Figure 20.5). Periphery elements, operators and linkage markers contribute an adjunct tree. Such a tree cannot fill an argument slot but has to be adjoined, i.e., attached to the clausal skeleton as an additional daughter of some appropriate node. An asterisk at the root node indicates that a tree is an adjunct tree (see the trees for the operator *the* and the modifier *blue* in Figure 20.5). In addition, there can also be non-lexicalized elementary trees (see the SENTENCE–CLAUSE tree in Figure 20.5).

The formalization of RRG syntax presented here allows for a further level of (de)composition: Elementary trees are specified in a so-called *metagrammar*, a system of tree constraints, which captures generalizations across elementary trees (more on this in Section 20.4.2). Moreover, lexical items are stored in a separate lexicon and enter their elementary trees by a process of lexical anchoring under constraints. In other words, we distinguish unanchored elementary trees from the anchored trees that enter tree composition. The former are called *elementary tree templates*. Figure 20.6 shows the decomposition of three of the elementary trees from Figure 20.5 into their tree template and the lexical anchor. The place where the anchor has to be inserted is marked with a diamond symbol.

Note that introducing elementary trees as ordered trees does not prevent us from defining grammars for languages with free word order. If, for instance, a language allows the arguments of a verb to appear in an arbitrary order, then the verb would have elementary trees for each such ordering at its disposal. What is important here is that elementary trees are not atomic building blocks but are generated in a modular fashion from classes of tree constraints in the metagrammar.

Figure 20.5. *Elementary RRG trees and their composition*

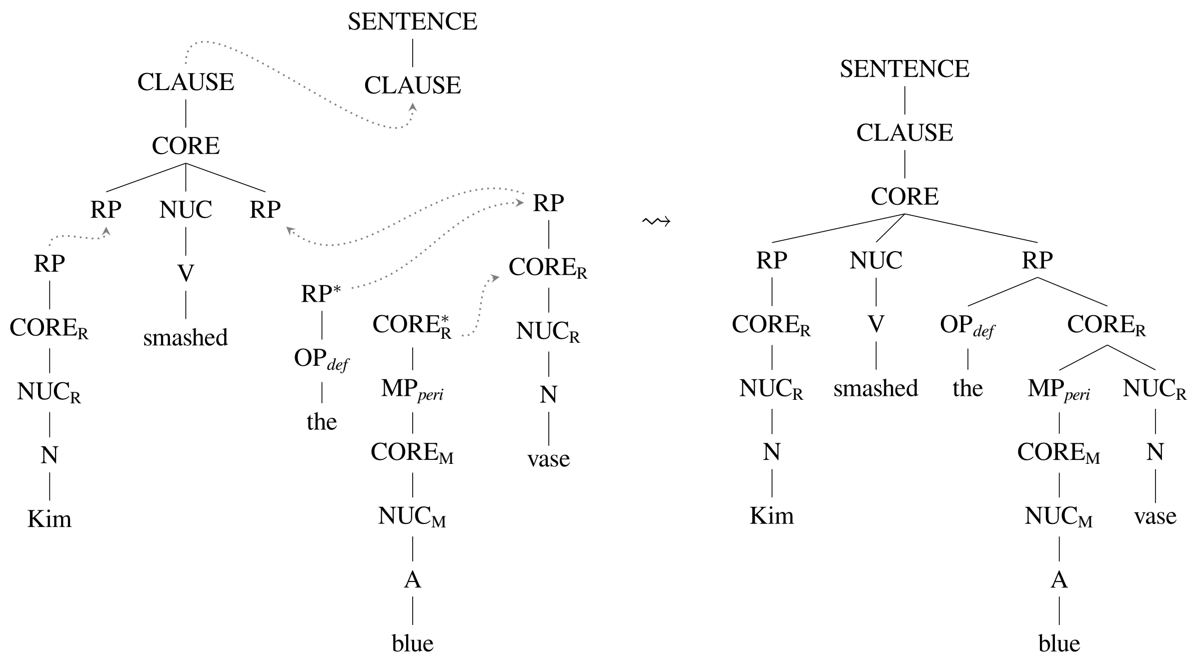
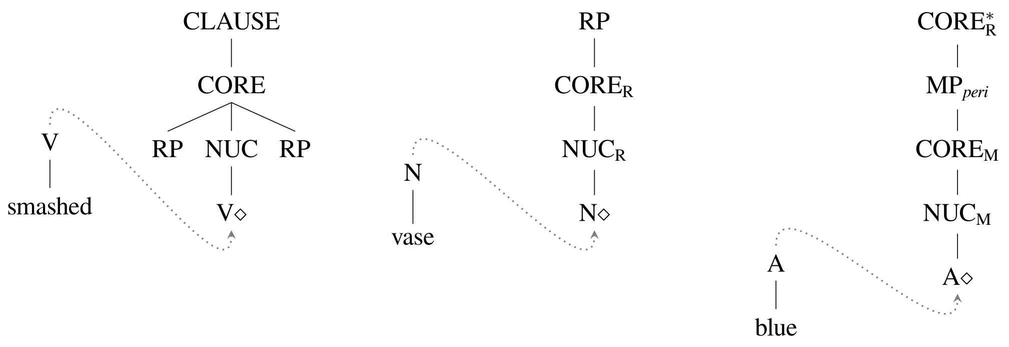


Figure 20.6. *Lexical anchoring of elementary tree templates*

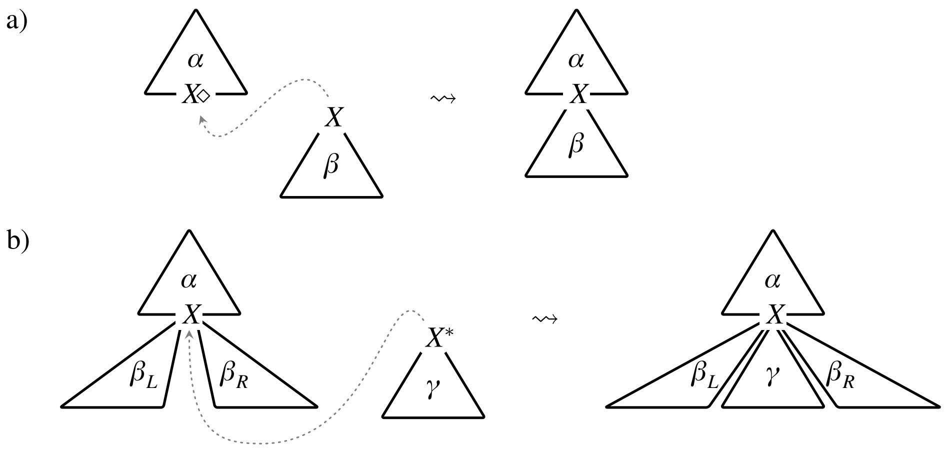


Since fewer ordering constraints in the metagrammar correspond to a larger set of elementary tree templates, it follows, loosely speaking and without taking into account morphological markings, that free word order is descriptively more simple in the metagrammar than strict word order.

20.3.2 Simple substitution and sister adjunction

The most basic mode of composition for syntactic templates is *substitution*. The trees for *Kim* and for *vase* in Figure 20.5, for instance, are added by substitution to the tree of *smashed*, filling the two RP argument slots. In the following, a *tree with label X* is meant to be a tree whose root carries the category label X. A tree β with label X can be substituted for a leaf node labeled with X (the *substitution node*) of a tree α by "identifying" the root node of β with the substitution node (cf. Figure 20.7a). More generally, if the nodes are labeled by feature structures, the two feature structures must be compatible, and the node of the resulting tree is labeled by the *unification* of the two feature structures. Each non-terminal leaf node in an elementary tree is a *substitution node* and must obligatorily be filled by substitution or by the substitution part of wrapping substitution

Figure 20.7. *Schematic sketch of simple substitution a) and sister adjunction b)*



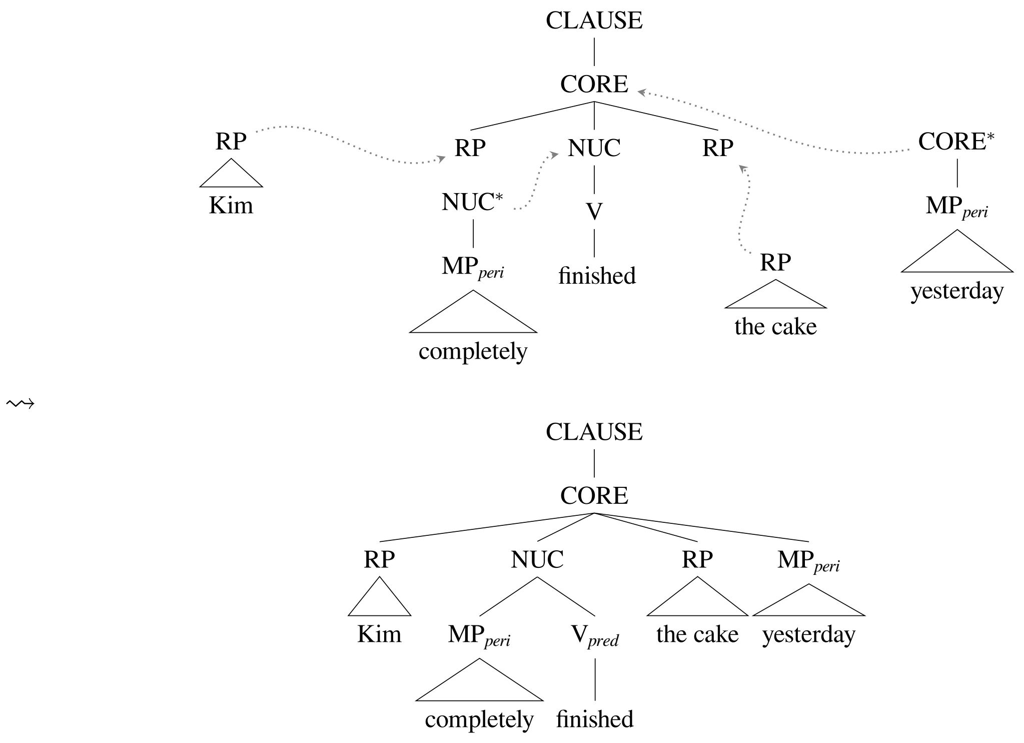
(see below). Substitution is the main mode of composition for expanding argument nodes by the syntactic representations of specific argument realizations.

In RRG, periphery elements and operators are only indirectly linked to the layered structure, by a PERIPHERY edge or by an edge to their layer in the operator projection. We include them in the layered constituent structure while making sure the information about being operator or periphery elements is preserved. More concretely, the directed edge with label PERIPHERY is replaced with an unlabeled immediate dominance edge between the target layer and the periphery element and a feature [PERI +] on the root of the peripheral structure, often abbreviated as a subscript *peri* on the node category. In the case of operators, they also attach as a daughter to the target node in the constituent tree. Their category (OP) indicates that they are operators and their contribution is indicated within further features attached to the OP node. Their elementary trees are rooted by a node that merges with the target tree. An advantage of this is that one can formulate constraints by means of feature specifications. For instance, one can require that there is only one definiteness operator in an RP. We will see how to use features for this purpose in more detail in Section 20.5.

Peripheral structures cannot be added by substitution since they do not attach to leaves but to internal nodes, in general, and the same holds for the operators; cf. Figure 20.5. The mode of composition proposed for these cases is *(sister) adjunction* (see also Kallmeyer et al. 2013).[4] As with substitution, we assume that the templates available for adjunction have a root label which coincides with the label of the target node (cf. Figure 20.5). For convenience, the root of an adjunction tree is marked by an asterisk in the graphical presentations. We call elementary trees with this marking *adjunct trees*. A further constraint on adjunct trees is that their root node has only a single daughter. The idea behind this is, as explained above, that the tree below the root is the actual periphery element while the root node captures more or less the PERIPHERY edge from RRG textbooks. The root label of an adjunction tree specifies the attachment site at the phrasal skeleton. In Figure 20.8 for example, the adverbial *completely* adjoins at the nucleus while *yesterday* adjoins at the core.

Sister adjunction is defined in such a way that the target node must not be a leaf node. Because of their single daughter property, roots of adjunct trees are excluded as target nodes for sister adjunction, but it is of course possible to adjoin more than one adjunct tree to the same node of the target tree. The adjunction operation consists in merging the root of the adjunct tree with the target node (which amounts to unifying their feature structures) and adding the daughter of the adjunct tree root as a new daughter to the target node. This can take place in any position among

Figure 20.8. *Sister adjunction of periphery elements at different layers*



the already existing daughters of that target node.[5] See Figure 20.7b for a general schematic illustration of sister adjunction.

A complication arises when an operator or periphery element targeting a specific layer occurs between elements that are part of a different layer. Examples are given in (4):

(4) a. He ate the apple completely.
    b. John did not eat the apple.

In (4a), the nucleus, which consists of the verb *ate*, is followed by an argument, which is part of the core, and after that comes an adverb that is in the nuclear periphery. In (4b), the tense operator *did* is placed between an argument (part of the core) and the verb (also part of the core). But it should attach at the clausal level. For the moment, we will ignore this complication and come back to it in Sections 20.5 and 20.7.

20.3.3 Wrapping substitution

Control constructions and extraction from complements pose a problem for the modes of composition presented so far. Consider the examples of wh-extraction in (5).

(5) a. What does John think Kim smashed?
    b. What does John think Mary claimed Kim smashed?

Clearly, it would not be appropriate to assume a separate complex template for each of these constructions. The syntactic representations are to be composed of basic argument structure templates in a systematic way. There are several options for achieving this goal, depending on the presumed inventory of elementary templates. First we need to decide on the proper syntactic representations of the examples in (5). While sentences of this type are discussed in the context

Figure 20.9. *Two possible syntactic representations of wh-extraction from complements*

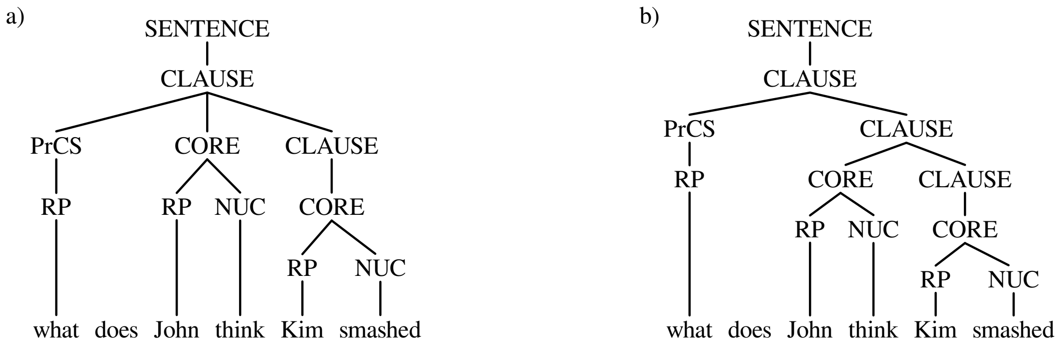
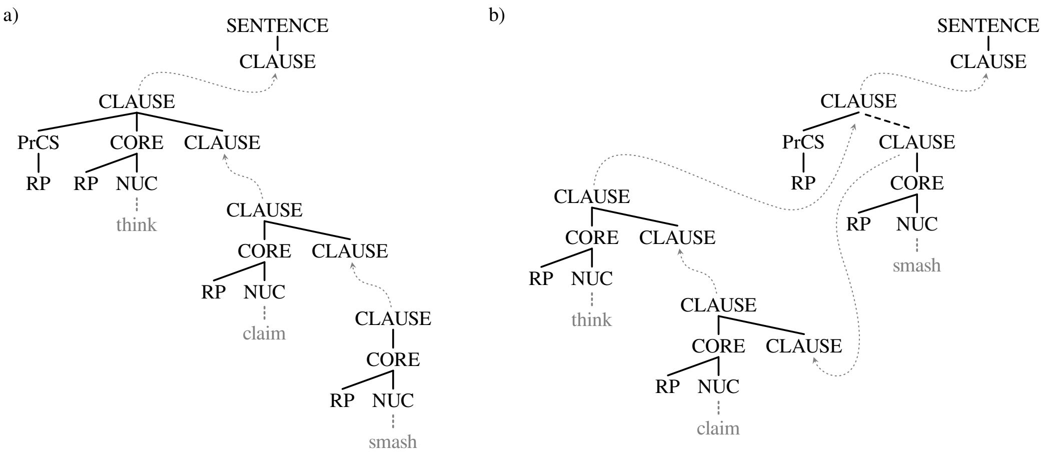


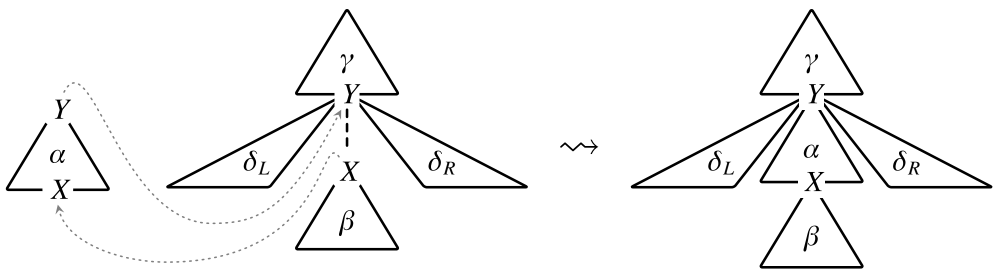
Figure 20.10. *Wh-extraction via simple substitution a) and wrapping substitution b)*



of island constraints in Van Valin and LaPolla (1997, p. 615) and Van Valin (2005, p. 273), no structural analysis is provided there. Due to the nature of the embedding constructions, the basic binary precore slot pattern [$_{\text{CLAUSE}}$ [$_{\text{PrCS}}$ ...][$_{\text{CORE}}$ ...]] shown in Figure 20.1 does not apply to the present case. Figure 20.9 shows two possible alternatives. The analysis in Figure 20.9a assumes a precore variant of the clausal subordination pattern [$_{\text{CLAUSE}}$ [$_{\text{CORE}}$ ...][$_{\text{CLAUSE}}$ ...]]. The structure in Figure 20.9b, by contrast, assumes an additional clause node, and the precore slot pattern is [$_{\text{CLAUSE}}$ [$_{\text{PrCS}}$ ...][$_{\text{CLAUSE}}$ ...]]. In the following, we restrict the discussion to the first option since we regard it as difficult to come up with an independent motivation for the additional clausal node in Figure 20.9b.

Figure 20.10 sketches two ways of composing the syntactic representation of example (5b). Figure 20.10a employs substitution only, but at the price of assuming a special elementary template associated with *think* that has a precore slot in addition to its normal argument slots. Assuming such a template would raise the further problem of providing information about which of the arguments within the embedded clauses is referred to by the referent phrase in the precore slot. The templates in Figure 20.10b are more straightforward in this respect since they represent proper argument structure templates in that the precore slot is locally connected to the core from which the wh-word is extracted. The dashed edge between the CLAUSE node dominating the precore slot and the lower CLAUSE node stands for a dominance relation, i.e., a (possibly empty) sequence of immediate dominance edges in the final derived tree. We call such an edge a *d-edge*. Under this analysis, the long-distance dependency comes about by the insertion of material at the d-edge, that is, between the precore slot and the corresponding core.[6] More flexible templates and a more complicated tree composition mechanism are needed in this case. The clause node of the *smash*

Figure 20.11. *Wrapping substitution*



structure is split in two and its upper part is unified with the upper clause node of the *think* structure (similar to the adjunction mechanism introduced in the previous section), thereby keeping the precore slot on the left side of the structure; the lower part of the split node is substituted at the lower clause node of the *claim* structure. The mode of composition just described is referred to as *wrapping substitution* (Kallmeyer et al., 2013; Osswald and Kallmeyer, 2018).

Wrapping substitution has the general form depicted in Figure 20.11.[7] By definition, this operation involves a tree with a d-edge. Such an edge stands for a dominance relation in the final derived tree, i.e., it specifies a place in an elementary tree where additional nodes and edges from other elementary trees can be inserted.[8] Wrapping consists basically of splitting the tree at the d-edge and wrapping it around a target tree. More specifically, with reference to Figure 20.11, the subtree β rooted at the lower node (labeled *X*) of the d-edge fills a substitution slot in the target tree α while the upper node of the d-edge (labeled *Y*) merges with the root of α. Concerning this upper node, all descendants to the left (resp. right) of the d-edge are located to the left (resp. right) of the target tree α in the resulting combined tree, and all nodes of γ dominating the upper node (labeled *Y*) of the d-edge also dominate the merged node after the wrapping. The example in Figure 20.10b is a case of wrapping with empty γ and δ_R, i.e., a wrapping where the upper part only adds nodes to the left of the d-edge but neither to its right nor above.[9]

It is instructive to compare the different tree composition options of Figure 20.10 with respect to their applicability for linking, especially with respect to template selection. As explained before, the syntactic inventory does not provide a single template for the complex syntactic structure of (5b). Rather, the structure has to be composed from argument structure templates which in turn are selected by the chosen lexical entries. The composition in Figure 20.10a has the disadvantages that the template selected by *think* is not an argument structure template but has to be stipulated for embedded wh-questions, and that an advanced mechanism for coreference is needed. The composition scheme in Figure 20.10b, by comparison, has the advantage that the precore slot can be immediately linked when the template is selected. Here, the underlying assumption is that the syntactic inventory provides argument structure templates for wh-fronting. Note that this assumption differs slightly from RRG's standard understanding of syntactic templates, which would assume a reduced core template plus a precore slot template in this case (Van Valin, 2005, p. 15). Elementary trees in LTAG, by comparison, are commonly assumed to have substitution nodes for all arguments of the lexical head, irrespective whether they are realized within the CORE or outside of it (cf. Section 20.3.4). A further decomposition of elementary trees can be expressed by means of tree descriptions in the metagrammar (cf. Section 20.4.2).

20.3.4 Extended domain of locality

As mentioned in Section 20.3.1, an important characteristics of the LTAG formalism is the *extended domain of locality* (EDL) of elementary trees (Joshi and Schabes, 1997, pp. 95f), which means that elementary trees represent full argument projections and that they can have a complex constituent structure. As explained above, the present formalization of RRG builds on a similar assumption. This is particularly crucial in the case of long distance dependencies across clausal complements, as in Figure 20.10b. Here, the wh-element in the precore slot originates from the same elementary tree as the verb *smash* that it depends on. In the final derived structure, the two are separated from each other by the intervening structures of *claim* and *think*.

A possible alternative solution, avoiding this extended domain of locality and using only simple substitution, is to have the information about the wh-marked argument percolate to the top of the tree. A constraint-based formalization of this percolation process could roughly work as follows: The core node of the reduced template for *smash* carries a (set-valued) feature which contains the referential index of the participant not locally realized as well as its wh-marking. A general constraint then ensures that clause and sentence nodes collect the non-realized indices of their subordinate clauses and cores minus the indices that are realized in precore slot daughters and the like. This way of bookkeeping for modeling long distance dependencies is in fact closely related to the use of 'slash' or 'gap' features in the approaches of Sag and Wasow (1999) and Ginzburg and Sag (2001), among others.

To sum up, the approach just discussed can get along with simple substitution at the price of a considerable amount of bookkeeping. By comparison, the approach exemplified in Figure 20.10b allows a fully local argument linking but requires a more complex method of tree composition. The former strategy is employed in GPSG/HPSG-related frameworks while the latter strategy is characteristic of approaches in the line of LTAG (cf. Kroch 1987).

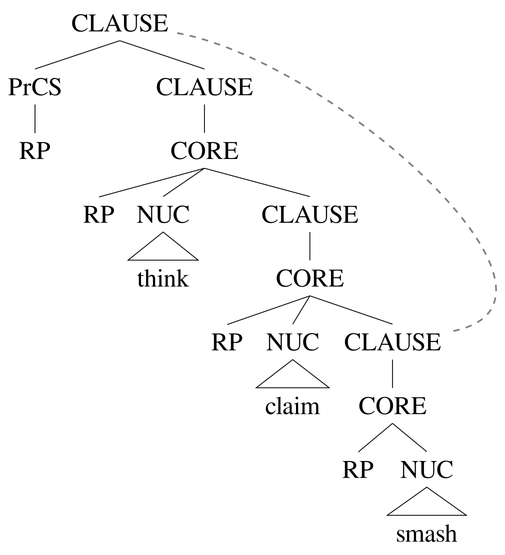
20.3.5 Formal and computational properties of Tree Wrapping Grammars

The type of tree rewriting grammar formalism introduced above, with composition operations sister adjunction, substitution and wrapping substitution, is called *Tree Wrapping Grammar* (TWG, Kallmeyer et al., 2013; Kallmeyer, 2016). Note that Kallmeyer et al. (2013) use a slightly different definition of wrapping substitution, which, however, gives rise to more binary trees. We use the TWG version presented in Kallmeyer (2016) and Osswald and Kallmeyer (2018).

Independent from the concrete shape of elementary trees chosen for RRG, one can investigate the formal properties of TWGs. We do not want to go into details here but restrict ourselves to pointing out the following (see Kallmeyer, 2016): TWGs are more powerful than context-free grammars (CFGs), which is due to the wrapping substitution operation. The expressive power depends in particular on how many d-edges are allowed to stretch across a single node in the final derived tree. In our example in Figure 20.10b, for instance, we have one single d-edge stretching across the roots and cores of the *claim* and the *think* elementary trees (see the gray dashed edge in Figure 20.12, which is the d-edge from the elementary tree of *smash*). A TWG where the maximal number of d-edges stretching across a node is limited to some constant k is called a k-TWG, and Kallmeyer (2016) links k-TWG to simple Context-Free Tree Grammars (CFTG, Kanazawa 2016) of rank k by showing that for every k-TWG, a simple Context-Free Tree Grammar (CFTG) of rank k can be constructed that generates the same language. Simple Context-Free Tree Grammars (CFTG) of rank k are, in turn, equivalent to well-nested Linear Context-Free Rewriting Systems (LCFRS) of fan-out $k + 1$. Consequently, k-TWGs are in particular mildly context-sensitive.

The notion of mildly context-sensitive languages and formalisms was introduced in Joshi (1985) in an attempt to characterize the amount of context-sensitivity required for natural languages where context-sensitivity is used in a formal language sense, in contrast to context-free languages and grammars. Roughly, a mildly context-sensitive grammar formalism is able to generate more than all context-free languages, can account for cross-serial dependencies, is polynomially parsable,[10] and has the constant-growth property, which means that, if we order the sentences of a language according to their length, the length grows in a linear way. It is interesting that

Figure 20.12. *D-edge in final derived tree for Figure 20.10b*



coming from RRG, a typologically motivated linguistic theory, and developing a tree rewriting formalization of it, we end up within a class of formalisms that seems to support Joshi's hypothesis that natural languages are mildly context-sensitive.

20.3.6 Crossing branches and discontinuous structures

The formal syntactic framework introduced so far presupposes *ordered* trees (cf. Section 20.3.1). This means that every two nodes stand either in a dominance relation or a linear precedence relation, and if one node precedes another node then all descendants of the first node precede the descendants of the second node. It follows that crossing branches, which are a frequent phenomenon in RRG syntactic analyses, are not supported by the formalism. Crossing branches often occur with peripheral adjuncts at the nucleus or the clause, for example. Because of the single syntactic tree assumption of the present approach (cf. Section 20.2.2), operators are another potential source of crossing branches. The proposed solution in both cases is to attach the respective elements at a lower node of the phrasal skeleton, in order to avoid a crossing of branches in the tree, while keeping track of their scope by means of appropriate features. Details on how to do this will be given in Sections 20.5 and 20.7.

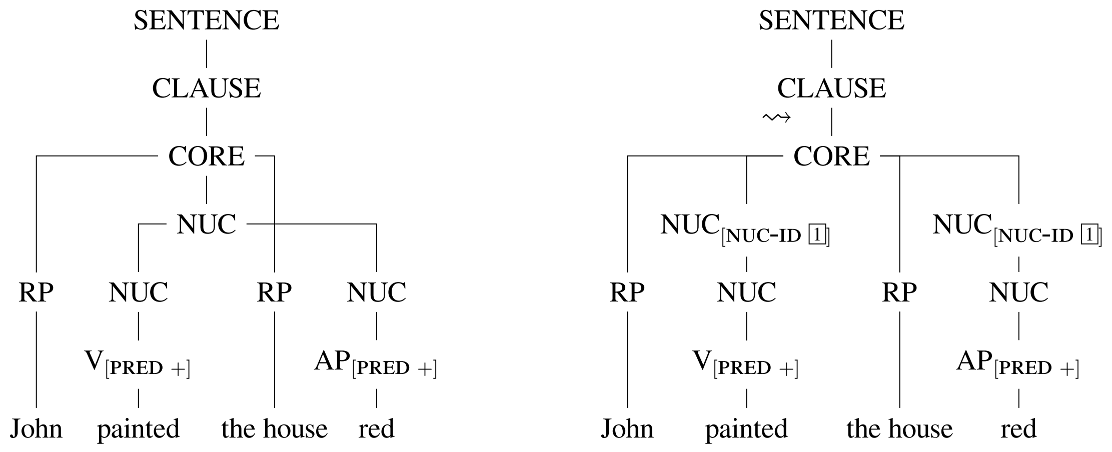
Another reason for crossing branches can be discontinuities in the constituents in the clausal skeleton. Discontinuities in the nucleus may occur with predicates that are multi-word expressions (e.g., particle verbs) or complex predicates (nuclear cosubordinations). An example of the latter is given by the English resultative construction shown on the left of Figure 20.13. In such cases, we assume that the discontinuous node can be split into two components, both carrying the same identifier NUC-ID as a feature (see the right of Figure 20.13), in order to signal that the two nodes are actually components of the same nucleus.[11] Note that the two components would be part of the same elementary tree.

20.4 Specifying the syntactic inventory

20.4.1 The structure of syntactic templates

The syntactic templates are the basic building blocks of syntactic representations. Let us now turn to the question of how to specify the templates of the syntactic inventory in a systematic way.

Figure 20.13. *Discontinuous complex predicates*



The discussion of the two approaches to wh-fronting in Section 20.3.3 shows that the question of which templates must be available in the inventory is not independent of the modes of composition employed. If extraction structures are fully linked locally, as in Figure 20.10b, then there is no need for a precore slot template to be attached separately. As mentioned in Section 20.3.1, the present formalization proposal of RRG syntax assumes that argument structure templates have slots for each of the arguments of the lexical anchor. We call this the *full argument projection* assumption.[12]

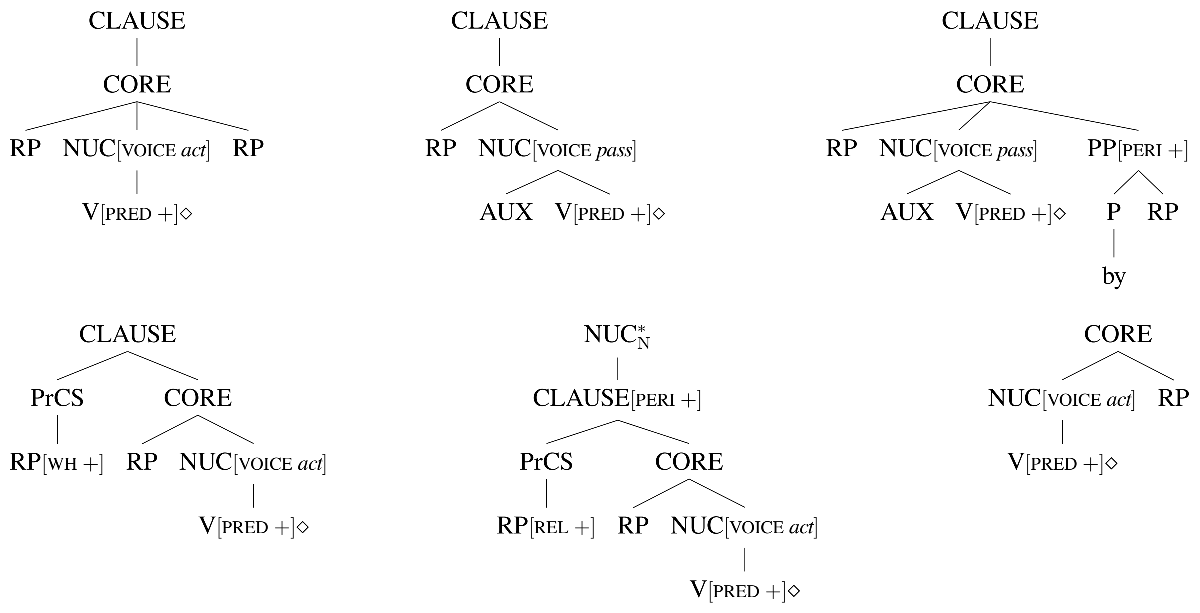
For example, the templates in Figure 20.14 represent (some of the) alternative realization patterns of transitive verbs in English. They would be used for the different forms of *eat* in (6) (in the respective order in which they occur in the figure).

(6) a. The boys ate the entire cake.
b. The cake was already eaten.
c. The entire cake was eaten by the boys.
d. What did the boys eat today?
e. I planned to serve the cake that the boys just ate.
f. I expected them to eat the cake.

Notice the two templates for the passive, one in which only the undergoer occurs as an argument, and a second template which includes the realization of the actor by a peripheral *by*-phrase.[13] Standard RRG would probably regard the peripheral *by*-phrase as a separate template that can be adjoined to the passive core. In the context of the present framework, the question is whether the *by*-phrase should be added by (sister) adjunction like an adverbial. By listing the two realization patterns for the passive as elementary templates in Figure 20.14, we opt against adjunction in this case. The main disadvantage of modeling the *by*-phrase by adjunction would lie in the constraints that need to be imposed, since adding a peripheral *by*-phrase that encodes the actor is restricted to specific constructional circumstances. The optional peripheral *by*-phrase is part of the description of how passive voice is realized in English. Of course, a mere enumeration of the two passive templates in Figure 20.14 is not satisfactory for a theory of grammar. At some point, the theory should state explicitly that it is the addition of the *by*-phrase which relates the second template to the first. In our framework, this relation is encoded at the level of template *specifications* (cf. Section 20.4.2).

The first template in the second row in Figure 20.14 represents an elementary precore slot

Figure 20.14. *Basic transitive predication template for English with variants*



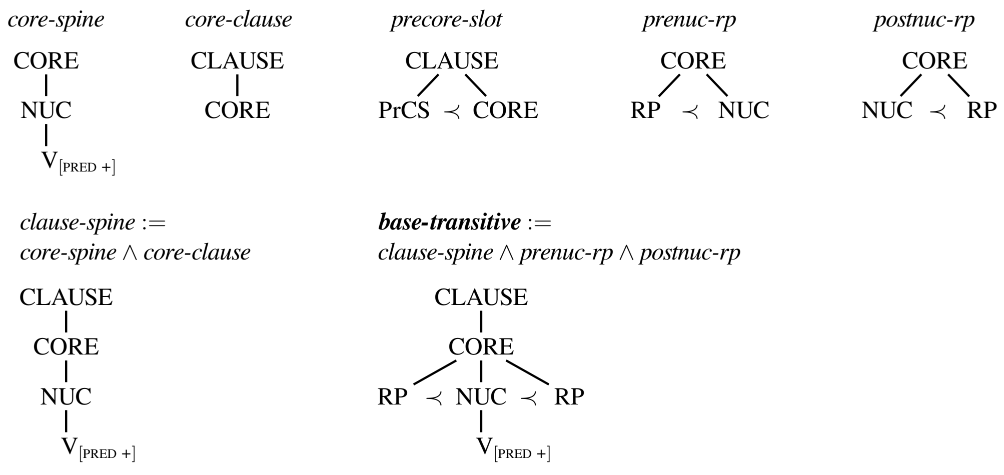
argument structure template. This differs from the template system informally suggested in Van Valin (2005, p. 15), in which the structure in question is composed of a reduced core template and a precore slot-clause fragment (cf. Figure 20.1b). Again, we propose that compositions of this type are best modeled at the level of template specifications. The middle template in the second row is used when the predicate is the anchor of a restrictive relative clause where the relative pronoun is the object. In this case, the peripheral tree is an adjunct tree that is intended to be adjoined to the nucleus of the noun phrase that it modifies. The last template is the one for a subject-less infinitival core, which can be embedded as an argument of category CORE, as in (6f).

20.4.2 Template specification by tree descriptions

In the previous section, we raised the question of how to characterize the (universal and language-specific) syntactic templates in a systematic way. The templates proposed above as elementary are more complex than the fragmentary templates originally assumed in the RRG literature. The key advantage of the formalization approach described here is that the composition of syntactic representations can be reduced to the three modes of composition introduced in Section 20.3, namely simple substitution, sister adjunction, and wrapping substitution. Furthermore, linking can be performed locally within elementary trees (cf. Section 20.8). However, this leaves us with the problem of how to describe in which way the templates are built from more elementary components, and how they are related to each other. As mentioned before, it would be rather unsatisfactory if we had to regard the two passive templates in Figure 20.14 as independent units without being able to make explicit the relation between them.

The proposed solution is to treat syntactic templates as *minimal models* of tree descriptions. Relations between templates can then be captured by relating the respective descriptions. For instance, the specification of the passive template with *by*-PP consists of the specification of the simple passive template plus a specification of the *by*-PP and constraints on its position. The use of tree descriptions for specifying syntactic templates in a modular way is inspired by the *metagrammar* approach of Crabbé and Duchier (2005), where a metagrammar is basically a system of tree descriptions that defines the syntactic inventory as the set of the associated minimal models.

Figure 20.15. *Example specifications of syntactic fragments*



Tree descriptions in this sense consist of dominance and precedence constraints as well as category and feature assignments. Consider the example specifications in Figure 20.15, which are depicted in tree-like diagrams but are to be read as tree descriptions. For instance, the specification with the name *precore-slot* says that there are three distinct nodes n_0, n_1 and n_2 labeled respectively by CLAUSE, PrCS, and CORE, where n_1 and n_2 are daughters of n_0 and n_1 immediately precedes n_2 (expressed by $\prec$). Figure 20.15 illustrates how the basic transitive template of Figure 20.14 can be defined by a piecewise combination of such specifications. The precore slot template of Figure 20.14 can be likewise defined by conjoining the specifications *clause-spine*, *precore-slot*, and *prenuc-rp*. In this way, common components of elementary templates are made accessible in the metagrammar.[14] Furthermore, syntactic tree descriptions can be linked to descriptions of semantic representations (either predicate-logical formulas or frames), and the metagrammar can include a constraint-based formulation of RRG's linking algorithm (cf. Kallmeyer et al., 2016). The syntax-semantics interface will be briefly discussed in Section 20.8.

20.5 Formalizing the operator projection

20.5.1 Operators in RRG

RRG links the representation of operators to the layered structure of the clause. Recall that the layered structure reflects the distinction between predicates, arguments, and non-arguments. The core layer consists of the nucleus, which specifies the verb, and its arguments. The clause layer contains the core as well as extracted arguments. Operators are closed-class grammatical categories such as aspect, modality, and tense. Each type of operator is assumed to attach to a specific layer: for instance, tense operators attach to the clause, modality to the core, aspect to the nucleus (see Table 20.1 for the mapping between operator types and corresponding layers). Moreover, the surface order of the operators reflects their attachment site in that the higher the layer an operator is attached to, the further away from the nucleus the operator occurs on the surface. The mapping from operators to levels of the layered structure explains (i) the scope behavior of operators, since structurally higher operators take scope over lower ones, and (ii) surface order constraints for operators; higher operators are further away from the nucleus of the structure.

The problem is that the constituent and the operator structure are not completely parallel, i.e., one can have structures where an operator belonging to a specific layer is, on the surface level, surrounded by elements belonging to a lower layer in the constituent structure. Examples of

Table 20.1. *Operators in the layered structure of the clause (cf. Van Valin, 2005, p. 9)*

Layer	**Operators**
Nucleus	Aspect Negation Directionals
Core	Directionals Event quantification Modality Negation
Clause	Status Tense Evidentials Illocutionary Force

a clause-level tense operator occurring within the core are given in (7) and (8) (taken from Van Valin, 2005, p.10) for English and Turkish, respectively.

(7) [Mary enter-ed$_{\text{TNS}}$ the room]$_{\text{CORE}}$.

(8)

[Gel-	emi-	yebil-	ir-	im]$_{\text{CORE}}$.
come-	ABLE.NEG$_{\text{MOD}}$-	PSBL$_{\text{STA}}$-	AOR$_{\text{TNS}}$-	1SG

‘I may be unable to come’

In (8), the clause-level status and tense operators occur between the verb and the pronominal affix, which is part of the core.

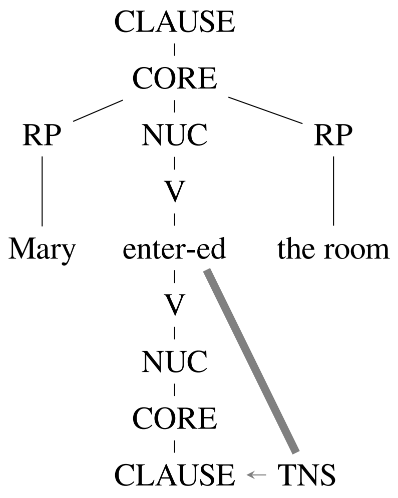
Even though the constituent structure and the operator structure are not fully aligned, they depend on each other. Their hierarchical order is the same and the existence of a layer in the operator projection requires that this layer also exists in the constituent structure. For instance, one can only have clause-level operators if a clause node exists in the constituent structure.

While the ordering among the operators is thus systematically correlated with the scope given by their attachment site at the clausal skeleton, the surface order of the operators relative to arguments and adjuncts is much less transparent and would require crossing branches if everything were captured in a single tree with operators attaching where they take scope. For this reason, RRG usually represents the constituent structure and the operator structure as different projections of the clause. The syntactic representation for (7) on the left side of Figure 20.16 illustrates this idea: The upper part gives the constituent projection while the lower part gives the operator projection. If we integrate the operator projection into the clausal skeleton of the constituent structure, we obtain the tree with crossing branches shown on the right of Figure 20.16, due to the above-mentioned mismatches between the two projections.[15]

Besides crossing branches, additional complications arise from the following two facts: First, a functional element can contribute more than one operator and, second, an operator can be distributed over more than one element in the sentence. An example is (9) where the functional element *will* contributes both tense and illocutionary force (IF), while aspect (ASP) is jointly contributed by *be* and *-ing*.

(9) Will$_{\text{IF,TNS}}$ they be$_{\text{ASP}}$ leav-ing$_{\text{ASP}}$

Figure 20.16. *Constituent structure and operator projection for (7)*



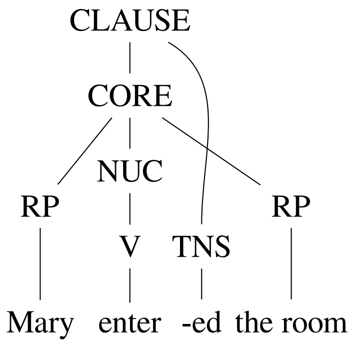
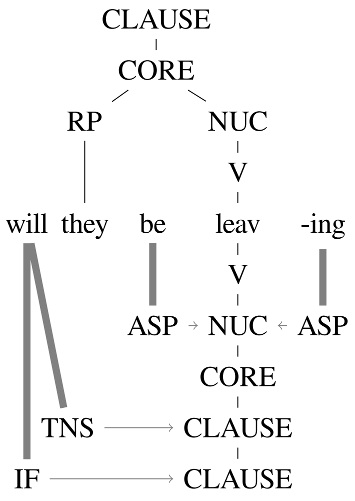
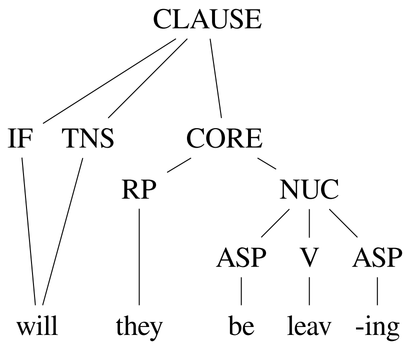


Figure 20.17. *Constituent structure and operator projection for (9)*





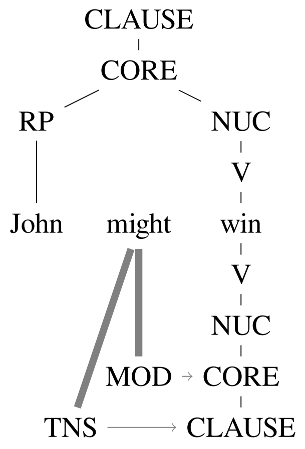
RRG assumes the structure on the left of Figure 20.17 (adapted from Van Valin 2005, p.14). If the operator projection is integrated into the constituent structure, we obtain the graph on the right of Figure 20.17, which is not even a tree but only a directed acyclic graph since the node labeled *will* has two incoming edges. Note, however, that as long as nodes with more than one incoming edge arise only from contributing several operator categories to the same level of the layered structure, the graph can easily be turned into a tree by putting all these categories into a single node with a label (or, rather, a feature structure) that captures the fact that the element is an operator that contributes several categories to the same layer. The node dominating *will* for instance in Figure 20.17 could have a category OP with features TENSE $=$ *fut* and IF $=$ *int*.

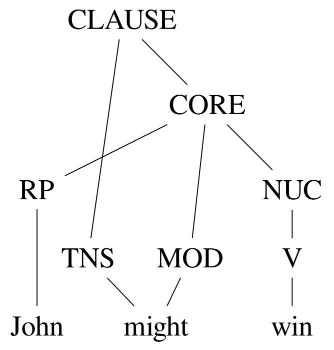
Cases with a single element contributing two operators that take scope at different layers exist as well. Examples are finite modal verbs as in (10) that provide modality at the core level and tense at the clause.

(10) John might$_{\text{TNS,MOD}}$ win.

The corresponding constituent structure and operator projection in RRG is shown on the left of Figure 20.18, and on the right we see the graph that one would obtain when integrating the operator

Figure 20.18. *Constituent structure and operator projection for (10)*





projection into the constituent tree. This example shows that if we integrate the operator projection into the constituent tree while assuming that each operator has a separate node that is the daughter of its syntactic layer, we obtain directed acyclic graphs instead of trees, which might come with considerable complications concerning the formal properties of such a framework. In particular, it is clear that such structures cannot be generated with the TWGs introduced in the previous chapter.

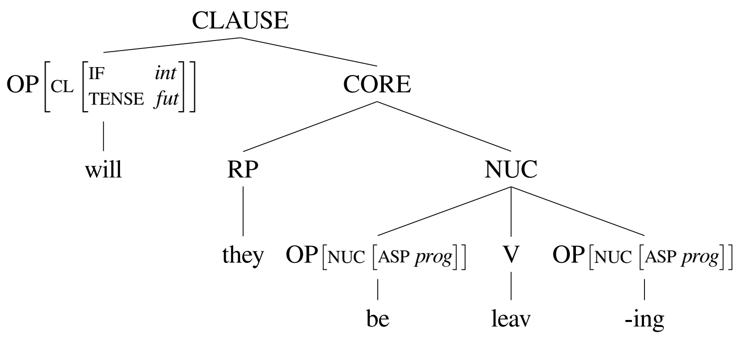
If we choose to disentangle the operator projection to a certain extent from the constituent structure, we have two options. Either we can generate the operator projection tree separated from but to a certain extent in parallel with the constituent structure. This is what Johnson (1987) proposed in his formalization which uses two different context free grammars, one for analyzing the sequence consisting of the verb plus arguments, and one for the sequence consisting of the verb plus operators (cf. Section 20.2.1). Each takes those parts of the sentence as input (or yield) that correspond to its set of terminals. This approach makes sure that verb and arguments on the one hand and verb and operators on the other hand appear in the right order, independent from each other. But there are some shortcomings: First, the formalization does not guarantee that each layer targeted in the operator projection is actually present in the constituent structure. Secondly, a problem remains with elements that contribute more than one operator, possibly to more than one layer, since the different projection grammars are each assumed to generate trees.

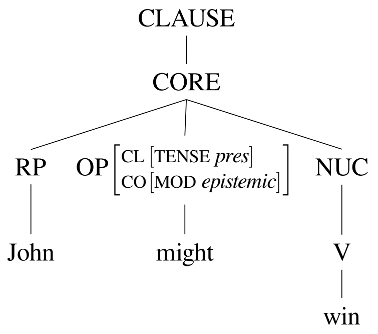
20.5.2 A feature-based implementation of the operator projection

In the following, we will present the proposal of Kallmeyer and Osswald (2017), which integrates operators according to their surface position into the constituent structure, thereby avoiding crossing branches, while keeping track of the operator projection within the feature structures. The classification of an element as an operator and the contribution of that element to the various layers is captured within the category of its node. More specifically, an operator element has the category label OP and a feature structure that specifies its contribution to the CLAUSE (feature CL), the CORE (feature CO) and the NUC (feature NUC) layer. The tree structures corresponding to the sentences (9) and (10) are shown in Figure 20.19. Note that the operator scope information is now entirely captured within the feature structure at the OP node and does not depend on the category of the node to which the OP attaches.

In order to obtain trees of the type given in Figure 20.19, we use sister-adjunction for adding operators. This leads to spurious ambiguities since operators could in principle adjoin to any of the three layers that they attach to, provided this does not yield crossing branches. In the first tree

Figure 20.19. *Encoding the operator structure in node features for (9) and (10)*





in Figure 20.19 for instance, we could also attach the clausal operator *will* lower, at the CORE node, and *be* could attach higher, also at the CORE node. In the following we assume that among the different combinations of attachment sites that are possible, the preferred one is the one where operators are placed as close as possible to the layer that corresponds to their scope (if it scopes at several layers, the lowest is considered relevant).

Aside from this spurious ambiguity issue, if we adjoin operators in an unrestricted way, it leads to undesired overgeneration since we can leave out operators that are obligatory, adjoin several operators of the same type even in cases where this is ungrammatical, and we can generate ungrammatical linear orders of operators, for instance the ones in (11).

(11) a. *They be will leaving?
b. *John not might win.

Kallmeyer and Osswald (2017) introduce edge features in order to express the necessary constraints on adjunction. The basic idea is that each edge (or, under a different but equivalent view, each node) should be able to pass information on elements already attached and elements still required to the left and to the right. Once the derivation is completed, the edge information of sister nodes must be compatible with each other. To this end, each node (or, as Kallmeyer and Osswald (2017) put it, each edge) has a left and a right feature structure that are not part of the proper feature structure of the node but interact with the left and right feature structures of neighbouring nodes. They are graphically depicted in shaded boxes to the left and the right of the node in question. (The feature structure of the node itself will often be omitted or, if needed, given in the middle.) Note that the features NUC, CO and CL mentioned above and exemplified in Figure 20.19 are not part of the edge features but they are proper node features. Edge features are less about the actual content of nodes but rather about requirements and information that needs to be passed around in order to constrain syntactic composition.

Let us illustrate this approach by using it to characterize a certain operator as obligatory. In Figure 20.20, we have a verb without tense marking. Therefore, to the left of the NUC node, we

Figure 20.20. *Obligatory adjunction of a tense operator*

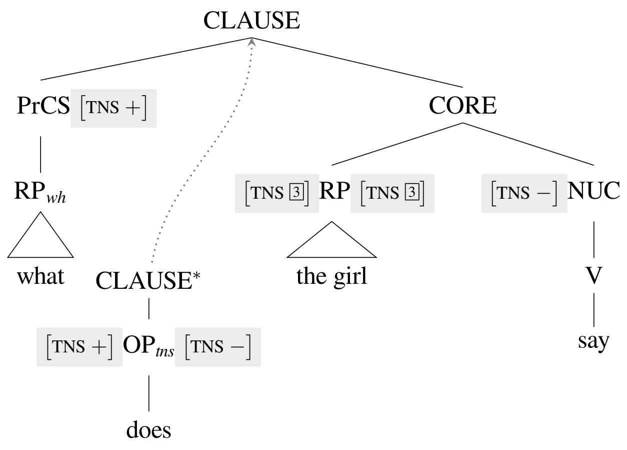
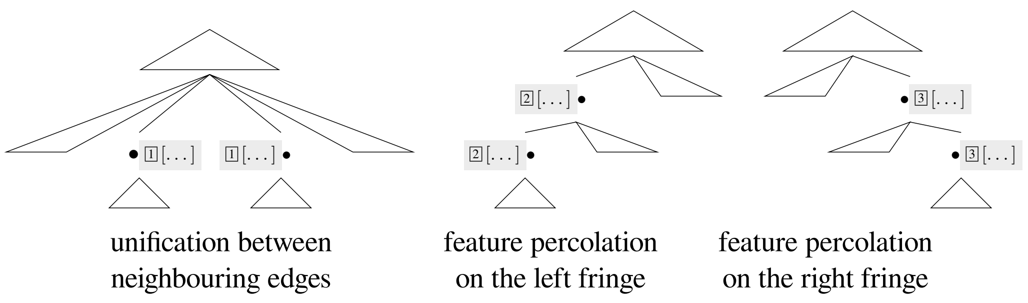


Figure 20.21. *Final feature unifications*

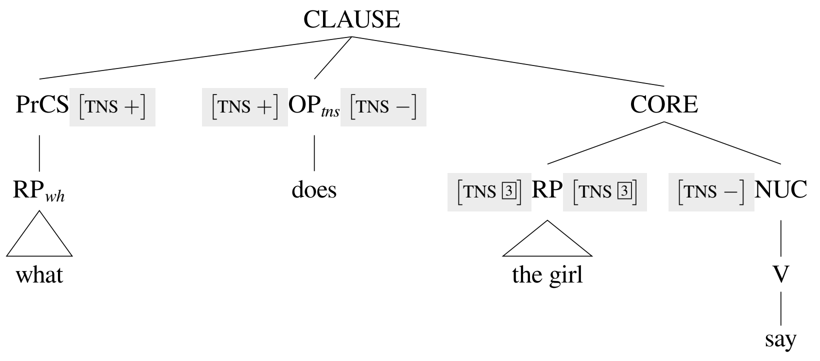

unification between neighbouring edges
feature percolation on the left fringe
feature percolation on the right fringe

signal [TNS−]. The features we are using here are mostly binary features (i.e., values are only + or −), and their intended meaning is that a value − expresses the absence of something while a value + expresses the presence or the requirement (presence in the derived tree) of something. The feature [TNS+] on the right of the PrCS signals that to the right of this node, a tense operator has to be present. If no tense operator adjoins, the absence and the requirement of the presence will eventually unify (we will explain further down why) and this will lead to a unification failure. If, however, a tense operator adjoins as in Figure 20.20, the requirement is satisfied. The features to the left and to the right of the tense operator tell us that to its right, no other tense operator exists while to its left one can pass the information that there now is a tense operator. (OP$_{tns}$ is short for category OP with feature structure [CL [TENSE....]].)

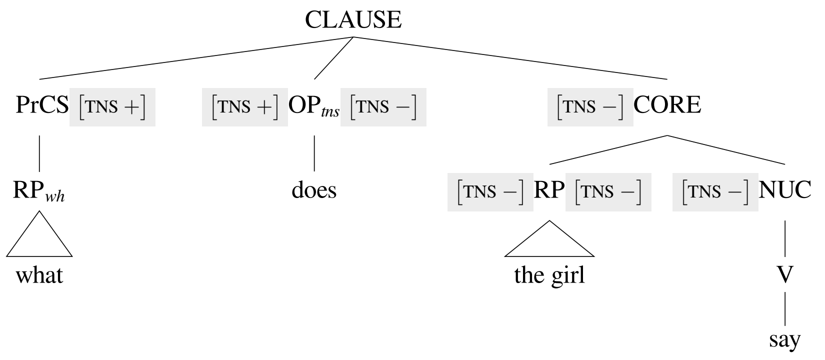
Let us explain now how exactly edge features work (see Kallmeyer and Osswald, 2017), in particular how they percolate through the tree (cf. Figure 20.21). As mentioned, nodes can have special left and right feature structures. In the final derived tree, the left feature structure of a node *v* unifies with the right feature structure of its immediate sister to the left. Furthermore, the left feature structure of a node *v* that does not have a sister to the left unifies with the left feature structure of the mother of *v*, provided this mother is not the root node of an elementary tree or the lower node of a d-edge. Similarly, the right feature structure of a node *v* that does not have a sister to the right unifies with the right feature structure of the mother of *v*, again provided this node is not the root node of an elementary tree or the lower node of a d-edge. In our example, once we have performed the adjunction, we obtain the tree at the top of Figure 20.22, and after the final edge feature unifications, the result is the tree at the bottom. Such edge features can be used to require certain adjunctions and, as is the case in this example, to require them exactly once.

Figure 20.22. *Obligatory adjunction of a tense operator: derived tree*

before final unifications:



after final unifications:



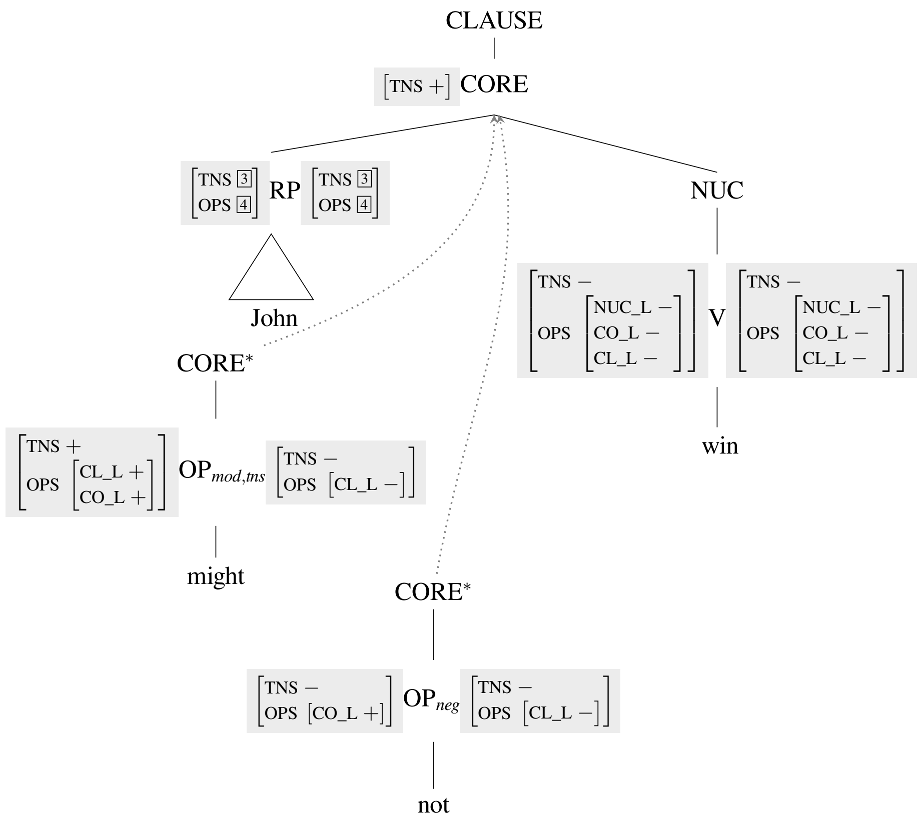
The adjunction of the operators needs to be controlled in the following two respects: (i) Adjoining an operator is *obligatory* if the information conveyed by the operator is required for a sentence to be complete. (ii) The scope-related *ordering* of the operators must be respected. In our approach, these constraints are both implemented with the help of edge feature structures; we have just seen an example of type (i). We will now introduce features that guarantee (ii) as well. Concretely, we want to achieve that, from the nuclear predicate outwards, one encounters first the NUC operators, then the CORE operators and then the CLAUSE operators. To this end, we assume an edge feature OPS (for operator structure) that specifies which operator projection layer(s) have been reached so far. Its value is a feature structure with features CL_L, CO_L and NUC_L for the three layers, each with possible values + or −. These features are used in such a way as to guarantee that nuclear, core and clausal operators have to appear in this order when moving outwards in the sentence, starting from the nuclear predicate.

Figure 20.23 illustrates this mechanism by applying it to the operator structure of the sentence in (12).

(12) John might not win.

For instance, a core operator such as *not* that adjoins to the left of the predicate has a requirement to the right that the level CL_L have a value −, i.e., is not reached yet. To the left, it just gives the information CO_L +, which has the effect of disallowing nuclear operators to appear here. The operator *might*, which contributes not only tense at the clausal level but also modality at the core level, comes with the same requirements to its right. But, in contrast to *not*, it states to the left of the edge that now CL_L has value + (as well as CO_L). An inverse order (*John not might win*) is therefore excluded. To the left of *might*, there cannot be any further core (or nuclear) operators.

Figure 20.23. *Keeping track of the operator projection in edge features*



CLAUSE
[TNS +] CORE
[TNS 3, OPS 4] RP [TNS 3, OPS 4]
John
NUC
[TNS −, OPS [NUC_L −, CO_L −, CL_L −]] V [TNS −, OPS [NUC_L −, CO_L −, CL_L −]]
win
CORE*
[TNS +, OPS [CL_L +, CO_L +]] OP mod,tns [TNS −, OPS [CL_L −]]
might
CORE*
[TNS −, OPS [CO_L +]] OP neg [TNS −, OPS [CL_L −]]
not

The derived tree for sentence (10) is shown in Figure 20.24.

Recall that in addition to the OPS feature, nodes of category OP have features NUC, CO and CL that indicate the contribution of the operator. For example, since *might* contributes tense at the clause level and epistemic modality at the core level, its OP feature structure is [CL[TENSE *pres*], CO[MOD *epistemic*]] (cf. Figure 20.19). (We left this out in Figures 20.23 and 20.24 for the sake of readability.) Since the features of the OP nodes specify which operator projection layer(s) the operator belongs to, we can deterministically map a derived tree to the standard RRG structure in which the constituent structure and the operator projection are separated.

20.6 The construction of complex sentences

20.6.1 Coordinate, subordinate and cosubordinate constructions

A crucial assumption of RRG concerning the structure of complex sentences is the distinction between embedded and non-embedded dependent structures. Embedded dependent structures correspond to *subordinations*. By contrast, non-embedded dependent structures, which are referred to as *cosubordination* structures, have basically the form $[[\ ]_X\ [\ ]_X]_X$. It is characteristic of this type of construction that operators that apply to category X are realized only once but have scope over both constituents. Cosubordination differs from the *coordination* of two independent structures in that the latter type of construction has the form $[[\ ]_X\ [\ ]_X]_Y$, where Y is a category one level above X in the layered structure. The general schemas for non-subordinate structures are shown in Figure 20.25 (cf. Van Valin and LaPolla 1997, p. 507 and Van Valin 2005, p. 224). However, these structures cannot serve as elementary trees in the sense introduced in Section 20.3 since there are infinitely many of them; that is, they have to be derived by operations in the syntax.

Figure 20.24. *Derived tree for (10) before (top) and after (bottom) final edge feature unification*

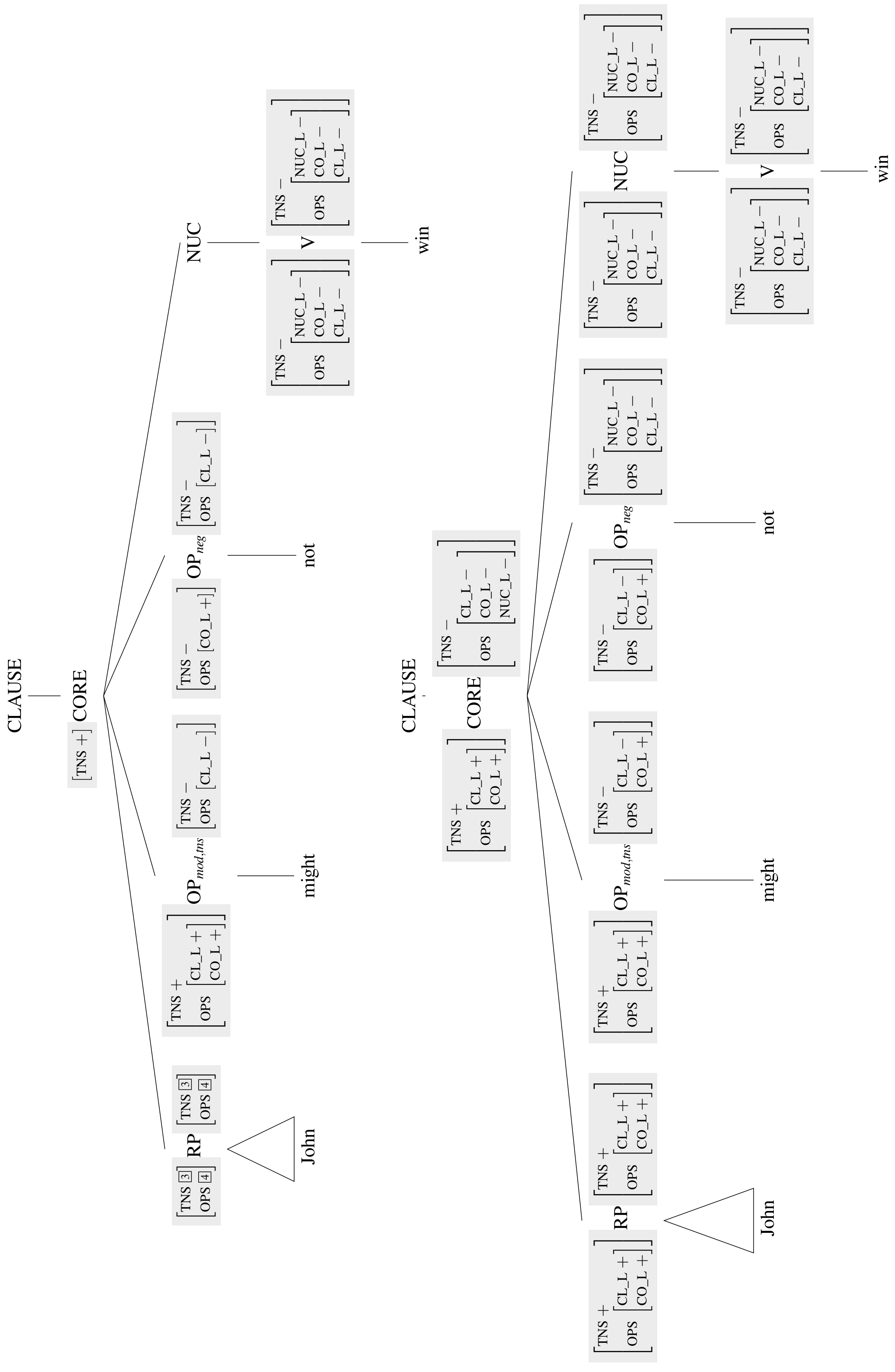


Figure 20.25. *RRG template schemas for non-subordinate nexus types*

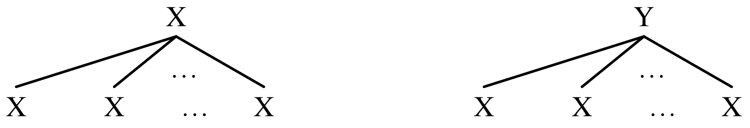
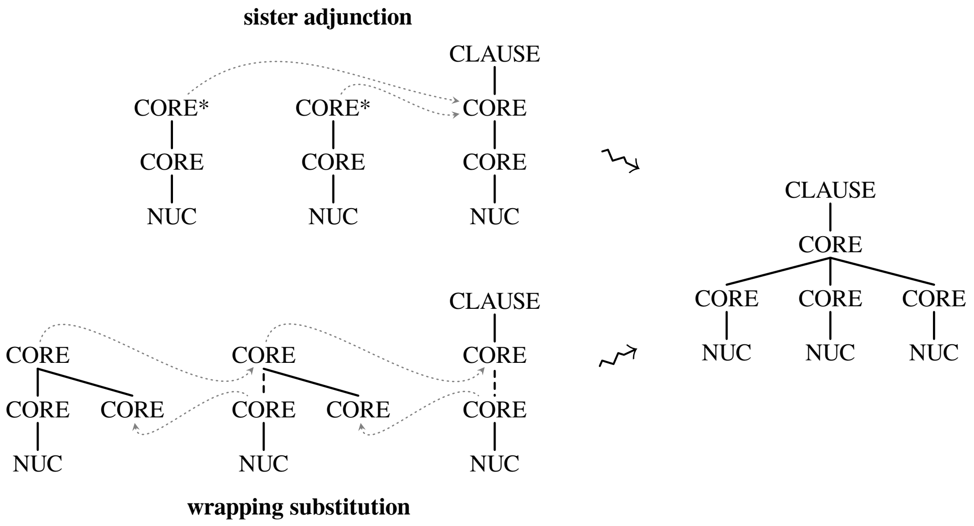


Figure 20.26. *Two ways of compositionally deriving cosubordination constructions*



Among the tree composition operations described in Section 20.3, both sister adjunction and wrapping substitution allow the derivation of cosubordinate structures, as sketched in Figure 20.26 for the case of core cosubordination. The following two examples of multi-verb constructions show possible use cases for the two kinds of derivations. Both examples describe transitive motion scenarios. The example in (13), taken from Ullrich (2011), illustrates a common pattern for simultaneous event constructions in Lakhota (Siouan) (cf. Ullrich, 2018; Osswald and Van Valin, 2021).

(13)	Yu-slóhaŋ	a-wíčha-∅-ye.
	by.pulling-slide	AM-3PL.UG.ANIM-3SG.AC-go
	‘She was dragging them away.’	

In Lakhota, arguments are marked by pronominal affixes at the main verb at the end of the clause. The simultaneous event construction in (13) contains a second verb that precedes the main verb and is dependent in that it is morphophonologically reduced and does not carry any pronominal markings. The construction has the properties of a core cosubordination (Ullrich, 2018). Since there are no shared arguments to be taken care of syntactically, due to the head marking, and because the construction has the same-subject/same-object constraint, the cosubordinate structure can be derived most naturally by sister adjunction, as illustrated in Figure 20.27. Under this analysis, the main verb selects an elementary tree that has already two core nodes and requires the adjunction of an additional core daughter. This requirement can be captured via a binary edge feature COSUB which signals that a structure is a cosubordination structure. In addition, the higher CORE node of a CORE cosubordination can have a feature [COSUB +] while the lower ones are marked [COSUB −], which ensures that the adjunction takes place at the upper CORE node. Note that these node features do not interact with the edge features.

The Japanese example in (14) (from Croft et al. 2010, p. 219) shows the use of the *te*-construction for combining cause, manner and direction of motion.

Figure 20.27. *Core cosubordination via sister adjunction: analysis of (13)*

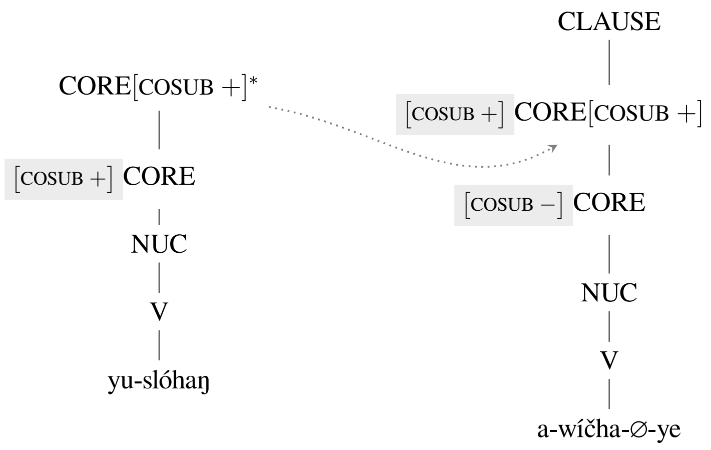
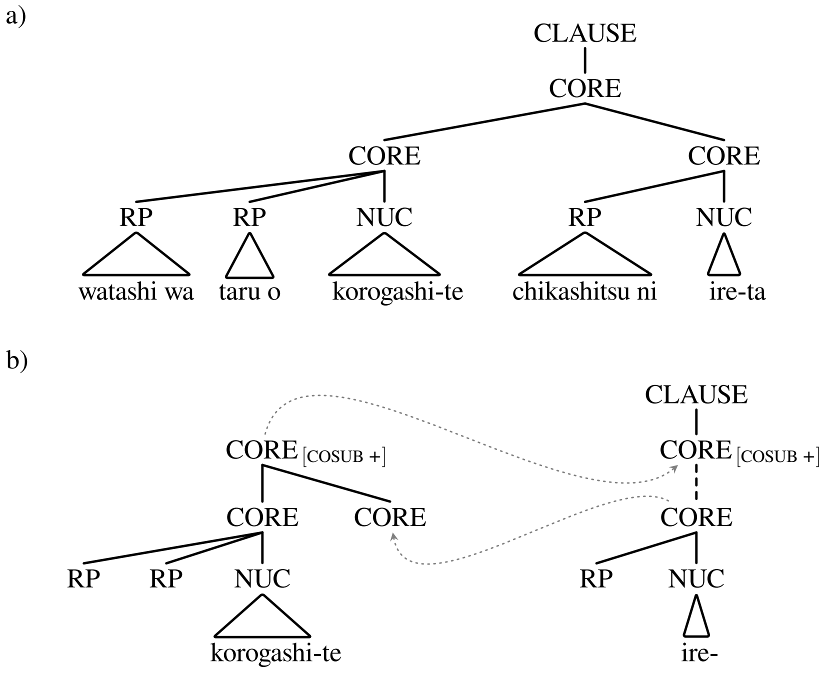


Figure 20.28. *Core cosubordination analysis of (14) derived by wrapping substitution*

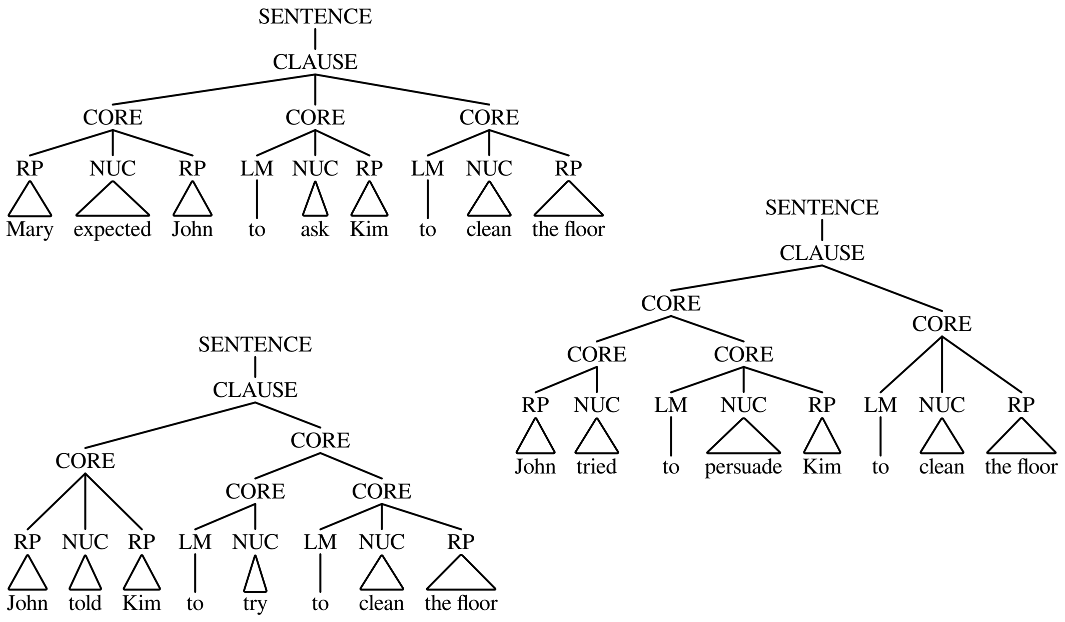


(14) Watashi wa taru o korogashi-te chikashitsu ni ire-ta.
1SG TOP barrel ACC roll(CAUS)-TE basement LOC take.into-PAST
'I rolled the barrel into the basement.'

Assuming that (14) is also an instance of a core cosubordination construction, the derivation seems best described as an application of wrapping substitution in this case (cf. Figure 20.28). Under this analysis, the first two arguments are introduced by the first verb and are shared by the second verb. The elementary tree chosen for the second verb has, therefore, a reduced number of arguments. Its remaining arguments are "controlled" by the elementary tree of the *te*-marked first verb, like in a control construction.

Multiply embedded control and matrix-coding constructions pose a number of challenges for RRG's syntactic analysis as presented in Van Valin and LaPolla (1997) and Van Valin (2005). Consider the examples in (15) and their syntactic representations in Figure 20.29.

Figure 20.29. *Syntactic representations of the examples in (15)*


SENTENCE
CLAUSE
CORE
RP
NUC
LM
Mary expected John to ask Kim to clean the floor
John tried to persuade Kim to clean the floor
John told Kim to try to clean the floor

(15) a. Mary expected John to ask Kim to clean the floor.
b. John tried to persuade Kim to clean the floor.
c. John told Kim to try to clean the floor.

Structures like the ones in Figure 20.29 show a mismatch between syntax and semantics in the following sense: semantically, the infinitival complements are arguments of the respective control predicates, but syntactically, they do not behave like core arguments. Example (15a) is instructive in showing how multiply embedded constructions can give rise to coordinating syntactic configurations. Simple substitution cannot be applied here if we assume that the verbs *expect* and *ask* select elementary templates that contain a core daughter of the clause for the infinitival complement. Figure 20.30a sketches a possible solution that uses wrapping substitution, where split nodes carry differing top and bottom categories. Figure 20.30b shows that the same compositional mechanisms works for control verbs such as *try* which call for a core cosubordination template. Moreover, the treatment of wh-fronting introduced above straightforwardly extends to embedded control and matrix-coding constructions like those in (16).

(16) a. Whom did Mary expect John to ask to clean the floor?
b. What did Mary expect John to ask Kim to clean?

For example, the syntactic representation of (16a) can be composed of elementary argument structure templates as illustrated in Figure 20.31.

Note that clausal complements as in (17a) show also a mismatch between syntax and semantics in that they do not attach to the core but to the clause. But in contrast to the cosubordination cases just discussed, multiple embeddings of clausal complements correspond to embeddings on the syntactic side, though at the clause level and not at the core level. For this reason, simple substitution is sufficient for clausal complementation (cf. Figure 20.32a).

(17) a. John thinks (that) Kim smashed the vase.
b. What does John think (that) Kim smashed?

Figure 20.30. *Composition of templates by wrapping substitution for (15a) and (15b)*

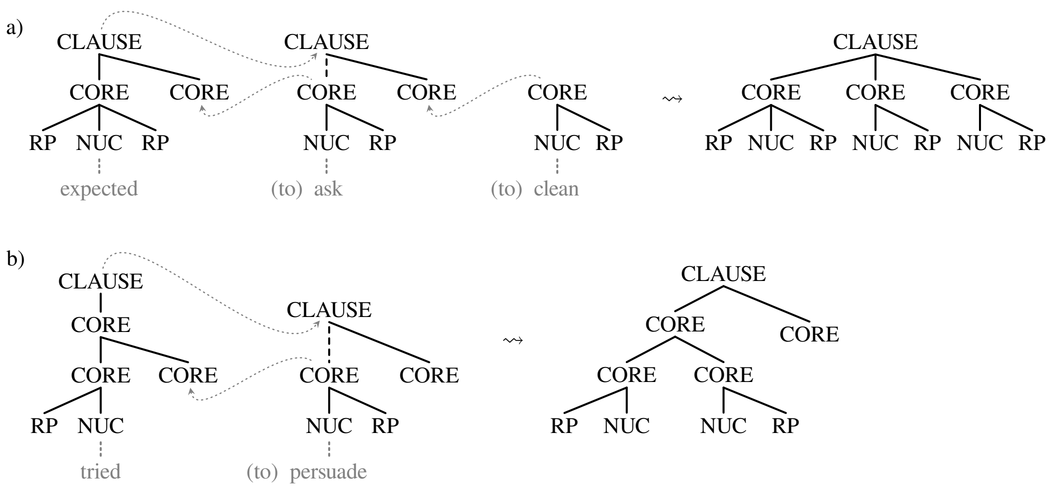
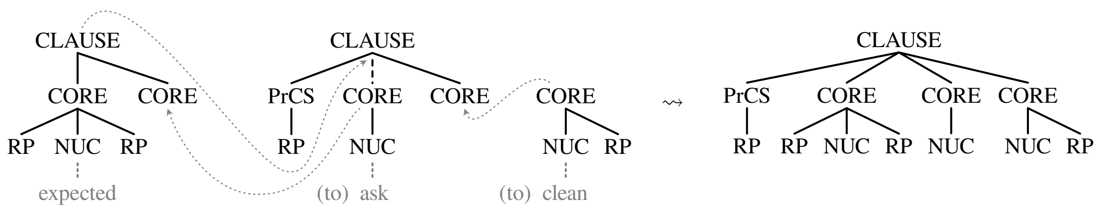


Figure 20.31. *Composition of the syntactic representation of (16a)*



However, as already discussed in Section 20.3.3, cases of extraction out of subordinated constituents, like in (17b), require wrapping substitution (cf. Figure 20.32b).

20.6.2 Operators in complex sentences

Operators in cosubordination constructions that appear in one of the components and that target the cosubordination layer have scope over all components. In the following, we will illustrate how this characteristic property of cosubordination can be formally achieved, going through one example involving sister adjunction and one example involving substitution.

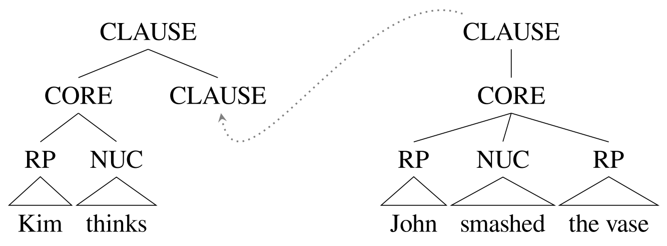
The Turkish sentence in (18) (taken from Van Valin, 2005, p. 201) is an example of a core cosubordination construction (see also Bohnemeyer and Van Valin, 2017, p. 155f). On the surface, the deontic modal operator *-meli* ('should, ought to') is embedded in the second core, but it takes scope over the entire complex core.

(18) [[Gid-ip]$_{\text{CORE}}$ [gör-meli-yiz]$_{\text{CORE}}$]$_{\text{CORE}}$.
go-LM see-MOD-1PL
'We ought to go and see.'

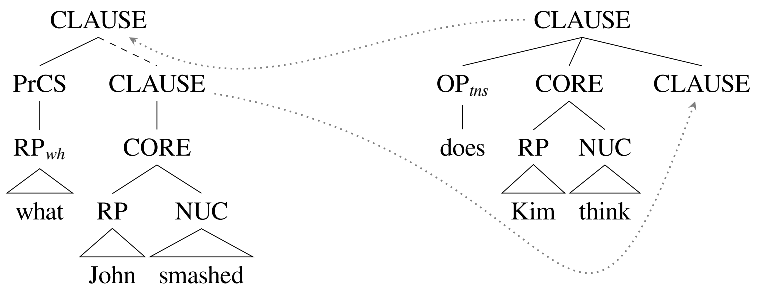
Let us assume that the first core is added to the second core by sister adjunction, similar to the analysis of the Lakhota construction in Figure 20.27. The modal operator in (18) adjoins to the second embedded core node and carries a feature indicating that it is a core operator. The result should be the derived structure in Figure 20.33 with the MOD feature shared between all the three CORE nodes involved (cf. Van Valin, 2005, p. 204).[16] In a cosubordination, an operator embedded in one part of the complex structure generally takes scope over the larger category.

Figure 20.32. *Subordination via simple and wrapping substitution for (17a) and (17b), respectively*

a)



b)



Accordingly, in all elementary trees for cosubordination configurations, the relevant features (here MOD) are shared between the lower and the higher category in question (here the two CORE nodes). This is taken to be a general property of cosubordination structures. Corresponding to this, we assume that when mapping our derived structure to the standard RRG structure, the operator targets the highest corresponding node, as long as there is no higher operator level and no substitution node in between. In the case of Figure 20.33, this is the core of the entire sentence.

Figure 20.34 gives the derivation of the tree in Figure 20.33, attaching the tree for *gid ip* by sister adjunction. Note that this implies that there is a special tree for cores such as *gör yiz* in this example, providing two CORE nodes and requiring an adjunction at the higher one in order to be completed to a cosubordination structure. This requirement is again expressed by an edge feature COSUB. We can adjoin several cores such as *gid ip* but we have to adjoin at least one in order to switch the feature from $-$ to $+$. Besides the edge feature, we also assume (as before, see Figure 20.27) a boolean node feature COSUB that expresses whether a node roots a cosubordination structure or not (see also the CORE nodes in Figure 20.34). This enforces an adjunction of the *gid* tree at the cosubordination CORE node and not at the lower CORE node.

A CORE cosubordination example that involves wrapping substitution instead of sister adjunction is the English example in (19a) (cf. Van Valin, 2005, p.203) where we have a core consisting of three embedded core constituents where the first contains the modal operator *must*. This operator takes scope over the entire large core. By contrast, in (19b) we have a structure consisting of two cores which constitute a clause. That is, we have a core coordination and not a core cosubordination. In this case, the modal embedded in the first core scopes only over this one and not over both cores.

(19) a. [[Kim must$_{MOD}$ go]$_{\text{CORE}}$ [to try]$_{\text{CORE}}$ [to wash the car]$_{\text{CORE}}$]$_{\text{CORE}}$

Figure 20.33. *Derived tree for (18)*

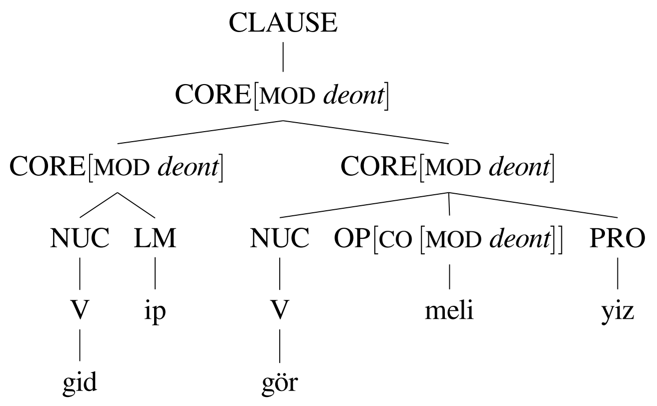
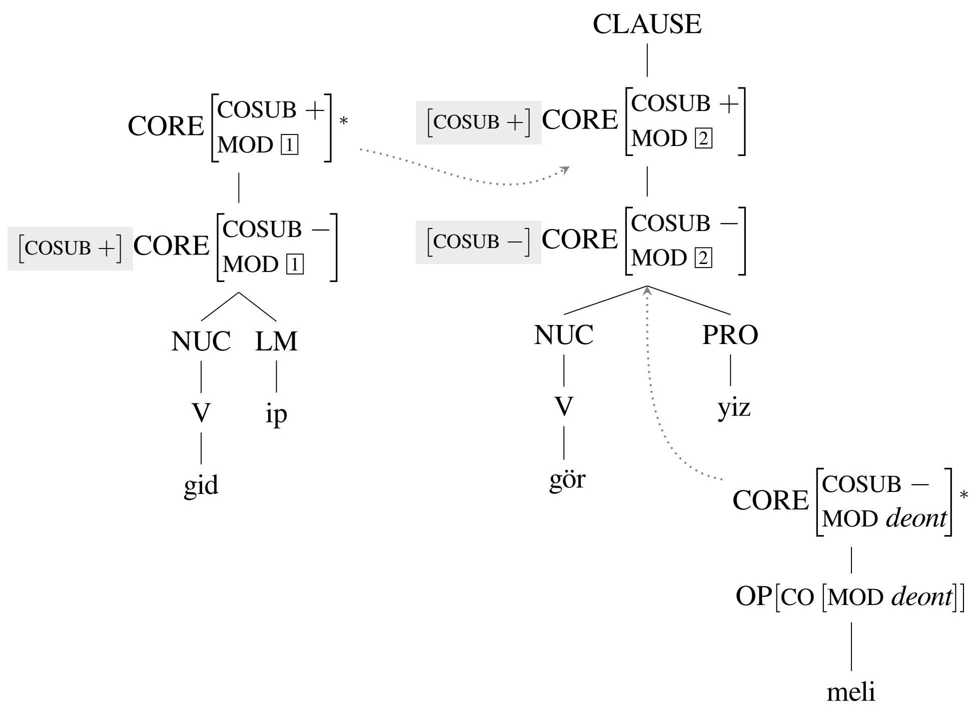


Figure 20.34. *Derivation for (18)*

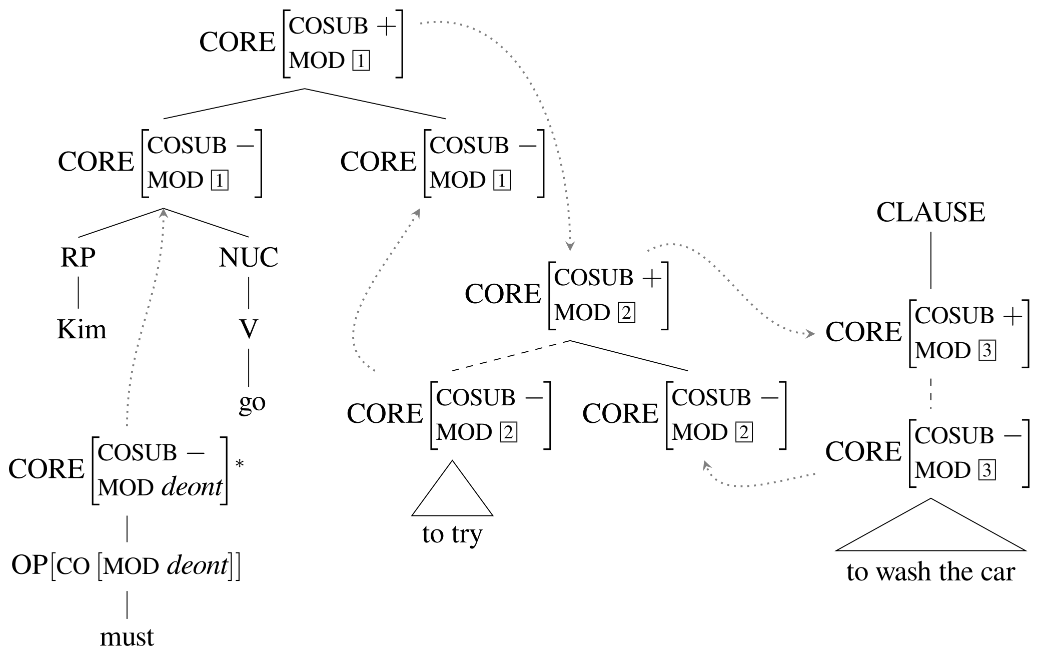

CORE [COSUB + MOD 1]*
[COSUB +] CORE [COSUB − MOD 1]
NUC
LM
V
ip
gid
CLAUSE
[COSUB +] CORE [COSUB + MOD 2]
[COSUB −] CORE [COSUB − MOD 2]
NUC
PRO
V
yiz
gör
CORE [COSUB − MOD deont]*
OP[CO [MOD deont]]
meli

b. [[Kim must$_{MOD}$ ask Pat]$_{\text{CORE}}$ [to wash the car]$_{\text{CORE}}$]$_{\text{CLAUSE}}$

Concerning the analysis of (19a), we propose that the different cores are combined with each other via wrapping substitution and not via sister adjunction. The difference, compared to (18) above is that the verbs *go* and *try* both syntactically select for an infinitival complement clause, i.e., for a CORE argument, which should be expressed via a corresponding substitution node. The derivation is shown in Figure 20.35. The node features contributed by CORE operators (here [MOD *deont*]) are shared between the different CORE nodes that are part of the cosubordination construction. An edge feature for enforcing the adjunction of further CORE sisters in a cosubordination is not necessary since the substitution nodes guarantee that CORE arguments have to be added.

The shared operator scope in (19a) is a standard criterion for distinguishing cosubordinate from coordinate constructions. Another diagnostic is the independent accessibility of the embedded cores by time-positional adverbials, which are analyzed as core-level modifiers (cf. Bohnemeyer and Van Valin, 2017). While (19b) does allow independent time-positional modification,

Figure 20.35. *Derivation for (19a)*



as in (20a), this is not an option for (19a): Both, (20b) and (20c) are excluded.

(20) a. Kim must ask Pat now to wash the car tomorrow
b. #Kim must go now to try to wash the car tomorrow
c. #Kim must go to try now to wash the car tomorrow

As to operators in subordinated CORE or CLAUSE arguments, since these arguments are added by substitution, their root nodes block edge feature percolation and we can have different operators within the argument and outside. In (21) for instance (adapted from Van Valin, 2005, p. 200), the tense operator *will* is part of the argument CLAUSE added by substitution, and it does not percolate upwards to the matrix clause.

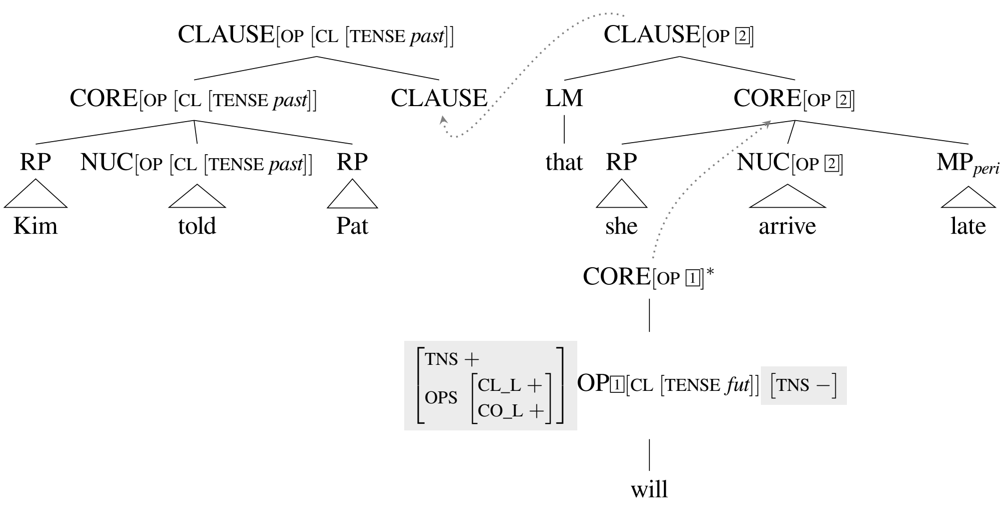
(21) Kim told Pat that she will arrive late.

The substitution of the clausal argument into the tree anchored by *told* is shown in Figure 20.36. The operator *will* in the embedded clause contributes tense at the clausal layer. The corresponding feature [CL [TENSE *fut*]] is shared along the clausal skeleton of the embedded sentence, via the [OP $\boxed{2}$] features. But it is not transported into the embedding clause, which can have a different tense feature. The edge feature percolation also stops at the substitution node, which means that the tense in the embedded clause cannot satisfy a tense requirement in the embedding clause.

20.7 Modification and Periphery

Adjuncts in RRG are part of the periphery, which can be seen a structure similar to the operator projection since each adjunct targets a specific layer and, starting from the nucleus and moving outside, the nuclear adjuncts have to precede the core adjuncts, which in turn have to precede the clausal adjuncts. Kallmeyer and Osswald (2017) briefly mention that this can be modeled in a way similar to the treatment of operators explained above. In the following, we will illustrate with an example how to capture the ordering constraints for periphery elements via edge features in a way analogous to the corresponding word order constraints for operators. We will restrict ourselves to

Figure 20.36. *Derivation for (21)*



adverbs but other modifiers would be treated in a similar way.

According to (Van Valin, 2005, p.41), adverbs may "modify all three layers of the clause; aspectual adverbs like *completely* and *continuously* modify the nucleus, pace adverbs like *quickly* and manner adverbs like *carefully* modify the core, and epistemic adverbs like *probably* and evidential adverbs like *evidently* modify the clause." Depending on their position, manner adverbs can actually be clausal or core modifiers, see (22), from Van Valin (2005).

(22)	a.	Ruth cleverly hid the cash.	(ambiguous)
	b.	Ruth hid the cash cleverly.	(core modifier)
	c.	Cleverly, Ruth hid the cash.	(clause modifier)

The core modifier reading signifies that Ruth hid the cash in a clever way while the clausal modification has the meaning that it was clever of Ruth to hide the cash. (22a) is ambiguous between the two readings while (22b) seems to have only the "in a clever way" reading and (22c) only the wide scope reading ("it was clever to hide the cash").

As to linear precedence, the main difference between modifiers and operators is that modifiers can often be placed both to the left or to the right of the nucleus (see (22)). Moreover, modifiers differ from operators concerning constraints on adjunction: they are rarely obligatory and there can be multiple modifiers targeting the same layer, as in (23). If multiple modification is not possible, this is due to semantic or pragmatic ill-formedness (e.g., conflicting aspectual information).

(23)	a.	Ruth carefully hid the cash slowly.	(several core modifiers)
	b.	Evidently, Ruth possibly hid the cash.	(several clause modifiers)

In order to model the word order constraints for periphery modifiers, while going through the sentence in a direction away from the nuclear predicate, we have to keep track of the modifier layer that we have already reached. As an example, consider the possible placements of the adverbs *evidently* (evidential, clausal periphery), *slowly* (pace adverb, core periphery) and *completely* (aspectual adverb, nuclear periphery) within the sentence in (24), an example taken from Van Valin (2005, p. 20).

(24) Leslie has been immersing herself in the new language.

Figure 20.37. *Syntactic tree for (24)*

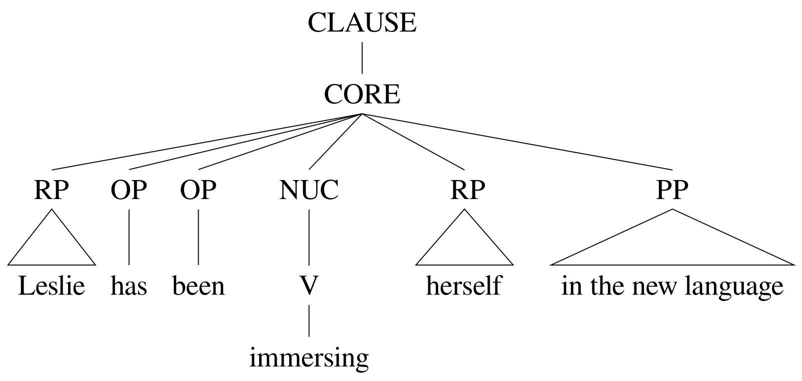
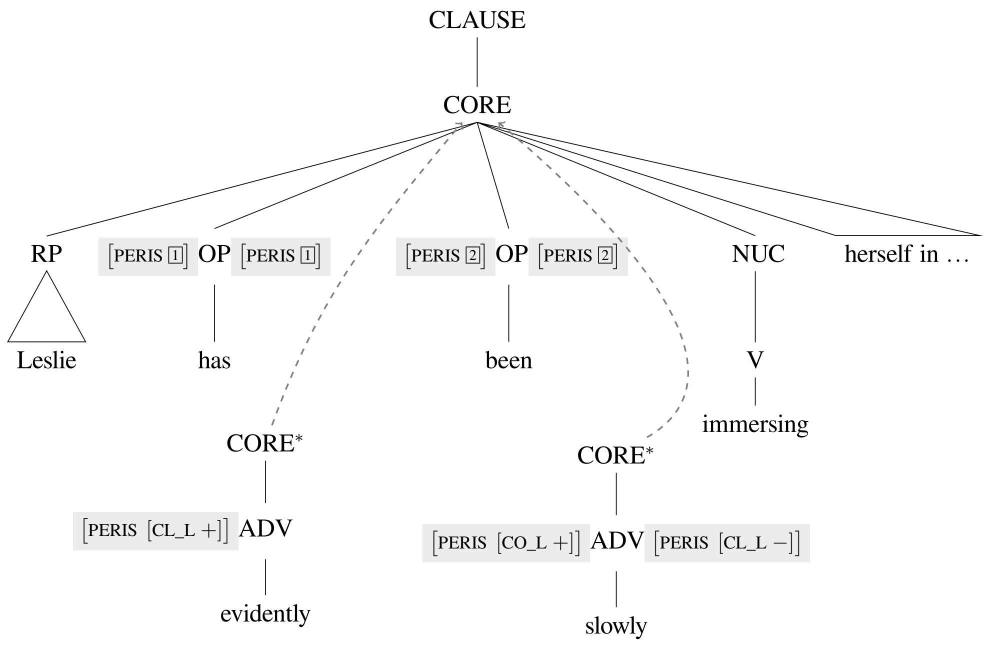


Figure 20.38. *Derivation for (25a)*



The corresponding syntactic tree (without periphery elements) is given in Figure 20.37. Note that here we choose a version where the aspectual operator *been*, which scopes at NUC, attaches to CORE. This is what a parser would choose in order to avoid crossing branches while being able to adjoin an adverbial at CORE that is placed between *been* and *immersing*. All three adverbs can be placed either to the left or to the right of the nucleus, and combinations of this are also possible (cf. Van Valin, 2005, p. 20). The position of the adverbs has to be such that if more than one is to the left of the verbal nucleus, within this group, clausal adverbs have to precede core adverbs, which in turn have to precede nuclear adverbs, while for the group of adverbs placed to the right of the verb, the opposite order has to be respected (first nuclear, then core and then clausal adverbs).

In order to control the order of modifiers with respect to the layered structure, we assume an edge feature PERIS (periphery structure) similar to the feature OPS (operator structure) introduced in Section 20.5. The way this feature controls modifier order is exactly as in the operator case. The operator feature OPS has of course to be percolated along the edge features of the periphery trees and, similarly, operator trees have to percolate the PERIS feature. Figure 20.38 illustrates how the example in (25a) is derived using the PERIS edge feature for keeping track of the periphery structure level reached.

(25) a. Leslie has evidently been slowly immersing herself in the new language.
b. *Leslie has slowly been evidently immersing herself in the new language.

Sentence (25b), by contrast, would be ruled out since the requirement to have PERIS [CL_L −] when adjoining *slowly* would not be satisfied.

In addition to the edge features, which constrain the order of the periphery elements, the ADV node itself can have a feature that indicates at which layer it takes scope. For instance [SCOPE *cl*] for *evidently* and [SCOPE *co*] for *slowly*.

20.8 Linking syntax and semantics

The main focus of the present chapter is on formalizing the syntactic side of RRG. This section shows briefly how the described tree rewriting approach can be extended to combining syntactic and semantic composition along the lines of Kallmeyer and Osswald (2013) and Kallmeyer et al. (2016). In this approach, the linking between syntactic and semantic components is encoded by an interface feature IND(EX), which can occur at nodes in the syntactic tree. The value of this feature is an identifier, or label, which refers to a specific component of the semantic representation.

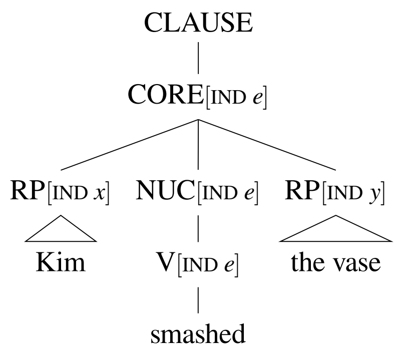
Consider the simple example in (26) and its syntactic representation in the upper part of Figure 20.39.

(26) Kim smashed the vase.

The figure illustrates how the syntactic constituents, represented by nodes in the tree, are linked to components of the associated semantic representation. In fact, the figure shows two alternative semantic representations: a logical structure of the type common in RRG (cf., e.g., Van Valin, 2005, p. 151), and a *semantic frame*, represented as an attribute value matrix, that captures basically the same information about the event and its participants using types, attributes and values. Possible ways of translating RRG's logical structures into semantic frames are discussed in Osswald and Van Valin (2014) and Osswald (2021). An advantage of using frames is that constraints on semantic representations can be formalized in terms of types and attribute-value constraints, and that semantic composition comes down to frame unification.

In accordance with the general approach introduced in Section 20.3, the syntactic tree in Figure 20.39 is derived by, first, combining elementary tree templates with lexical anchors and, second, combining the resulting lexicalized elementary trees via substitution, sister adjunction or wrapping substitution. For the given example, the grammar would contain a transitive template (an elementary tree) that gets anchored by the lexical item *smashed* (cf. Figure 20.40). The anchoring step, i.e., the insertion of the *smashed* tree into the V$\diamond$ node, induces a unification of the feature structures on the two V nodes. The linking between semantic participants and syntactic argument slots can then be computed via a constraint-based formulation of the linking algorithm along the lines of Kallmeyer et al. (2016). The result of the lexical anchoring and the argument linking is the elementary tree plus the semantic frame at the top of Figure 20.41. The trees for *Kim* and *the vase* and their respective frames can now be added by substitution, which means that the frames labeled x' and x unify and likewise the frames labeled y' and y. The result is the interlinked pair consisting of a syntactic tree and a semantic frame shown in Figure 20.39.

Figure 20.39. *Linking syntax and semantics: derived tree*



$$e \;=\; [\mathbf{do}'(x,\varnothing)]\ \text{CAUSE}\ [\text{BECOME}\ \mathbf{smashed}'(y)],\ \ x = \text{Kim},\ \ y = \text{vase}$$

$$e\begin{bmatrix} \textit{causation} & \\ \text{CAUSE} & \begin{bmatrix}\textit{activity} & \\ \text{EFFECTOR} & x\begin{bmatrix}\textit{person} & \\ \text{NAME} & \textit{‘Kim’}\end{bmatrix}\end{bmatrix} \\ \text{EFFECT} & \begin{bmatrix}\textit{change\_of\_state} & \\ \text{PATIENT} & y\begin{bmatrix}\textit{vase}\end{bmatrix} \\ \text{RESULT\_STATE} & \begin{bmatrix}\textit{smashed\_state} \\ \text{PATIENT}\ \ y\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

20.9 Conclusion

The formalization of RRG syntax presented in this chapter puts emphasis on drawing a clear line between declarative and procedural elements and on the proper modeling of syntactic compositionality. Leaning on concepts from the formalism of Tree Adjoining Grammars, we formalized the syntactic dimension of RRG as a tree rewriting grammar consisting of elementary tree templates that are anchored by lexical items and that are combined by three modes of composition: (simple) substitution, (sister) adjunction, and wrapping (substitution). Moreover, elementary tree templates are specified by classes of tree constraints (in the so-called metagrammar), which allows us to make explicit in which way the different templates are structurally related to each other.

Employing wrapping substitution turns out vital for the appropriate modeling of syntactic composition in RRG. Besides allowing the compositional derivation of extraction from complement constructions from elementary argument construction templates (cf. Section 20.3.3), wrapping substitution is also well suited for deriving coordination and cosubordination chains, including those that can arise in embedded control and matrix-coding constructions (cf. Section 20.6.1).

The proposed formalization does not introduce a separate tree structure for representing the operator projection but treats operators as part of the constituent tree. In order to avoid crossing branches, operators need not be directly attached to the constituent nodes that define their scope taking behavior. Rather, this information is encoded in node features associated with the operator tree. An elaborate system of features is also responsible for enforcing the correct surface order of the operators, which reflects their scopal domain (cf. Section 20.5.2).

While the formalization of RRG's semantic structures and its linking system is beyond the scope of the present chapter, we sketched in Section 20.8 how semantic representations can be compositionally integrated with the formal syntactic framework introduced here. We proposed the use of semantic frames for this purpose, but the described interface between syntax and semantics is open to other formal semantic approaches, including a formalized version of RRG's logical

Figure 20.40. *Anchoring the default transitive template with 'smashed'*

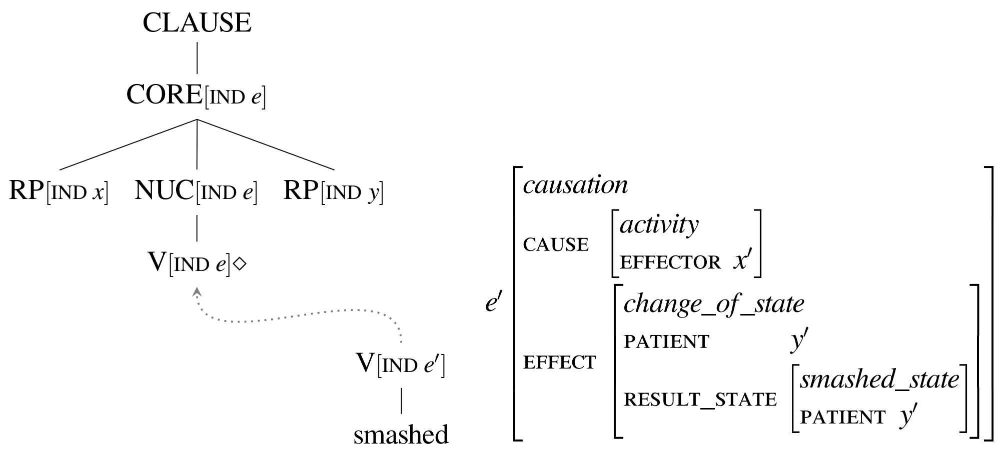
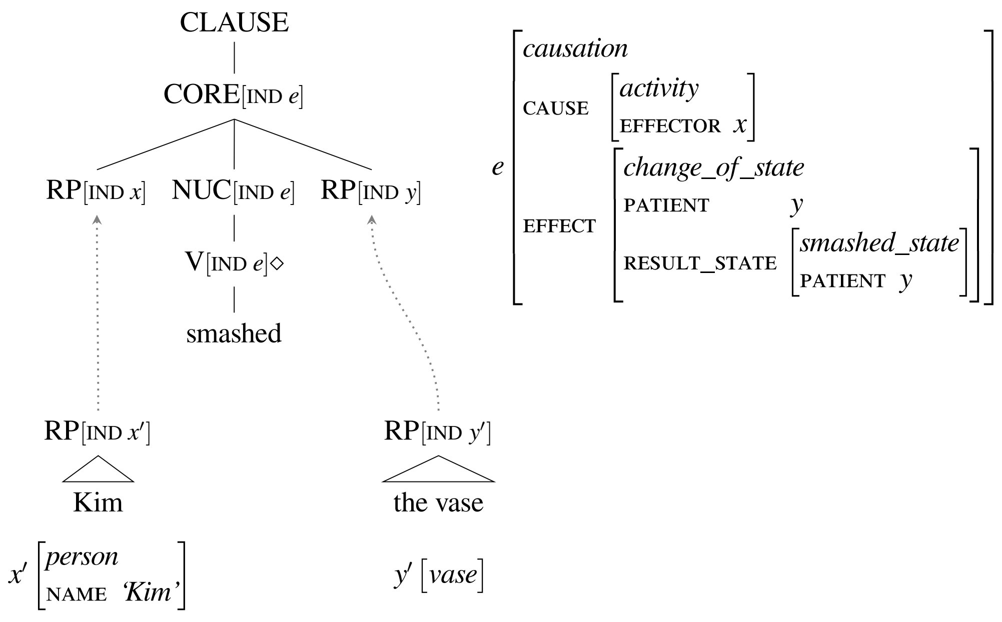


Figure 20.41. *Syntactic composition for (26) after argument linking*



structures.

A complete formalization of RRG as a theory of grammar along the lines of the present chapter requires of course additional steps. In particular, RRG's linking algorithm should be fully spelled out as a system of constraints that make reference to the chosen formal syntactic and semantic representations. Another important and nontrivial task is the formal representation of information structure and its integration with the syntactic and semantic representations.

Acknowledgements

The research presented in this chapter was supported by the Collaborative Research Centre 991 "The Structure of Representations in Language, Cognition, and Science" funded by the German Research Foundation (DFG) and the project TreeGraSP funded by the European Research Council (ERC).

Notes

[1]Round brackets indicate optionality, curly brackets mean that the category in question may not be available in every language. DP stands for ‘detached position’ and ECS for ‘extra-core slot’ (covering pre- and post-core slots).

[2]The given analysis is slightly simplified in that the operator *did* contributes also to other layers than CLAUSE; see Section 20.5 for a more detailed exposition, which also covers operators that are realized as bound morphemes.

[3]The possible occurrence of “crossing branches” due to mismatches between the surface position of an operator or a periphery element and its scope will be discussed in Sections 20.5 and 20.7.

[4]Sister adjunction has been introduced as a composition operation on so-called *d-trees* by Rambow et al. (1995). Notice that sister adjunction differs from adjunction in TAG. In fact, the latter operation is more akin to the wrapping operation described in Section 20.3.3.

[5]We will see later how edge features can be used to constrain the positions of operators or periphery elements among the daughters of their target nodes.

[6]This analysis is inspired by TAG where the slot for the wh-phrase and its predicate originate from the same elementary tree and the elementary tree of the embedding verb adjoins in between; see Kroch (1987).

[7]Wrapping substitution is related to the concept of *flexible composition* proposed in Joshi et al. (2008), which allows one to interpret TAG adjunction as a wrapping operation.

[8]The idea of d-edges or dominance links has been used in various extensions of TAG, for instance in V-TAG (Rambow, 1994), Tree Description Grammars (Kallmeyer, 1999), and D-Tree-Substitution Grammar (Rambow et al., 2001).

[9]Bladier et al. (2020) propose a slight extension of wrapping substitution where in cases of an empty tree γ, the upper node targeted by the operation need not be a root node.

[10]This means that for a given grammar, one can implement a parsing algorithm such that for a sentence of length n, the program computes in at most cn^m steps whether the sentence belongs to the language, c and m being fixed constants.

[11]This transformation of splitting a node with a discontinuous span into several nodes for the different components was also proposed by Boyd (2007), though not with a unique identifier for the split node.

[12]A similar constraint on elementary trees is often proposed in TAG-based approaches to linguistic analysis; cf., e.g., Frank (2002).

[13]Since in passive constructions, the auxiliary *be* is structurally required for nucleus formation, it has a representation in the constituent projection as an AUX node; cf. Van Valin (2005, p. 13, footnote 3).

[14]A computational implementation is provided by the metagrammar compiler XMG (eXtensible MetaGrammar, Crabbé et al., 2013), which allows specifications of RRG trees including d-edges (`xmg.phil.hhu.de`). The resulting grammars can then be processed using the TuLiPA parser (`github.com/spetitjean/TuLiPA-frames`).

[15]Such a representation is chosen in the RRGbank (`rrgbank.phil.hhu.de`) and RRGparbank (`rrgparbank.phil.hhu.de`), both treebanks of syntactic RRG structures, see Bladier et al. (2018).

[16]We slightly simplify here because the MOD feature would actually be embedded under features OP and CO, for instance [OP[CO[MOD *deont*]]].

References

Bladier, Tatiana, Andreas van Cranenburgh, Kilian Evang, Laura Kallmeyer, Robin Möllemann, and Rainer Osswald. 2018. Rrgbank: a role and reference grammar corpus of syntactic structures extracted from the penn treebank. In *Proceedings of the 17th International Workshop on Treebanks and Linguistic Theory*, pp. 5–16. Linköping Electronic Conference Proceedings.

Bladier, Tatiana, Laura Kallmeyer, Rainer Osswald, and Jakub Waszczuk. 2020. Automatic extraction of tree-wrapping grammars for multiple languages. In *Proceedings of the 19th Workshop on Treebanks and Linguistic Theories*, pp. 55–61. Association for Computational Linguistics.

Bohnemeyer, Jürgen and Robert D. Van Valin, Jr. 2017. The macro-event property and the layered structure of the clause. *Studies in Language*, 41(1):142–197.

Boyd, Adriane. 2007. Discontinuity revisited: An improved conversion to context-free representations. In *The Linguistic Annotation Workshop at ACL 2007*, pp. 41–44, Prague, Czech Republic.

Crabbé, Benoit and Denys Duchier. 2005. Metagrammar redux. In Christiansen, Henning, Peter Rossen Skadhauge, and Jørgen Villadsen, editors, *Constraint Solving and Language Processing*, Lecture Notes in Computer Science 3438, pp. 32–47. Springer, Berlin.

Crabbé, Benoit, Denys Duchier, Claire Gardent, Joseph Le Roux, and Yannick Parmentier. 2013. XMG: eXtensible MetaGrammar. *Computational Linguistics*, 39(3):1–66.

Croft, William, Jóhanna Barðdal, Willem Hollmann, Violeta Sotirova, and Chiaki Taoka. 2010. Revising Talmy's typological classification of complex events. In Boas, Hans C., editor, *Contrastive Studies in Construction Grammar*, pp. 201–236. John Benjamins, Amsterdam.

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

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

Johnson, Mark. 1987. A new approach to clause structure in Role and Reference Grammar. In *Davis Working Papers in Linguistics 2*, pp. 55–59. University of California, Davis, CA.

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

Joshi, Aravind K. and Yves Schabes. 1997. Tree-adjoining grammars. In Rozenberg, Grzegorz and Arto Salomaa, editors, *Handbook of Formal Languages. Vol. 3: Beyond Words*, pp. 69–123. Springer, Berlin.

Joshi, Aravind K., Laura Kallmeyer, and Maribel Romero. 2008. Flexible composition in LTAG: Quantifier scope and inverse linking. In Bunt, Harry and Reinhard Muskens, editors, *Computing Meaning*, volume 3, pp. 233–256. Springer, Berlin.

Kallmeyer, Laura. 1999. *Tree Description Grammars and Underspecified Representations.* PhD thesis, Universität Tübingen. Technical Report IRCS-99-08 at the Institute for Research in Cognitive Science, Philadelphia.

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

Kallmeyer, Laura. 2016. On the mild context-sensitivity of k-tree wrapping grammar. In Foret, Annie, Glyn Morrill, Reinhard Muskens, Rainer Osswald, and Sylvain Pogodalla, editors, *Formal Grammar: 20th and 21st International Conferences, FG 2015, Barcelona, Spain, August 2015, Revised Selected Papers. FG 2016, Bozen, Italy, August 2016, Proceedings*, number 9804 in Lecture Notes in Computer Science, pp. 77–93, Berlin. Springer.

Kallmeyer, Laura and Rainer Osswald. 2013. Syntax-driven semantic frame composition in Lexicalized Tree Adjoining Grammars. *Journal of Language Modelling*, 1(2):267–330.

Kallmeyer, Laura and Rainer Osswald. 2017. Combining predicate-argument structure and operator projection: Clause structure in Role and Reference Grammar. In Kuhlmann, Marco and Tatjana Scheffler, editors, *Proceedings of the 13th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+13)*, pp. 61–70. Association for Computational Linguistics.

Kallmeyer, Laura, Rainer Osswald, and Robert D. Van Valin, Jr. 2013. Tree wrapping for Role and Reference Grammar. In Morrill, Glyn and Mark-Jan Nederhof, editors, *Formal Grammar (FG 2012/2013)*, Lecture Notes in Computer Science 8036, pp. 175–190. Springer.

Kallmeyer, Laura, Timm Lichte, Rainer Osswald, and Simon Petitjean. 2016. Argument linking in LTAG: A constraint-based implementation with XMG. In Chiang, David and Alexander Koller, editors, *Proceedings of the 12th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+12)*, pp. 48–57. Association for Computational Linguistics.

Kanazawa, Makoto. 2016. Multidimensional trees and a Chomsky-Schützenberger-Weir representation theorem for simple context-free tree grammars. *Journal of Logic and Computation*, 26(5):1469–1516.

Kroch, Anthony. 1987. Unbounded dependencies and subjacency in a Tree Adjoining Grammar. In Manaster-Ramer, Alexis, editor, *The Mathematics of Language*, pp. 134–172. John Benjamins, Philadelphia.

Nolan, Brian. 2004. First steps toward a computational RRG. In *Proceedings of the International Role and Reference Grammar Conference 2004*, pp. 196–223, Dublin. Institute of Technology Blanchardstown.

Osswald, Rainer. 2021. Activities, accomplishments and causation. In Van Valin, Robert D., Jr., editor, *Challenges in the analysis of the syntax-semantics-pragmatics interface: A Role and Reference Grammar perspective*, pp. 3–30. Cambridge Scholars Publishing, Newcastle upon Tyne.

Osswald, Rainer and Laura Kallmeyer. 2018. Towards a formalization of Role and Reference Grammar. In Kailuweit, Rolf, Eva Staudinger, and Lisann Künkel, editors, *Applying and Expanding Role and Reference Grammar*, (NIHIN Studies), pp. 355–378. Albert-Ludwigs-Universität, Universitätsbibliothek, Freiburg.

Osswald, Rainer and Robert D. Van Valin, Jr. 2014. FrameNet, frame structure, and the syntax-semantics interface. In Gamerschlag, Thomas, Doris Gerland, Rainer Osswald, and Wiebke Petersen, editors, *Frames and Concept Types*, pp. 125–156. Springer, Dordrecht.

Osswald, Rainer and Robert D. Van Valin, Jr. 2021. The description of transitive directed motion in Lakhota (Siouan). In Sarda, Laure and Benjamin Fagard, editors, *Neglected Aspects of Motion Events Description: Deixis, Asymmetries, Constructions*. John Benjamins, Amsterdam.

Pollard, Carl J. and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago.

Rambow, Owen. 1994. *Formal and Computational Aspects of Natural Language Syntax*. PhD thesis, University of Pennsylvania.

Rambow, Owen, K. Vijay-Shanker, and David Weir. 1995. D-tree grammars. In *Proceedings of the 33rd Annual Meeting*, pp. 151–158. Association for Computational Linguistics.

Rambow, Owen, K. Vijay-Shanker, and David Weir. 2001. D-tree substitution grammars. *Computational Linguistics*, 27(1):87–121.

Sag, Ivan. 2012. Sign-Based Construction Grammar: An informal synopsis. In Boas, Hans and Ivan Sag, editors, *Sign-Based Construction Grammar*, pp. 61–188. CSLI Publications, Stanford.

Sag, Ivan A. and Thomas Wasow. 1999. *Syntactic Theory: A Formal Introduction*. CSLI Publications, Stanford, CA.

Ullrich, Jan, editor. 2011. *New Lakota Dictionary*. Lakota Language Consortium, Bloomington, 2nd edition.

Ullrich, Jan. 2018. *Modification, Secondary Predication and Multi-Verb Constructions in Lakota*. PhD thesis, Heinrich-Heine-Universität Düsseldorf.

Van Valin, Robert D., Jr. 2005. *Exploring the Syntax-Semantics Interface*. Cambridge University Press, Cambridge.

Van Valin, Robert D., Jr. and Randy J. LaPolla. 1997. *Syntax*. Cambridge University Press, Cambridge.