
Research on Language and Computation **1:** 3–58, 2003.

Factoring Predicate Argument and Scope Semantics: Underspecified Semantics with LTAG

LAURA KALLMEYER[1] and ARAVIND K. JOSHI[2]
[1]*UFRL, University Paris 7, 2 place Jussieu, 75251 Paris Cedex 05, France (E-mail: laura.kallmeyer@linguist.jussieu.fr);* [2]*IRCS, University of Pennsylvania, 3401 Walnut Street, Philadelphia, PA 19104-6228, USA (E-mail: joshi@linc.cis.upenn.edu)*

Abstract. In this paper we propose a compositional semantics for lexicalized tree-adjoining grammar (LTAG). Tree-local multicomponent derivations allow separation of the semantic contribution of a lexical item into one component contributing to the predicate argument structure and a second component contributing to scope semantics. Based on this idea a syntax-semantics interface is presented where the compositional semantics depends only on the derivation structure. It is shown that the derivation structure (and indirectly the locality of derivations) allows an appropriate amount of underspecification. This is illustrated by investigating underspecified representations for quantifier scope ambiguities and related phenomena such as adjunct scope and island constraints.

Key words: computational semantics, lexicalized tree-adjoining grammar, quantifier scope, underspecification

1. Introduction

A lexicalized tree-adjoining grammar (LTAG) consists of a finite set of trees (elementary trees) associated with lexical items and composition operations of substitution (replacing a leaf with a new tree) and adjoining (replacing an internal node with a new tree). The elementary trees represent extended projections of lexical items and encapsulate syntactic/semantic arguments of the lexical anchor. They are minimal in the sense that all, and only the syntactic/semantic arguments, are encapsulated and further, all recursion is factored away. This factoring of recursion is what leads to the trees being extended projections. The elementary trees of LTAG are therefore said to possess an extended domain of locality.

The extended domain of locality of the trees in an LTAG gives a way to formulate a syntax-semantics interface on a more abstract level, namely as a relation between elementary trees and semantic representations, without violating the principle of compositionality. Consequently, semantic representations do not need to reproduce the internal structure of elementary trees and therefore one can use 'flat' semantic representations in the style of Minimal Recursion Semantics (MRS, Copestake et al., 1999). One advantage of the more flexible relation between syntax and semantics is that, in contrast to phrase structure based approaches, elements

such as quantifiers, whose semantic interpretation does not directly correspond to their (syntactic) surface position, do not pose a problem.

LTAG derivations are represented by derivation trees that record the history of how the elementary trees are put together. A derived tree is the result of carrying out the substitutions and adjoinings. Because of the localization of the arguments of a lexical item within elementary trees the proper way to define compositional semantics for LTAG is with respect to the derivation tree rather than the derived tree (Candito and Kahane, 1998a, b). We assume that each elementary tree is related to a semantic representation. The derivation tree indicates how to combine the semantic representations. As already mentioned, this contrasts with traditional approaches where each node in the syntactic structure is associated with a semantic representation. Although this insight has been present from the beginning of the work on LTAG (Shieber and Schabes, 1990) a systematic formulation was begun only recently by Joshi and Vijay-Shanker (Joshi and Vijay-Shanker, 1999). One of their goals was to investigate the role of underspecification in compositional semantics; they suggested that LTAG derivation trees provide just the right amount of underspecification necessary for scope semantics. Their discussion was preliminary, however.

In our approach we use a LTAG variant called multicomponent TAG (MC-TAG). A MC-TAG consists of elementary sets of trees. The locality of composition in LTAG is extended to MC-TAG as follows: Basically, when two multicomponent tree sets are combined, the components of one set combine with only one of the components of the other set. We use tree-local MC-TAG with at most two components in each set. The key idea is that one of the components of a tree set contributes to the predicate argument aspects of semantics and the other component contributes to the scope semantics.

In order to obtain underspecified representations for scope ambiguities, we adopt ideas from Hole Semantics (Bos, 1995) and enrich the semantic representations with propositional metavariables called holes. A partial order on holes and propositional labels describes the scope structure of a semantic representation. A disambiguation function maps holes to propositional labels in such a way that the scope constraints are respected. We will see that the LTAG derivation trees (restricted by the tree-locality of the grammar) provide the right amount of underspecification to generate suitable representations for scope semantics. This will be illustrated investigating phenomena such as quantifier scope and adjunct scope.

Among recent approaches to underspecified semantics, in particular Muskens and Krahmer (1998) and Kallmeyer (1999b) are closely related to the work presented in this paper. Both proposals also separate scope information from predicate argument semantics. In Muskens and Krahmer (1998), however, there is no locality constraint and therefore its use of underspecification is too general. Tree descriptions and locality of TAGs are used in Kallmeyer (1999b). But in order to control the amount of underspecification that comes with the use of descriptions, rather complex formal definitions are necessary. This problem is avoided in our

approach where syntactic structures are represented by trees and underspecification is used only in a very limited way for scope relations between propositional formulas.

The structure of the paper is as follows: In the next section, we will introduce LTAG and, proposing an alternate perspective on adjoining, motivate the use of tree-local MCTAG with elementary tree sets containing at most two trees. Section 3 explains the architecture of the syntax semantics interface we propose referring in particular to quantifier and adjunct scope. The formal definitions underlying this approach are then described in detail in Section 4. The last section we investigate restrictions on quantifier scope such as island constraints.

2. Lexicalized Tree-Adjoining Grammar (LTAG)

2.1. LEXICALIZATION AND EXTENDED DOMAINS OF LOCALITY

Tree-adjoining grammar (TAG) is a formal tree rewriting system originally introduced in Joshi et al. (1975). TAG and Lexicalized Tree-Adjoining Grammar (LTAG) have been studied extensively both with respect to their formal properties (see for example Vijay-Shanker and Joshi, 1985; Vijayashanker, 1987; Joshi, 1987) and to their linguistic relevance (e.g., Joshi, 1985). TAG and LTAG are formally equivalent. However, from the linguistic perspective, LTAG is the system we will be concerned with in this paper. We will often use the terms TAG and LTAG interchangeably.

The motivations for the study of LTAG are both linguistic and formal. The elementary objects manipulated by LTAG are structured objects (trees or directed acyclic graphs) and not strings. Using structured objects as the elementary objects of the formal system, it is possible to construct formalisms whose properties relate directly to the study of strong generative capacity (i.e., structural descriptions), which is more relevant to the linguistic descriptions than the weak generative capacity (sets of strings).

Each grammar formalism specifies a domain of locality, i.e., a domain over which various dependencies (syntactic and semantic) can be specified. It turns out that the various properties of a formalism (syntactic, semantic, computational, and even psycholinguistic) follow, to a large extent, from the initial specification of the domain of locality.

In a context-free grammar (CFG) the domain of locality is the one level tree corresponding to a rule in a CFG (Figure 1). It is easily seen that in general, the arguments of a predicate (for example, the two arguments of *likes*) are not in the same local domain. The two arguments are distributed over the two rules (two domains of locality) $S \rightarrow NP\ VP$ and $VP \rightarrow V\ NP$. They can be brought together by introducing a rule $S \rightarrow NP\ V\ NP$. However, then the structure provided by the VP node is lost.

We should also note here that not every rule (domain) in the CFG in Figure 1 is lexicalized. The three rules on the right are lexicalized, i.e., they have a lexical

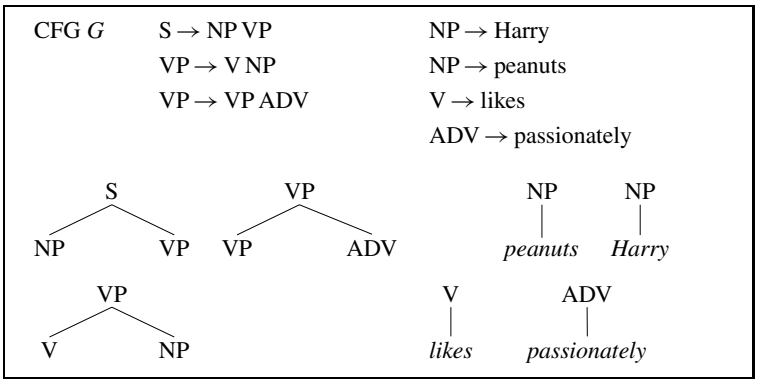


Figure 1. Domain of locality of a context-free grammar.

anchor. The rules on the left are not lexicalized. The second and the third rules on the left are almost lexicalized, in the sense that they each have a preterminal category (V in the second rule and ADV in the third rule), i.e., by replacing V by *likes* and ADV by *passionately* these two rules will become lexicalized. However, the first rule on the left (S → NP VP) cannot be lexicalized. It can be shown (see Joshi and Schabes, 1997) that CFGs cannot be lexicalized, i.e. that for a given CFG G, it is in general not possible to construct another CFG G', such that every rule in G' is lexicalized and $T(G)$, the set of (sentential) trees (i.e., the tree language of G) is the same as the tree language $T(G')$ of G'. Of course, if we require only the string languages of G and G' to be the same (i.e., G and G' are weakly equivalent) then any CFG can be lexicalized. This follows from the fact that any CFG can be put in the Greibach normal form where each rule is of the form $\mathrm{A} \rightarrow a\,\mathrm{B}_1\,\mathrm{B}_2\,\ldots\,\mathrm{B}_n$ where a is a terminal symbol. The lexicalization we are interested in requires the tree languages (i.e., the set of structural descriptions) to be the same, i.e., we are interested in the 'strong' lexicalization. To summarize, in general, a CFG cannot be strongly lexicalized by a CFG because the domain of locality is a one level tree corresponding to a rule in the grammar.

Note that there are two issues we are concerned with here:

1. lexicalization of each elementary domain of locality, and
2. encapsulation of the the arguments of the lexical anchor in the elementary domain.

The second issue, the so-called predicate argument co-occurrence, is independent of the first issue. From the mathematical point of view the first issue, i.e., the lexicalization of the elementary domains of locality is the crucial one. We can obtain strong lexicalization without satisfying the requirement specified in the second issue (encapsulation of the arguments of the lexical anchor). Of course, from the linguistic point of view the second issue is very crucial. What this means is that among all possible strong lexicalizations we should choose only those that

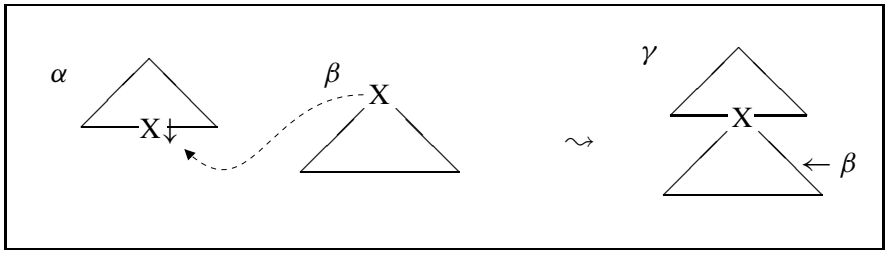


Figure 2. Substitution.

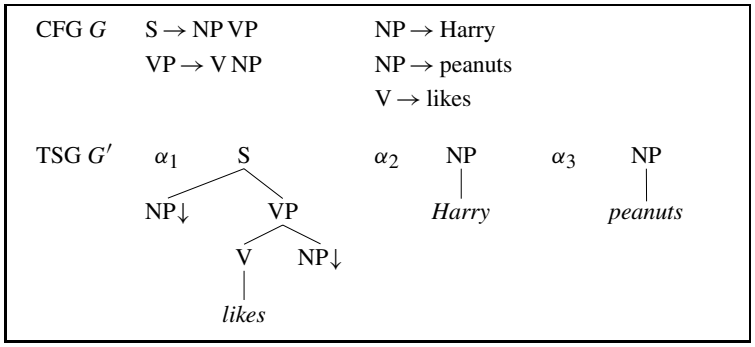


Figure 3. Tree substitution grammar.

meet the requirements of the second issue. For our discussions in this paper we will assume that we always make such a choice.

Now we can ask the following question: Can we strongly lexicalize a CFG by a grammar with a larger domain of locality? Figures 2 and 3 show a CFG and a corresponding tree substitution grammar where the elementary objects (building blocks) are the three trees in Figure 3 and the combining operation is the tree substitution operation shown in Figure 2. (Leaves that need to be replaced by a tree in a substitution operation, e.g., the two NP nodes in α_1 in G', are marked with a vertical arrow). Note that each tree in the tree substitution grammar (TSG), G' is lexicalized, i.e., it has a lexical anchor. It is obvious that G' indeed strongly lexicalizes G. However, TSGs fail to strongly lexicalize CFGs in general. We show this by an example. Consider the CFG, G, in Figure 4 and a proposed TSG, G'. It is easily seen that although G and G' are weakly equivalent they are not strongly equivalent. In G', suppose we start with the tree α_1 then by repeated substitutions of trees in G' (a node marked with a vertical arrow denotes a substitution site) we can grow the right side of α_1 as much as we want but we cannot grow the left side. Similarly for α_2 we can grow the left side as much as we want but not the right side. However, trees in G can grow on both sides. Hence, the TSG, G', cannot strongly lexicalize the CFG, G (Joshi and Schabes, 1997).

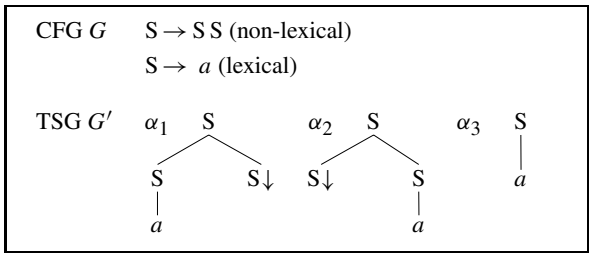


Figure 4. A tree substitution grammar.

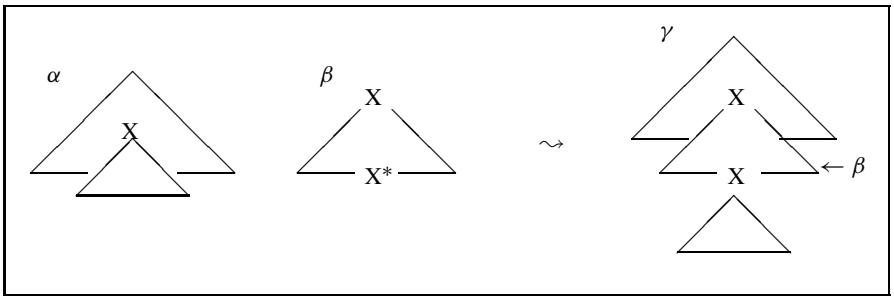


Figure 5. Adjoining.

We now introduce a new operation called 'adjoining' as shown in Figure 5. Adjoining involves splicing (inserting) one tree into another. More specifically, a tree β as shown in Figure 5 is inserted (adjoined) into a tree α at a node u with label X resulting in the tree γ. The tree β, called an auxiliary tree, has a special

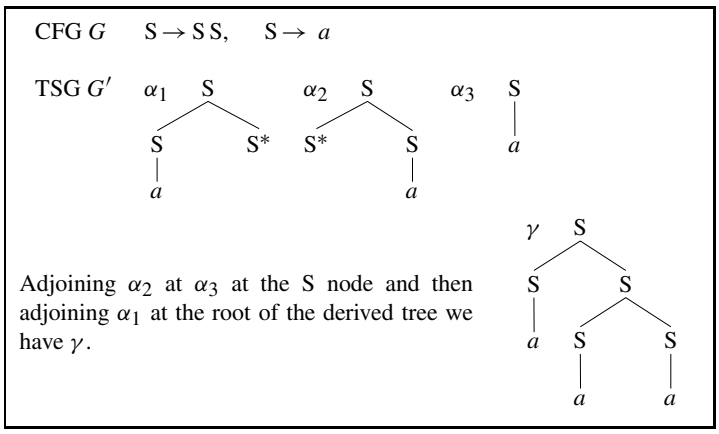


Figure 6. Adjoining arises out of lexicalization.

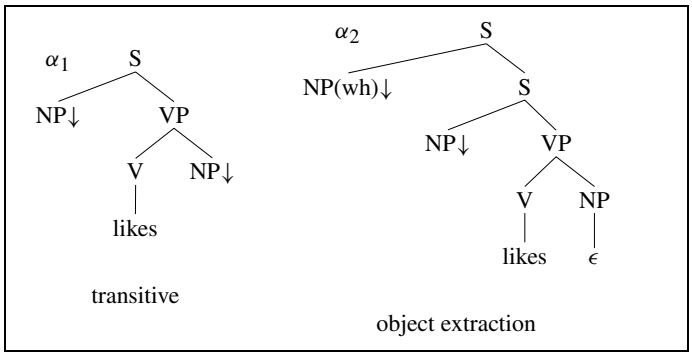


Figure 7. LTAG: Elementary trees for *likes*.

form. The root node is labelled with the same nonterminal as the node u in α and on the frontier of β there is also a node labelled X called a foot node (marked with *). There could be other nodes (terminal or nonterminal) on the frontier of β, the nonterminal nodes will be marked as substitution sites (with a vertical arrow). Thus if there is another occurrence of X (other than the foot node marked with *) on the frontier of β, it will be marked with the vertical arrow, and that will be a substitution site. Given this specification, adjoining of β to α at the node u in α is uniquely defined. Adjoining can also be seen as a pair of substitutions as follows. The subtree at u in α is detached, β is substituted at u and the detached subtree is then substituted at the foot node of β. A tree substitution grammar when augmented with the adjoining operation is called the tree-adjoining grammar (lexicalized tree-adjoining grammar if each elementary tree is lexically anchored). In short, LTAG consists of a finite set of elementary trees, each lexicalized with at least one lexical anchor. The elementary trees are either initial or auxiliary trees. Auxiliary trees have been defined already. Initial trees are those for which all nonterminal nodes on the frontier are substitution nodes. It can be shown that any CFG can be strongly lexicalized by an LTAG (Joshi and Schabes, 1997).

In Figure 6 we show a TSG, G', augmented by the operation of adjoining, which strongly lexicalizes the CFG, G. Note that the LTAG looks the same as the TSG considered in Figure 4 before. However, now trees α_1 and α_2 are auxiliary trees (marked with *) that can participate in adjoining. Since adjoining can insert a tree in the interior of another tree it is possible to grow both sides of the trees α_1 and α_2, which was not possible earlier with substitution alone.

In summary, we have shown that by increasing the domain of locality we have achieved the following: (1) lexicalized each elementary domain, (2) introduced an operation of adjoining, which would not be possible without the increased domain of locality (note that with one level trees as elementary domains adjoining becomes the same as substitution since there are no interior nodes to be operated upon), and (3) achieved strong lexicalization of CFGs.

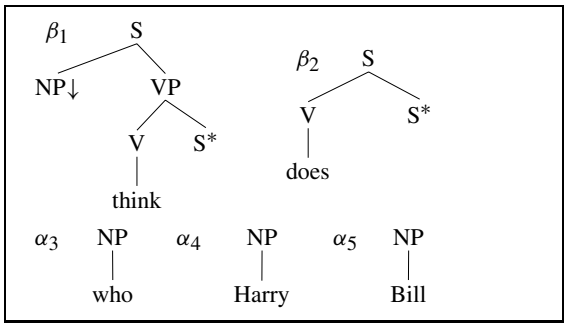


Figure 8. LTAG: Sample elementary trees.

Rather than giving formal definitions for LTAG and derivations in LTAG we will give a simple example to illustrate some key aspects of LTAG. We show some elementary trees of a toy LTAG grammar of English. Figure 7 shows two elementary trees for a verb such as *likes*. The tree α_1 is anchored on *likes* and encapsulates the two arguments of the verb. The tree α_2 corresponds to the object extraction construction. Since we need to encapsulate all the arguments of the verb in each elementary tree for *likes*, for the object extraction construction, we need to make the elementary tree associated with *likes* large enough so that the extracted argument is in the same elementary domain. Thus, in principle, for each 'minimal' construction in which *likes* can appear (for example, subject extraction, topicalization, subject relative, object relative, passive, etc.) there will be an elementary tree associated with that construction. By 'minimal' we mean when all recursion has been factored away. This factoring of recursion away from the domain over which the dependencies have to be specified is a crucial aspect of LTAG as they are used in linguistic descriptions. This factoring allows all dependencies to be localized in the elementary domains. In this sense, there will, therefore, be no long distance dependencies as such. They will all be local and will become long distance on account of the composition operations, especially adjoining.

Figure 8 shows some additional trees. Trees α_3, α_4, and α_5 are initial trees and trees β_1 and β_2 are auxiliary trees with foot nodes marked with *. A derivation using the trees in Figure 8 is shown in Figure 9. The trees for *who* and *Harry* are substituted in the tree for *likes* at the respective NP nodes, the tree for *Bill* is substituted in the tree for *think* at the NP node, the tree for *does* is adjoined to the root node of the tree for *think* (adjoining at the root node is a special case of adjoining), and finally the derived auxiliary tree (after adjoining β_2 to β_1) is adjoined to the indicated interior S node of the tree α_2. This derivation results in the derived tree for *who does Bill think Harry likes* as shown in Figure 10. Note that the dependency between *who* and the complement NP in α_2 (local to that tree) has been stretched in the derived tree in Figure 10. This tree is the conventional tree associated with the sentence.

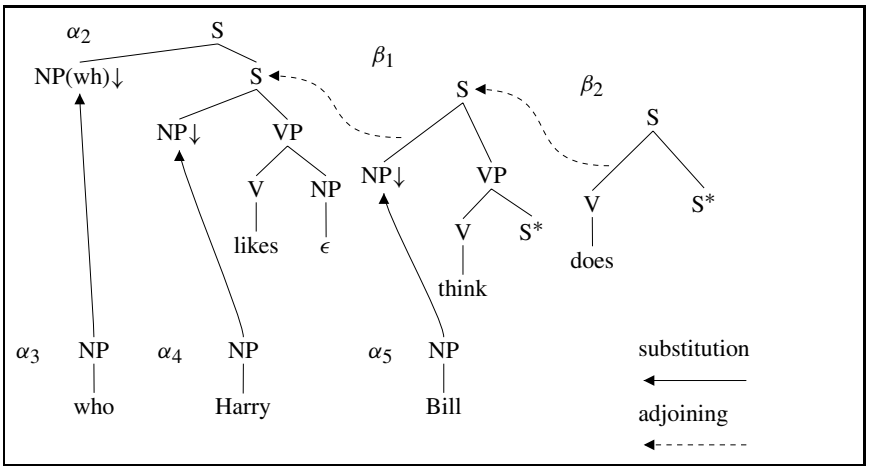


Figure 9. LTAG derivation for *who does Bill think Harry likes*.

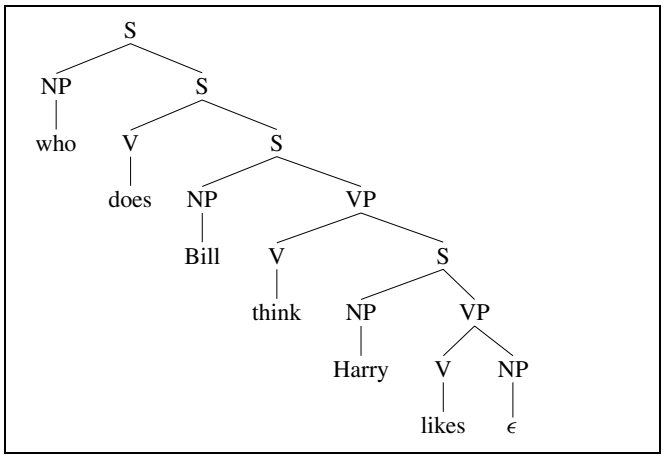


Figure 10. LTAG derived tree for *who does Bill think Harry likes*.

Besides the derived tree, in LTAG, there is also a derivation tree, the tree that records the history of composition of the elementary trees associated with the lexical items in the sentence. This derivation tree is shown in Figure 11. The nodes of the tree are labelled by the tree labels such as α_2 together with the lexical anchor.[1] Each edge is equipped with the position of the node at which the corresponding operation takes place. E.g. β_1 is adjoined at the node at position 01 (the daughter 1, counted from left to right and starting with 0, of the root node, which has position 0) in α_2. The derivation tree is the crucial derivation structure for LTAG. We can obviously build the derived tree from the derivation tree. For semantic computation the derivation tree (and not the derived tree) is the crucial object. Compositional semantics is defined on the derivation tree. The idea is that

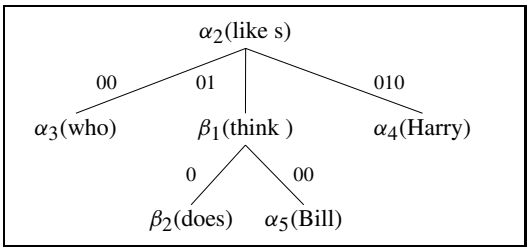


Figure 11. LTAG derivation tree.

for each elementary tree there is a semantic representation associated with it, and these representations are composed using the derivation tree. Since the semantic representation for each elementary tree is directly associated with it there is no need to reproduce unnecessarily the internal hierarchy in the elementary tree. This allows the so-called 'flat' semantic representation as well as helps in dealing with some non-compositional aspects as in the case of rigid and flexible idioms.

2.2. SOME IMPORTANT PROPERTIES OF LTAG

The two properties of LTAG are (1) extended domain of locality (EDL) (for example, as compared to CFG), which allows (2) factoring recursion from the domain of dependencies (FRD), thus making all dependencies local. All other properties of LTAG (mathematical, linguistic, and even psycholinguistic) follow from EDL and FRD. TAGs (LTAGs) belong to the so-called class of mildly context-sensitive grammars (Joshi, 1985; Weir, 1988). CFLs are properly contained in the class of languages of LTAG, which in turn are properly contained in the class of context-sensitive languages. There is a machine characterization for TAG (LTAG), called embedded pushdown automaton (EPDA) (Vijaya-Shanker, 1987), i.e., for every TAG language there is an EPDA which corresponds to this (and only this) language and the language accepted by any EPDA is a TAG language. EPDA's have been used to model some psycholinguistic phenomena (Joshi, 1990). The class of TAG languages enjoy all important properties of CFLs, including polynomial parsing (with complexity $O(n^6)$).

Large scale wide coverage grammars have been built using LTAG, the XTAG system (LTAG grammar and lexicon for English and a parser) being the largest so far (for further details see The XTAG Research Group, 1998). In the XTAG system, each node in each LTAG tree is decorated with two feature structures (top and bottom feature structures), in contrast with CFG based feature structure grammars, because adjoining can augment a tree internally, while in a CFG based grammar a tree can be augmented only at the frontier. It is possible to define adjoining and substitution (as it is done in the XTAG system) in terms of appropriate unifications of the top and bottom feature structures (Vijay-Shanker and Joshi, 1988).

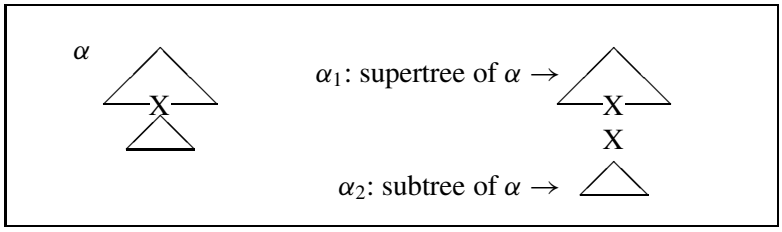


Figure 12. Adjoining as Wrapping 1.

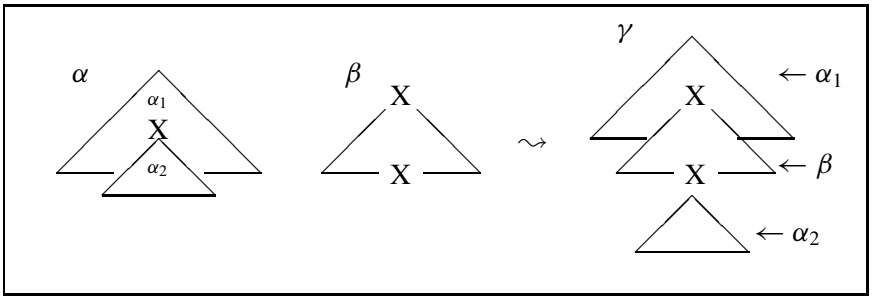


Figure 13. Adjoining as Wrapping 2.

Because of FRD (factoring recursion from the domain of dependencies), there is no recursion in the feature structures. Therefore, in principle, feature structures can be eliminated. However, they are crucial for linguistic descriptions. Constraints on substitution and adjoining are modelled via these feature structures. This way of manipulating feature structures is a direct consequence of the extended domain of locality of LTAG.

2.3. AN ALTERNATE PERSPECTIVE ON ADJOINING

In adjoining we insert an auxiliary tree, say with root and foot nodes labelled with X in a tree at a node u with label X. In Figures 12 and 13 we present an alternate perspective on adjoining. The tree α which receives adjunction at u (labelled X) can be viewed as made up of two trees, the supertree at u and the subtree at u as shown in Figure 12. Now, instead of the auxiliary tree β adjoined to the tree α at u we can view this composition as a wrapping operation – the supertree of α and the subtree of α are wrapped around the auxiliary tree β as shown in Figure 13. The resulting tree γ is the same as before. Wrapping of the supertree at the root node of β is like adjoining at the root (a special case of adjoining) and the wrapping of the subtree at the foot node of β is like substitution. Hence, this wrapping operation can be described in terms of substitution and adjoining.

As an example consider Figures 14 and 15. The auxiliary tree β can be adjoined to the tree α at the indicated node as shown in Figure 14. Alternatively, we can view

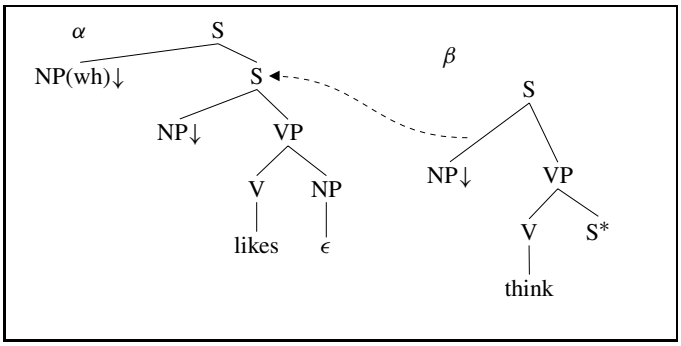


Figure 14. Wrapping as substitution and adjunction 1.

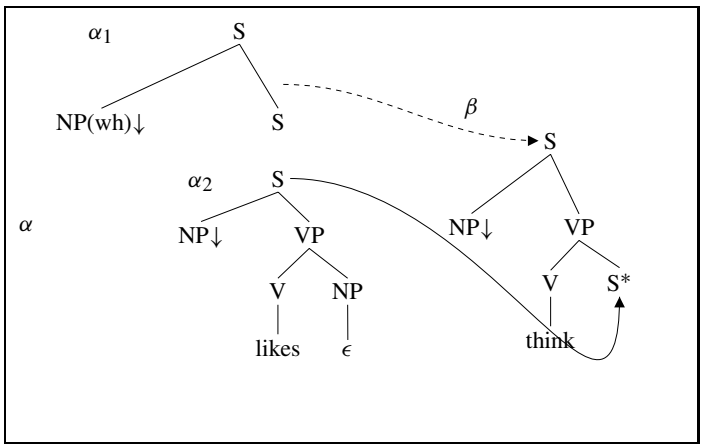


Figure 15. Wrapping as substitution and adjunction 2.

this composition as adjoining the supertree α_1 (the *wh* tree) at the root node of β and substitution of the subtree α_2 (the *likes* tree) at the foot node of β as shown in Figure 15. The two ways of composing α and β are semantically coherent.

The wrapping perspective can be formalized in terms of the so-called multi-component LTAG (MC-LTAG). They are called multi-component because the elementary objects can be sets of trees. In our examples, we have two components (in which α was split). When we deal with multi-components we can violate the locality of the composition very quickly because the different components may be 'attached' (by adjoining or substitution) to different nodes of a tree and these nodes may or may not be part of an elementary tree, depending on whether the tree receiving the multi-component attachments is an elementary or a derived tree. We obtain so-called tree-local MC-LTAG if we adopt the constraint that the tree receiving multi-component attachments must be an elementary tree. It is known

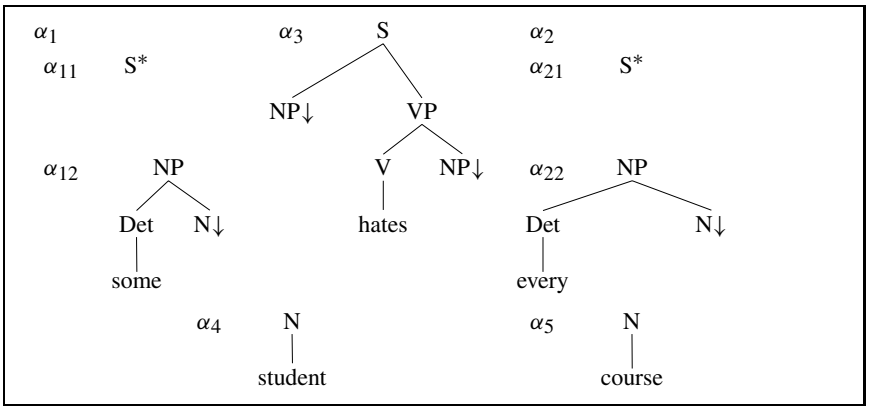


Figure 16. Scope ambiguity: An example.

that tree-local MC-TAGs are weakly equivalent to LTAG, however they can give rise to structural descriptions not obtainable by LTAG, i.e., they are more powerful than LTAG in the sense of strong generative capacity (Weir, 1988). Thus the alternate perspective leads to greater strong generative capacity, without increasing the weak generative capacity.

We will now illustrate how this alternate perspective can be used to characterize the scope ambiguity in *some student hates every course* as shown in Figures 16, 17 and 18. In Figure 16, we show a tree-local MC-LTAG for our example. The trees for *hates*, *student*, and *course* are standard LTAG trees. The trees for *some* and *every* are multi-component trees. For example, the tree α_1 for *some* has two components, α_{11} and α_{12}, one of the components α_{11} is a degenerate tree in this special case. The multi-component tree α_1 is lexically anchored by *some*. Similarly, for the tree α_2 for *every*. The main idea here is that the α_{12} component corresponds to the contribution of *some* to the predicate-argument structure of the tree for *hates* and the α_{11} component contributes to the scope structure (Joshi and Vijay-Shanker, 1999; Kallmeyer and Joshi, 1999). Similarly for the two components of α_2.

Figure 17 shows the derivation. The main point to note here is that the two components of α_1 are attached (by substitution or adjoining) to α_3 at the appropriate nodes simultaneously. This composition is tree local as α_3 is an elementary tree. Similarly for the tree α_2. The two top components α_{11} and α_{21} are attached to the same node (the root node) of α_3. This may give the impression that the composition is non-local because once α_1 is attached to α_3 we have a derived tree to which α_2 is attached. However, the two components, α_{11} and α_{21} are degenerate and it can be shown that in this case the composition of α_2 with α_3 (after α_1 has been composed with α_3) is still effectively tree-local (Kallmeyer and Joshi, 1999).

It is clear in this example that α_2 could have been attached to α_3 first and then α_1 attached to α_3. Figure 18 shows the derivation tree for the derivation in Figure 17. Note that both α_{11} and α_{21}, the scope information carrying components, are

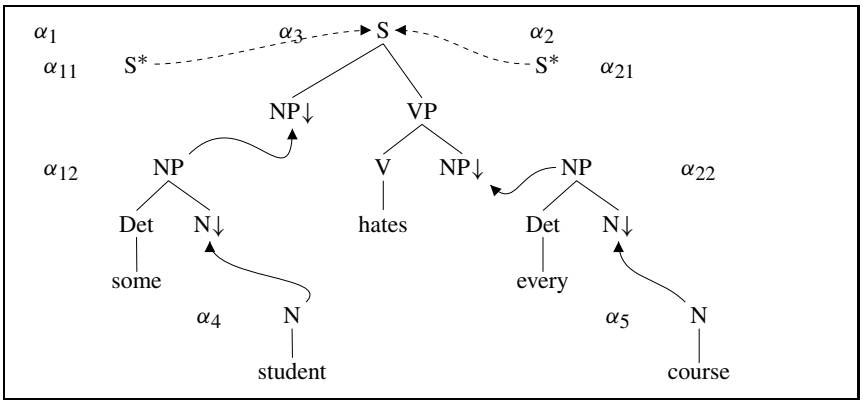


Figure 17. Derivation with scope information.

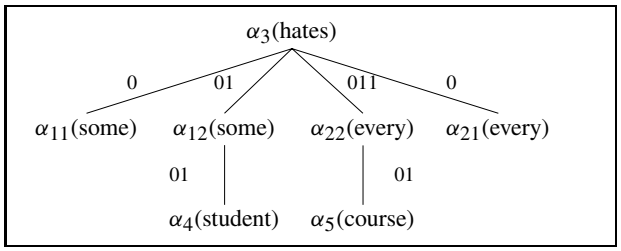


Figure 18. Derivation tree with scope underspecification.

attached to α_3 at the same node, and they could be attached in any order (strictly speaking, α_1 and α_2 could be attached to α_3 in any order). Hence α_{11} and α_{21} behave in exactly the same way with respect to the derivation. The scope ambiguity is thus directly reflected in the derivation tree for *some student hates every course*.[2] This is in contrast to all other approaches (which are essentially CFG based) where the scope ambiguity is represented at another level of representation. It is possible to represent in LTAG, scope ambiguities at the level of the derivation tree itself, because of the alternate perspective on adjoining, which in turn is due to the extended domain of locality discussed in this section. We will explore these ideas in detail later in this paper.

More recently, similar ideas have been explored in the context of other linguistic phenomena such as scrambling and clitic climbing, both with respect to linguistic coverage and certain psycholinguistic implications. A particularly interesting result is that all word order variations up to two levels of embedding (i.e., three clauses in all) can be correctly described by tree-local MC-LTAGs, correctly in the sense of providing the appropriate structural descriptions. Beyond two levels of embedding not all patterns of word order variation will be correctly described.

3. Compositional Semantics with LTAG

In this section, we will develop the architecture of a syntax-semantics interface for LTAG adopting the alternate perspective on adjoining as sketched in the last section. Some of the ideas which form the basis of the work described in this paper have been already investigated in Joshi and Vijay-Shanker (1999).

The overall approach is as follows. Each elementary tree is connected with a semantic representation. The way these semantic representations combine with each other depends on the combination of the elementary trees in a compositional way, i.e., it depends on the derivation structure rather than the derived trees. Since the local domains are extended compared to classical phrase structure grammars (CFGs), the relation is less close than in more traditional Montagovian systems. In this respect, the architecture resembles to earlier proposals as Shieber and Schabes (1990) and also to Kallmeyer (1999b). However, in contrast to these two approaches, in our proposal we will adopt 'flat' semantic representations (as in, for example, Minimal Recursion Semantics MRS, (Copestake et al., 1999).

The fact that the relation between syntax and semantics is a relation between elementary trees and semantic representations and not between single nodes together with their daughters and semantic representations, allows a definition of a monotonic semantics in spite of the nonmonotonicity of TAG derivations with respect to structural properties of trees (see also Joshi and Vijay-Shanker, 1999).

3.1. DERIVATION TREES AND SEMANTIC DEPENDENCIES

In this paper we will follow a key idea in Joshi and Vijay-Shanker (1999) and also Candito and Kahane (1998) that the semantics of a sentence can be built from the derivation structure of LTAG. Underlying this is the observation that TAG derivation trees express predicate argument dependencies.

The elementary trees of an LTAG represent extended projections of lexical items and encapsulate syntactic/semantic arguments of the lexical anchor. They are minimal in the sense that all and only the syntactic/semantic arguments are encapsulated and further, all recursion is factored away. Because of this localization of the arguments of a lexical item within elementary trees, the proper way to define compositional semantics for LTAG is with respect to the derivation tree, rather than the derived tree.

Each edge in a derivation tree represents one derivation step in the LTAG. In the case of a substitution, a new argument is inserted. Therefore, if we assume that a dependency relates a predicate to one of its arguments, the corresponding edge in the derivation tree in these cases must be considered to be directed from the mother to its daughter. However, in the case of an adjunction, we have an opposite direction. The new auxiliary tree represents a predicate that is applied to the tree to which it is adjoined. Therefore, in these cases, the dependency is directed from the daughter to the mother.

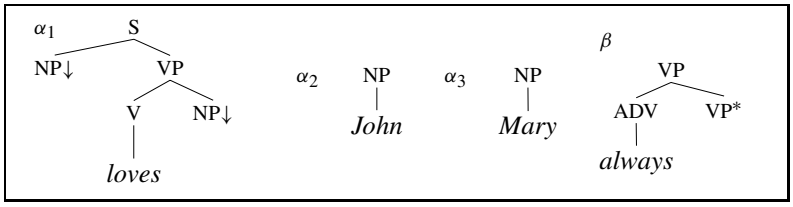


Figure 19. Elementary trees for (1).

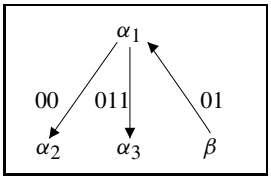


Figure 20. Derivation structure for (1).

Viewing the edges in a derivation tree as directed dependency relations in this way, the derivation tree specifies (independently from the order of the syntactic derivation steps) how to combine the elementary representations corresponding to the elementary trees in a derivation.

(1) John always loves Mary.

As an example, consider the derivation of the syntactic structure and the corresponding compositional semantics for (1). The elementary trees needed to generate (1) are shown in Figure 19.

The derivation starts with α_1. The initial trees α_2 and α_3 are added by substitution to α_1 and β is added by adjunction to α_1. In the corresponding derivation structure, given in Figure 20, the direction of the edges expresses the direction of the semantic dependencies.

The labels of the edges in the derivation tree are the positions of the nodes in the old tree where the new trees are added. 0 is the root position, and for each position p, pn for $n \in \mathbb{N}$ is the position of the $(n+1)th$ daughter (from left to right) of the node at positions p. α_2 is substituted for the node at position 00 in α_1, α_3 for the node at position 011 and β is adjoined to the node at position 01.[3]

The semantic representation of an elementary γ is called $\sigma(\gamma)$. (2) shows the semantic representations of $\alpha_1, \alpha_2, \alpha_3$ and β.

(2)

$\sigma(\alpha_1)$: $l_1 : \mathsf{love}(x_1, x_2)$ / arg: $\langle x_1, 00\rangle, \langle x_2, 011\rangle$

$\sigma(\alpha_2)$: $\mathsf{john}(x)$ / arg: –

$\sigma(\alpha_3)$: $\mathsf{mary}(y)$ / arg: –

$\sigma(\beta)$: $\mathsf{always}(s_1)$ / arg: s_1

Roughly, a semantic representation consists of a conjunctively interpreted set of formulas (typed lambda-expressions) and a set of argument variables. The formulas may have propositional labels $l_1, l_2, \ldots$. Argument variables may be linked to positions in the elementary syntactic tree, as it is the case in $\sigma(\alpha_1)$.

In each application of one semantic representation to another semantic representation, a partial assignment function f maps some of the argument variables of the first representation to labels or free variables of the second representation. This assignment function f is restricted to the two elementary semantic representations involved in this specific dependency relation. Furthermore, some of the argument variables might be related to argument slots in the syntactic structure, which would give a further restriction. After having applied the assignment f in a semantic composition, the union of the two semantic representations is built.

The derivation structure, shown in Figure 20, indicates that $\sigma(\alpha_1)$ is applied to $\sigma(\alpha_2)$ and $\sigma(\alpha_3)$, and $\sigma(\beta)$ is applied to $\sigma(\alpha_1)$. In each step, with an edge in the derivation tree directed from γ_1 to γ_2, $\sigma(\gamma_1)$ is applied to $\sigma(\gamma_2)$ depending on an assignment function that maps some of the arguments of $\sigma(\gamma_1)$ to elements in $\sigma(\gamma_2)$. The linking of argument variables and positions is supposed to restrict the possible assignment functions as follows: In a substitution derivation step at a position p, f is defined exactly for all argument variables linked to p. In an adjunction step, f is defined for all argument variables that are not linked to any positions. When applying $\sigma(\alpha_1)$ to $\sigma(\alpha_2)$, because of the linking between the position of the subject NP and the variable x_1 and since this NP is replaced by α_2, the assignment function f for this combination must be such that $f(x_1) = x$, and for x_2, f is not defined. In the same way, the assignment for the other substitution step is restricted. For the adjunction step, $\sigma(\beta)$ is applied to $\sigma(\alpha_1)$ assigning l_1 to s_1. As a result we obtain the semantic representation shown in (3).

(3)

l_1 : $\mathsf{love}(x, y)$
$\mathsf{john}(x)$
$\mathsf{mary}(y)$
$\mathsf{always}(l_1)$
arg: –

(3) is conjunctively interpreted, i.e. roughly, (3) is true in some world w iff $\mathsf{john}(x)$ and $\mathsf{mary}(y)$ and $\mathsf{always}(p)$ are true in w where p is true in some world w' iff $\mathsf{love}(x, y)$ is true in w'.

A formal definition of semantic representations and the way they combine depending on the derivation structure is given in Section 4.

3.2. SEPARATION OF SCOPE INFORMATION FROM PREDICATE ARGUMENT RELATIONS

For a compositional semantics it is always a problem that in some cases the semantic contribution of a lexical item is discontinuous in the logical semantic representation. One of the problematic cases is the case of quantifiers. On the one hand, the contribution of a quantifier is connected to its syntactic position because corresponding to this position, an argument is added to the semantic representation. On the other hand, quantifiers can rise and may have wide scope, even if they are embedded in some other quantifier. So the syntactic structure does not directly reflect the scope relations.

Following the Montagovian tradition (Montague, 1974), we will consider quantifying phrases like *every* in *every man*, *every* in *every student* etc. as constants of type $\langle\langle e, \langle s, t\rangle\rangle, \langle\langle e, \langle s, t\rangle\rangle, \langle s, t\rangle\rangle\rangle$. In other words, quantifying phrases take two properties and then give a proposition.

As a notational variant, we will write $quant(x, p_1, p_2)$ instead of $quant(\lambda x.p_1, \lambda x.p_2)$. (In particular, x is treated as a free variable and thereby available as a possible value for argument variables.)

(4) Every dog barks

(5) $\mathsf{every}(x, \mathsf{dog}(x), \mathsf{bark}(x))$

(5) shows the truth-conditional semantics for (4). The contribution of a quantifier consists of two parts:

1. $\mathsf{every}(x, p_1, p_2)$ with propositions p_1 and p_2, and
2. $P(x)$ and x (as argument of bark)

The first part can rise and is responsible for scope relations. The second part contributes to the restriction of the quantifier and it inserts the semantic argument corresponding to the quantified NP. Therefore, the first part is applied to the proposition whereas the second part adds an argument, i.e. this is a downwards semantic dependency.

For this reason we propose to separate the contribution of a quantifying phrase into two elementary trees (and two corresponding semantic representations) that are added in different ways:

1. One auxiliary tree bearing the scope contribution of the quantifier, and
2. one initial tree bearing the predicate argument contribution.

The auxiliary tree consists just of one single node with label S. Figure 21 shows how the two elementary trees for *every* (the trees on the left) are added to the tree of *barks*.

The semantic representations for quantifying phrases are introduced in the next section.

The separation between scope information and contribution to the predicate argument structure is partly inspired by Muskens and Krahmer (Muskens, 1998; Muskens and Krahmer, 1998). These approaches also make use of the extended

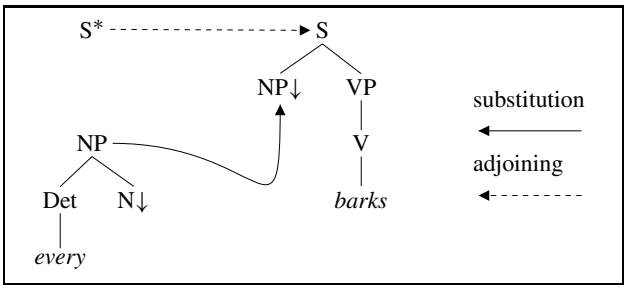


Figure 21. Combining *every* and *barks*.

local domains of TAG-like grammars where one can separate scope information from predicate-argument information and thereby avoid extra-mechanisms like quantifier raising to account for cases where the interpretation of a quantifier does not correspond to the (surface) position of the corresponding NP. However, Muskens and Krahmer do not make use of any locality restriction (as in tree-local TAGs for example) in the process of generating underspecified representations.

3.3. UNDERSPECIFIED QUANTIFIER SCOPE

In order to describe underspecified representations for scope ambiguities, we will apply the ideas in Hole Semantics (Bos, 1995) to our semantic representations, i.e. besides the labels already use in Section 3.1 we will use additional propositional metavariables called *holes*. A partial order on holes and labels describes the scope structure of a semantic representation. This way of underspecifying semantic representations is also used in other approaches, e.g. Underspecified Discourse Representation Structures (UDRS, Reyle, 1993). In this respect, our approach differs from Kallmeyer, 1999a; Kallmeyer, 1999b) where tree descriptions instead of trees are used, and the dominance relation can be directly interpreted as a description of scope relations. However, a problem with tree descriptions is that this is a very general approach, and a rather complicated system of axioms is needed in order to make sure that the grammar generates only the things one wants to have. Therefore, in our proposal, we will use trees (i.e. a lexicalized TAG) for the syntactic analysis combined with flat semantic representations enriched with labels and holes.

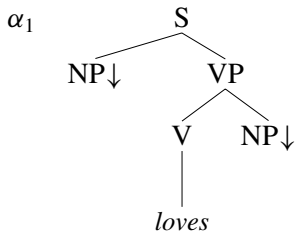
Although our approach has a resemblance to Hole Semantics, a crucial difference is that we have only propositional holes. As far as we can see, this might be sufficient. Concerning labels, however, any type is allowed. Labels of non-propositional type are necessary to make subformulas accessable, e.g. for modification. Another concept similar to our labels and holes are handles in MRS (Copestake et al., 1999). MRS, however, does not distinguish between handles acting as labels and handles acting as holes although these two different types of handles are present in MRS.

As an example for the use of Hole Semantics in order to obtain underspecification, we will consider the analysis of (6).

(6) Some student loves every course.

The elementary tree and elementary semantic representation for *loves* is shown in (7). Compared to the one proposed for *loves* in the last section, the truth-conditional formula corresponding to the whole proposition is labelled, and an additional constraint says that there is a hole subordinated by this labelled formula. The idea is that between this hole and the label, quantifiers might come in, i.e. quantifiers having scope over $\mathsf{loves}(x_1, x_2)$. Or, if there is nothing between h_1 and l_1, then in the end h_1 will be identified with l_1.

(7)

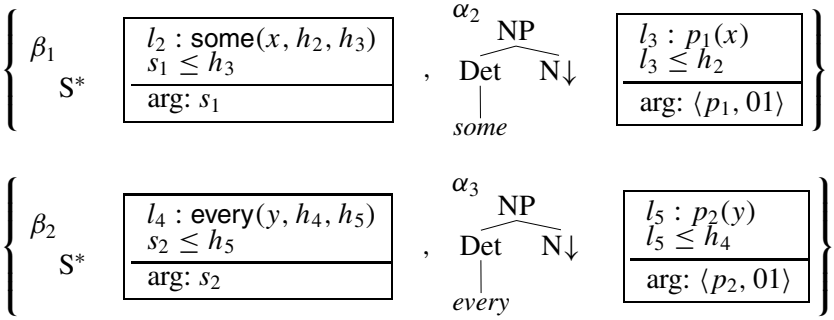


$$\boxed{\begin{array}{l} l_1 : \mathsf{loves}(x_1, x_2) \\ l_1 \leq h_1 \\ \hline \text{arg: } \langle x_1, 00\rangle, \langle x_2, 011\rangle \end{array}}$$

The key idea of the analysis of quantifiers we adopt here is that the contribution of a quantifier is separated into one predicate argument component and one scope component. Thereby, the scope of a quantifier does not strictly depend on the surface position of the quantifier. However, since tree-locality of the grammar must be respected, quantifiers cannot rise arbitrarily high. This locality restriction allows just the right amount of underspecification needed to treat scope ambiguities appropriately.

The elementary tree sets and semantic representations for the quantifying phrases *some* and *every* are shown in (8).

(8)



The contribution of a quantifier consists of two parts: on the one hand a quantifier adds an argument to the predicate-argument structure and on the other hand, it contributes some information about scope. This is separated in our analysis. The

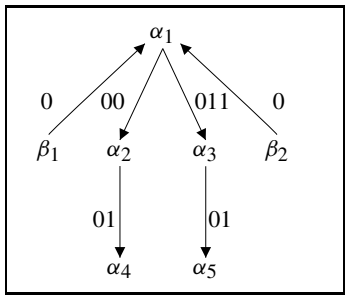


Figure 22. Derivation tree for (6).

auxiliary tree in the tree set of a quantifier, consisting of one single node, carries information about the variable corresponding to the quantifier and it introduces slots (h_2 and h_3 in the case of some) for the scope of the quantifier, i.e. its restriction and body. The NP part of the tree set is inserted as a syntactic argument and it contributes (a part of) the restriction of the quantifier. The argument variables p_1 and p_2 stand for the predicates denoted by the nouns in the NPs that will be added by substitution.

The separation into two parts also provides a separation of the proposition belonging to the restriction (l_3 in the case of *some*) on the one hand and the proposition belonging to the body on the other hand (s_1 in the case of *some*). As a consequence, when further adding, for example, a quantifier to the NP tree, the restriction of the first quantifier will be part of the body of this new quantifier since this is the only accessable proposition. The locality of derivations excludes to locate the body of the first quantifier in the body of the second. This is important for obtaining adequate constraints for quantifiers embedded in NPs. We will consider such examples in Section 5.1.

The elementary trees and semantic representations for the nouns *student* and *course* are shown in (9). They are very simple. q_1 and q_2 are labels. A noun denotes a predicate that is added to a quantifier by substitution, i.e. that is an argument of a quantifier.

(9)
$\left\{ \begin{array}{ll} \alpha_4 & \mathrm{N} \\ & \mid \\ & \textit{student} \end{array} \right.$ $\left| \begin{array}{l} q_1 : \text{student} \\ \hline \text{arg: } - \end{array} \right\}$ $\left\{ \begin{array}{ll} \alpha_5 & \mathrm{N} \\ & \mid \\ & \textit{course} \end{array} \right.$ $\left| \begin{array}{l} q_2 : \text{course} \\ \hline \text{arg: } - \end{array} \right\}$

The derivation tree for (6) is shown in Figure 22.

In standard TAG or MC-TAG a derivation structure as in Figure 22 is not possible since one cannot adjoin more than one auxiliary tree at one and the same node. However, as already proposed in Joshi and Vijay-Shanker (1999), we will allow this kind of multiple adjunction for quantifiers in order to account for scope ambiguities. The use of multiple adjunctions at a single node has already been

introduced by Schabes and Shieber in (1994). But in contrast to Schabes and Shieber (1994), we will allow multiple adjunctions in a much more restricted way. There are two reasons to do so. The first is that in cases of adjunct scope, multiple adjunctions are not linguistically adequate. We will look at this more closely in Section 3.4. The second reason is that combined with multicomponent adjunction, even in the tree-local case, unrestricted multiple adjunctions increase the generative capacity of the grammar considerably. This issue will be discussed more detailed in Section 3.5.

Figure 22 indicates that $\sigma(\alpha_1)$ is applied to $\sigma(\alpha_2)$ and $\sigma(\alpha_3)$ where x_1 is replaced by x and x_2 by y. Applying $\sigma(\beta_1)$ to $\sigma(\alpha_1)$ means replacing s_1 by l_1, and applying $\sigma(\beta_2)$ to $\sigma(\alpha_1)$ means replacing s_2 by l_1. Furthermore, when applying $\sigma(\alpha_2)$ to $\sigma(\alpha_4)$ and $\sigma(\alpha_3)$ to $\sigma(\alpha_5)$, p_1 is replaced by $(q_1 : \mathsf{student})$ and p_2 by $(q_2 : \mathsf{course})$. The labels q_1 and q_2 make the predicates accessable for the assignment function that maps p_1 to q_1 and p_2 to q_2 and thereby causes the replacing of p_1 and p_2 by $(q_1 : \mathsf{student})$ and $(q_2 : \mathsf{course})$ respectively. As a result, the semantic representation (10) is derived for (6).

(10)

$l_2 : \mathsf{some}(x, h_2, h_3), l_4 : \mathsf{every}(y, h_4, h_5),$ $l_1 : \mathsf{loves}(x, y), l_3 : (q_1 : \mathsf{student})(x), l_5 : (q_2 : \mathsf{course})(y)$ $l_3 \leq h_2, l_1 \leq h_3, l_5 \leq h_4, l_1 \leq h_5, l_1 \leq h_1$
arg: –

The constraints for scope order in the third line indicate that $\mathsf{student}(x)$ must be part of the restriction of some, $\mathsf{course}(y)$ must be part of the restriction of every, and $\mathsf{loves}(x, y)$ must be part of the body of some and the body of every. This leaves open whether some is in the body of every or every in the body of some. In other words, (10) is an underspecified representation in the sense that it describes two readings, wide scope of *some student* and wide scope of *every course*.

In order to obtain one of the readings described by the underspecified representation, a disambiguation mapping is needed. This is a bijection from holes to labels that is such that after having applied this mapping, the transitive closure of the resulting scope order is a partial order. In the case of (10), there are two possible disambiguation mappings:

$$\delta_1 : \begin{cases} h_1 \to l_2 \\ h_2 \to l_3 \\ h_3 \to l_4 \\ h_4 \to l_5 \\ h_5 \to l_1 \end{cases} \quad , \quad \delta_2 : \begin{cases} h_1 \to l_4 \\ h_2 \to l_3 \\ h_3 \to l_1 \\ h_4 \to l_5 \\ h_5 \to l_2 \end{cases}$$

δ_1 corresponds to wide scope of some and δ_2 to wide scope of every.

Formal definitions of the mechanisms sketched here can be found in Section 4.

In the approach presented here, the nuclear scope (i.e., the body) of a quantifier is partly specified by the elementary tree to which the scope part is adjoined. But it

does not depend on the specific node at which the adjunction takes place. However, it might be useful to view the attachment site of the scope tree as an indicator for scope and thereby perhaps to account for certain restrictions on relative quantifier scope. This is an issue we want to pursue in the future.

3.4. ADJUNCT SCOPE

As we have seen in the example in the preceding section, in order to obtain the desired syntactic derivations for quantifiers, it is necessary to allow multi-component adjunction and, furthermore, also to allow the adjunction of more than one auxiliary tree at one single node. For quantifiers, we need an elementary tree set with two trees, one auxiliary tree for the part that is responsible for the scope relations this quantifier occurs in, and one part that contributes the syntactic argument. Therefore we choose to use tree-local multicomponent TAGs instead of simple TAGs.

In the following, we will argue that unrestricted multiple adjunctions together with tree-local multicomponent derivations is not adequate and that we only need a restricted use of multiple adjunctions.

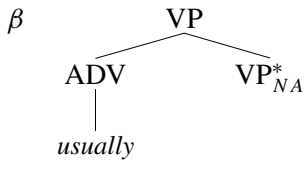
3.4.1. *Non-intersective modification*

(11) Pat allegedly usually drives a cadillac.

(11) is an example of adjunct scope taken from Bouma et al. (1998) involving two non-intersective adverbs. As pointed out in Bouma et al. (1998), the possible readings of (11) are restricted by the constraint that *usually* must be in the scope of *allegedly*. Considering only the readings where both adverbs are VP-modifiers, we therefore get three different scope orders: *allegedly* must have scope over *usually*, and the quantifier *a cadillac* can either have wide scope or be between the two adverbs or it can have narrow scope.

In our system, (12) is a natural elementary representation for VP-modifiers as *usually*:

(12)



l_1 : usually(h_1)
$s \leq h_1$
arg: s

Schabes and Shieber (1994) would argue that in (11), both adverbs are adjoined to the VP-node of *drives*, i.e. they would prefer multiple adjunction in this case. According to them, an adjunction of *allegedly* at the root of the auxiliary tree of *usually* corresponds to the reading where *allegedly* modifies only the adverb *usually* and not the whole VP. Schabes and Shieber propose to consider the fact that

in (11), *usually* must be in the scope of *allegedly* in the case where both are VP-modifiers as a consequence of the specific syntactic derivation order. However, one of our underlying assumptions was that the composition of the semantic representation depends only on the dependencies expressed in the derivation structure. In particular, it should be independent from syntactic derivation order.

Therefore, contrary to Schabes and Shieber, we assume that for tree sets containing single auxiliary trees, multiple adjunctions of several such trees at one and the same node are not allowed. The difference between adverbs modifying the whole VP and adverbs modifying only an embedded adverb is accounted for by adjoining in the first case at the VP-node and in the second case at the node with label ADV. The restriction that several adverbial modifier trees cannot be adjoined at the same node reflects our assumption about operator scope, namely that operators adjoined at the same node (even the same elementary tree) should be equivalent with respect to their scoping possibilities. In (11) where we have different scope properties, adjunction at the same node therefore should be excluded.

If multiple adjunction at the VP-node of *drives* is not allowed in this case, the only possible derivation is to adjoin *usually* to the VP-node of *drives*, and then to adjoin *allegedly* to *usually*. With this derivation, the desired restriction is obtained since the argument of *allegedly* is the label of $\mathsf{usually}(h_1)$. (13) shows how the adverbs combine with the semantic representation σ_1 of *Pat drives a cadillac*.

(13)

σ_1

$l_1 : \mathsf{drive}(e, x, y), \mathsf{Pat}(x),$ $l_2 : \mathsf{a}(y, h_2, h_3), l_3 : \mathsf{cadillac}(y)$ $l_1 \leq h_1, h_3 \leq h_2, l_1 \leq h_3$
arg: –

σ_2

$l_4 : \mathsf{usually}(h_4)$ $s \leq h_4$
arg: s

σ_3

$l_5 : \mathsf{allegedly}(h_5)$ $s \leq h_5$
arg: s

$[\sigma_3([\sigma_2(\sigma_1)]_{f_1})]_{f_2}$
with $f_1(s) = l_1$ and $f_2(s) = l_4$:

$l_1 : \mathsf{drive}(e, x, y), \mathsf{Pat}(x),$ $l_2 : \mathsf{a}(y, h_2, h_3), l_3 : \mathsf{cadillac}(y)$ $l_4 : \mathsf{usually}(h_4), l_5 : \mathsf{allegedly}(h_5)$ $l_1 \leq h_1, h_3 \leq h_2, l_1 \leq h_3, l_1 \leq h_4, l_4 \leq h_5$
arg: –

Similar examples are sentences like (14).

(14) Robin reboots the Mac frequently intentionally.

According to Bouma et al. (1998), the second adverb must have scope over the first. This can be obtained in the same way as in (11) if multiple adjunctions for single modifier trees are disallowed.

(15) shows examples of scope ambiguities between two modifiers.

(15) a. Usually, Pat allegedly drives a cadillac.

b. Sandy rarely visited a friend because of El Niño. (Bouma et al., 1998).

In (15)a. the relative scope of the two adverbial modifiers is not specified, and in (15)b. both scope orders of *rarely* and *because of El Niño* are possible. With our analysis, this is in fact the case, since in (15)a. and b. both modifiers adjoin to the elementary tree of the matrix verb, one at the VP-node, and the other one at the S-node. With respect to the derivation structure, these adjunctions are not distinguished except for the positions labeling the corresponding edges. In case of an adjunction, the positions do not influence the choice of a semantic assignment, and consequently there is no difference between the two modifiers concerning possible scope relations.

We do not want to claim that there are no cases of scope restrictions at all for non-adjacent adverbs, but (11) and (15) have clearly shown that in case of adjacency we want to obtain a restriction whereas in case of non-adjacency this is at least in general not the case.

A problem with the semantic representations proposed here for adverbs is that they allow adverbs that are not blocked by another adverb to have arbitrarily wide scope: in (11), l_5 can be arbitrarily high, and this would even be the case for adverbs occurring in embedded clauses. In order to overcome this problem, one needs to modify the semantic representations such that an adverb takes scope over the label of the proposition it adjoins to (this is already the case) and, additionally, the adverb is blocked by the hole belonging to this proposition. To obtain this, the semantic representations of adverbs might be as follows:

(16)

$l_1 : \mathsf{usually}(h_1)$
$l_1 \leq h'_1, h'_1 \leq h, s \leq h_1$
arg: s, h

In the semantic representation in (16), h is a hole variable, i.e., a variable that takes a hole as its value. The additional hole h'_1 above l_1 is needed since it will be used to block further adverbs adjoined to this adverb.

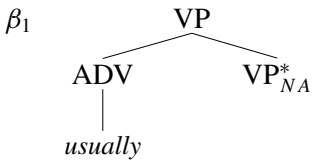
The examples in this section have shown that multiple adjunctions with tree sets containing only auxiliary trees would not be adequate.

Another reason to restrict the use of multiple adjunctions is that the idea behind tree sets with one initial tree and one auxiliary tree allowing multiple adjunction is the separation of predicate argument from scope semantics. Therefore, the scope bearing part (the auxiliary tree) should not interfere with syntactic properties. In particular, when adjoining such a scope bearing auxiliary tree, this should not modify the features of the node involved in this adjunction. Therefore we propose to restrict the use of multiple adjunctions such that it is allowed only for trees consisting of one single node with completely unspecified feature structures. This means that in the particular case of quantifiers, the auxiliary tree carrying the scope information can be adjoined to an elementary tree whenever this elementary tree also allows a substitution adding an initial NP tree. In Section 5.1 we will see that this is in fact more adequate than saying that scope bearing auxiliary trees must be adjoined at nodes with label S because there are cases where quantifiers are added to non-sentential argument structures.

3.4.2. *Recursive modifiers*

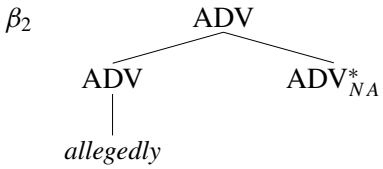
The readings of (11) where *allegedly* recursively modifies *usually* and not the whole VP *usually drives a cadillac* involve a different auxiliary tree and also a different semantic representation for *allegedly*. The elementary tree of *allegedly* does not adjoin at the VP-node but at the ADV-node of the elementary tree of *usually*. In order to make *usually* accessable for modification, the semantic representation of *usually* as VP-modifier must be slightly modified, we introduce a label for the adverb.

(17)



$l_1 : (m_1 : \text{usually})(h_1)$
$s \leq h_1$
arg: s

(18)



$m_2 : \text{allegedly}(m)$
arg: m

Adjoining β_2 to the ADV-node of β_1 (and thereby applying the semantic representation of β_2 to the one of β_1) gives the semantic representation shown in (19).

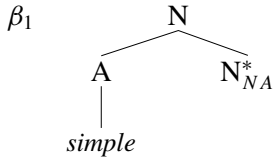
(19) $[\sigma(\beta_2)(\sigma(\beta_1))]_f$ with $f = \{m \rightarrow m_1\}$:

$l_1 : (m_2 : \mathsf{allegedly}(\mathsf{usually}))(h_1)$ $s \leq h_1$
arg: s

In the same way, examples like (20), (taken in a slightly simplified form from (Kasper, 1998) are analyzed.

(20) Bob shows an [[apparently] simple] example.

(21)



$\lambda z[(q_2 : \mathsf{simple})(z) \wedge p_1(z)]$
arg: p_1

Adding this to σ_1 in (22) with $p_1 \rightarrow q_1$ gives σ_2:

(22)

σ_1

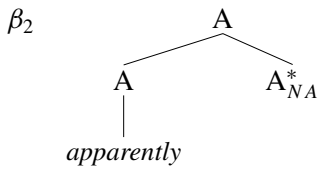
$l_1 : \mathsf{show}(x, y), \mathsf{Bob}(x),$ $l_2 : \mathsf{a}(y, h_2, h_3), l_3 : (q_1 : \mathsf{example})(y)$ $l_1 \leq h_1, l_3 \leq h_2, l_1 \leq h_3$
arg: –

σ_2

$l_1 : \mathsf{show}(x, y), \mathsf{Bob}(x),$ $l_2 : \mathsf{a}(y, h_2, h_3), l_3 : (q_2 : \mathsf{simple})(y) \wedge \mathsf{example}(y),$ $l_1 \leq h_1, l_3 \leq h_2, l_1 \leq h_3$
arg: –

For *apparently*. the elementary tree and semantic representation in (23)are chosen. Applying this to σ_2 with q_2 assigned to p leads to (24).

(23)



$q_3 : \mathsf{apparently}(p)$
arg: p

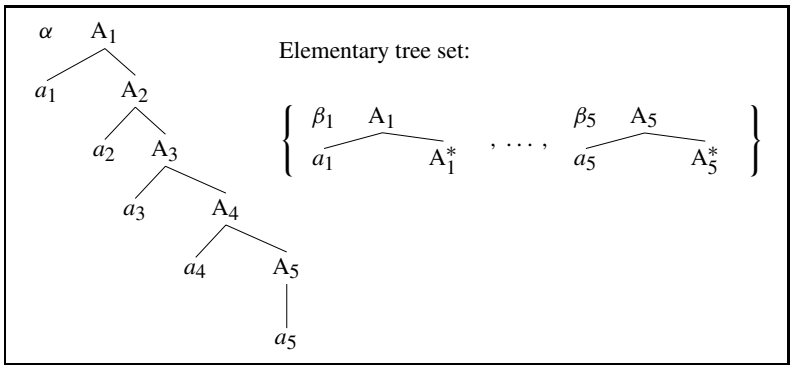


Figure 23. Combining multicomponent tree sets and multiple adjoining.

(24)

l_1 : show(x, y), Bob(x), l_2 : a(y, h_2, h_3), l_3 : $(q_3$: apparently(simple))(y) $\wedge$ example(y), $l_1 \leq h_1, l_3 \leq h_2, l_1 \leq h_3$
arg: –

3.5. MATHEMATICAL MOTIVATION FOR RESTRICTIONS ON MULTIPLE ADJUNCTION

It has been shown that tree-local MC-TAGs are strongly equivalent to TAGs, i.e., they both have the same strong generative capacity. In other words, the use of tree-local multicomponent derivations instead of standard TAG-derivations does not increase the generative capacity of the grammar.

More problematic is the issue of multiple adjunctions, i.e. adjunctions of several auxiliary trees at the same node. More precisely, the combination of multiple adjunctions with tree-local MC-TAGs causes problems. In certain cases, we need this in order to deal adequately with scope. One assumption underlying our approach is that the derivation structure reflects semantic dependencies. This means that for the specific case of scope relations, two operators showing the same properties concerning scope must be added to the same elementary tree.

Schabes and Shieber (1994) have shown that multiple adjunctions with TAGs do not increase the generative capacity of the grammar. However, if we allowed multiple adjunctions with tree-local MC-TAGs in a completely unrestricted way, we would considerably extend the generative power of the grammar formalism. As an example, consider the MC-TAG in Figure 23.

With the unrestricted version of multiple adjunction, in each derivation step in Figure 23, the five auxiliary trees in the elementary tree set would be adjoined to

the five nodes with nonterminal labels in α. This would generate $\{a_1^n \dots a_5^n \mid n \geq 1\}$ which is no TAL. It is obvious, that such a grammar can be found for any language $\{a_1^n \dots a_k^n \mid n \geq 1\}$ for some $k \in \mathbb{N}$.

This example shows that, although tree-local multicomponent derivations and multiple adjunctions do not increase the generative capacity when considered separately, the combination of the two is a problem in this respect. For this reason and for the reasons mentioned in the previous section, we will restrict multiple adjunctions to certain kinds of auxiliary trees, so-called *scope auxiliary trees*.

4. Formal Definition of the Syntax-Semantics Interface

In this section, we will give a formal definition of the objects and mechanisms motivated in the previous section. Section 4.1 defines the possible derivation trees for a given TAG in such a way that the derivation is tree-local and multiple adjunctions are restricted. Furthermore, the construction of the derived tree from the derivation tree is specified. In Section 4.2 semantic representations are defined together with their composition operation, a disambiguation mechanism is introduced, and the interpretation of disambiguated semantic representations is defined. Section 4.3 finally specifies how to obtain a semantic representation for a specific derivation structure.

4.1. TREE-LOCAL MC-TAG WITH RESTRICTED MULTIPLE ADJUNCTION

In the following, we will specify how to obtain a derived tree in a TAG from a given derivation tree, we will define scope auxiliary trees, and then we will give a definition of the derivation trees allowed in our MC-TAG. Scope auxiliary trees are degenerate auxiliary trees, they consist just of a single node and do not contribute anything to the syntactic structure. The derivation trees are such that the derivations must be tree-local and multiple adjunction is only allowed for scope auxiliary trees and only in cases where these trees occur together with one initial tree in an elementary tree set.

4.1.1. *Derivation Trees*

In order to restrict multiple adjunctions to certain kinds of trees, we have to define the possible derivation trees. Before doing this we will define scope auxiliary trees. These are the only auxiliary trees for which multiple adjunctions are allowed.

DEFINITION 4.1 (Scope auxiliary tree)
Let G be a TAG and β an auxiliary tree. β is a scope auxiliary tree iff

1. β consists of only one single node u, and
2. the top and bottom feature structures of u are empty.

For the derivation trees we suppose that the underlying grammar is a multi-component TAG (MC-TAG). This means that besides initial and auxiliary trees we have a set of pairwise disjoint elementary tree sets, i.e. tree sets containing initial and auxiliary trees that are added simultaneously.

A derivation tree must be such that the derivations are tree-local (all trees belonging to the same elementary set must be added to the same elementary tree), and multiple adjunction is only allowed for scope auxiliary trees that form a tree set together with one initial tree.

DEFINITION 4.2 (Derivation tree)
Let G be a MC-TAG.

A tree T whose node labels are the union of certain elementary tree sets in G and whose edges are labelled by positions of nodes in the tree that is label of the mother node, is called a tree-local derivation tree with multiple scope adjunctions iff

1. for each elementary tree set Γ: if there is a $\gamma \in \Gamma$ labeling a node u in T, then for all $\gamma' \in \Gamma$, $\gamma \neq \gamma'$: γ' is label of a sister of u (tree-locality).
2. if there are trees $\gamma, \gamma_1, \gamma_2$ such that $\langle \gamma, \gamma_1 \rangle$ and $\langle \gamma, \gamma_2 \rangle$ are edges in T that are labelled by the same position p, then for $i \in \{1, 2\}$, γ_i is a scope auxiliary tree in G, and there is an initial tree α such that $\{\gamma_i, \alpha\}$ is an elementary tree set in G (multiple scope adjunctions).

4.1.2. *Derived trees*

In the standard TAG definition of adjunction, a node u in the old tree is replaced by an auxiliary tree. Consequently, after having performed the adjunction, u is no longer there and in particular not available for further adjunctions. Therefore we have to give a slightly different definition of adjunction. Roughly, after an adjunction step, there is a node that is considered as being part of the old tree and at the same time part of the adjoined auxiliary tree. We will follow Schabes and Shieber (1994) with these definitions.

Schabes and Shieber (1994) define the derived tree on the basis of a derivation tree. All substitutions and adjunctions take place at certain positions in the tree that is already derived. Whether a node belongs to a specific elementary tree or not is not considered in this definition.

DEFINITION 4.3 (Derived tree of a derivation tree)
Let T be an ordered derivation tree with respect to a TAG G. The derived tree of T, written $\mathcal{D}(T)$ is defined as follows:

$$\mathcal{D}(T) := \begin{cases} \gamma & \text{if } T \text{ is a trivial tree of one node labelled with the elementary tree } \gamma \\ \gamma[\mathcal{D}(T_1)/p_1] \ldots [\mathcal{D}(T_n)/p_n] & \text{if } T \text{ is a tree with root node labelled with the } \gamma \text{ elementary tree and with } k \text{ child subtrees } T_1, \ldots, T_n \text{ whose edges are labelled with positions } p_1, \ldots, p_n \end{cases}$$

Here, $\gamma[\mathcal{D}(T_1)/p_1]\ldots[\mathcal{D}(T_n)/p_n]$ specifies the adjunction or substitution (depending on which elementary tree is added) of trees T_1 through T_n at positions p_1 through p_n (in this order). We suppose an appropriate updating of the tree addresses of any later adjunction to reflect the effect of earlier adjunctions that occur at addresses dominating the address of the later adjunction.

This definition does not exclude multiple adjunctions. Even if there are positions p_i, p_j with $i \neq j$, $1 \leq i, j \leq n$ and $p_i = p_j$, the derived tree is well defined.

With the restricted use of multiple adjunctions, it is clear that even in combination with tree-local multicomponent derivations, the strong generative capacity of the grammar is still the same as in the case of standard TAG.

For the derived tree of an ordered derivation tree satisfying the conditions for a tree-local derivation tree with multiple scope adjunctions, the linear order between sisters is not important. The derived tree is the same, no matter which linear order is chosen for sisters. But for the corresponding semantic representation it might be important.

4.2. FORMAL DEFINITIONS FOR SEMANTICS

4.2.1. *Semantic Representations*

The formulas in our semantic representations are typed. Types are defined in the usual recursive way, starting from basic types e, v, s and t for individuals, events, situations and truth values. For each type, there is not only a set of constants of this type and a set of variables but also a set of labels. Suppose that for type T, C_T is the set of constants, V_T the set of variables and L_T the set of labels of type T.

The use of propositional labels has already been motivated in the preceding section. Labels of other types can be useful in order to refer to subformulas, as in the semantic representations of *student* and *course* in the following section or as in terms like $\mathsf{usually}(\mathsf{hate}(e, x, y))$ where one might want to modify only $\mathsf{usually}$ and not the whole proposition. This is possible when the subformula has a label, e.g. $(m : \mathsf{usually})(\mathsf{hate}(e, x, y))$ where $m \in L_{\langle\langle s,t\rangle,\langle s,t\rangle\rangle}$.

Following Bos (1995), we will enrich our formulas with holes. Holes can be considered as a special kind of metavariables ranging over propositional labels (i.e., labels of type $\langle s, t\rangle$). They differ from the variables in the sets V_T since the values of holes are specified by disambiguation mappings. Even for holes, we need variables. An example is the semantic representation of *simple* in (21) (see p. 29). Other examples will be considered later when dealing with relative clauses. The use of hole variables is restricted in the following way: if a hole variable occurs in a set of terms in a semantic representation, it must also be one of its argument variables. This guarantees that hole variables disappear in the course of a derivation because they are replaced by holes. The set of holes is called H and the set of hole variables is V_H.

DEFINITION 4.4 (Terms with labels and holes)

1. For each type T, each $c_T \in C_T$ and each $v_T \in V_T$ is an unlabelled term of type T.
2. For each type T, each unlabelled term τ of type T and each label $l \in L_T$, $(l : \tau)$ is a labelled term of type T.
3. For all types T_1, T_2 and each (possibly labelled) terms τ_1 of type $\langle T_1, T_2 \rangle$ and τ_2 of type T_1, $\tau_1(\tau_2)$ is an unlabelled term of type T_2.
4. For each type T, each term τ of type $\langle \langle s, t \rangle, T \rangle$, and each $h \in H \cup V_H$, $\tau(h)$ is an unlabelled term of type T.
5. For all types T_1, T_2, each term τ of type T_2 and each $x \in V_{T_1}$, $\lambda x.\tau$ is an unlabelled term of type $\langle T_1, T_2 \rangle$.
6. Nothing else is a term.

Brackets will be omitted in cases where the structure of a formula is still unambiguously given.

A semantic representation is a set of such terms together with constraints on scope order, i.e. subordination constraints, and a set of argument variables. (The links between argument variables and positions of nodes are not part of the semantic representations but part of the syntax-semantics interface.)

DEFINITION 4.5 (Semantic representation)
A semantic representation is a triple $\langle \mathcal{T}, \mathcal{C}, \mathcal{A} \rangle$ such that:

- $\mathcal{T}$ is a set of terms with labels and holes, such that each label, hole or hole variable occurs at most once in $\mathcal{T}$.
- $\mathcal{C}$ is a set of constraints $x \leq y$ where each $z \in \{x, y\}$ is either a propositional label or a hole occurring in $\mathcal{T}$ or a propositional variable or a hole variable.
- $\mathcal{A}$ is a subset of the union of the set of free variables of $\mathcal{T}$ and of the set of variables occurring in $\mathcal{C}$. Each hole variable occurring in $\mathcal{T}$ or $\mathcal{C}$ occurs also in $\mathcal{A}$.

$\mathcal{A}$ is called the argument set of $\langle \mathcal{T}, \mathcal{C}, \mathcal{A} \rangle$.

A variable is free in a term iff it is not bound by a λ-operator and it is free in a set of terms iff it is free in one of the terms. This means in particular that quantifiers like some in (10) (see p. 24), that are treated as constants do not bind variables in this syntactic sense. Therefore, in (10), x and y are free variables.

The restriction that each label, hole and hole variable occurs at most once in $\mathcal{T}$ is motivated by the intuition that labels and holes stand for subformulas and this should be unique.

The constraints in $\mathcal{C}$ restrict the possible scope orders. Besides these constraints, also the terms in $\mathcal{T}$ contain information about possible scope orders. A hole or label that is in the scope of a hole h occurring in some term labelled l cannot have scope over l. Furthermore, scope order is transitive. The ordering relation on holes and

labels specified in such a way by $\mathcal{C}$ and $\mathcal{T}$, is called *subordination*. Its definition is more or less taken from Bos (1995).

DEFINITION 4.6 (Subordination)
Let $\sigma = \langle \mathcal{T}, \mathcal{C}, \mathcal{A} \rangle$ be a semantic representation with holes and hole variables H_σ, propositional labels L_σ and free propositional variables V_σ.

$SUB_\sigma \subseteq (H_\sigma \cup L_\sigma \cup V_\sigma) \times (H_\sigma \cup L_\sigma \cup V_\sigma)$ is called the subordination relation of σ iff

1. for all $k \in H_\sigma \cup L_\sigma \cup V_\sigma$: $\langle k, k \rangle \in SUB_\sigma$,
2. for all k, k' with $k \leq k' \in \mathcal{C}$: $\langle k, k' \rangle \in SUB_\sigma$.
3. for all $l \in L_\sigma$ and $k \in H_\sigma \cup L_\sigma \cup V_\sigma$ such that there is a $l : \tau \in \mathcal{T}$, and k occurs in τ: $\langle k, l \rangle \in SUB_\sigma$ and $\langle l, k \rangle \notin SUB_\sigma$, and
4. for all k, k', k'': if $\langle k, k' \rangle \in SUB_\sigma$ and $\langle k', k'' \rangle \in SUB_\sigma$, then $\langle k, k'' \rangle \in SUB_\sigma$.
5. Nothing else is in SUB_σ.

4.2.2. *Semantic Composition*

Next, we have to define the way semantic representations are combined with each other. The idea is that, when applying one semantic representation σ_1 to another semantic representation σ_2, some of the arguments of σ_1 are mapped to free variables (except hole variables), holes or labels from σ_2, and apart from this, the union of the two semantic representations is built. The mapping from some of the arguments of σ_1 to values in σ_2 is given by an assignment function f. The choice of a suitable f depends on the specific derivation step. This issue will be treated in Section 4.3.

DEFINITION 4.7 (Composition of semantic representations)
Let $\sigma_1 = \langle \mathcal{T}_1, \mathcal{C}_1, \mathcal{A}_1 \rangle$ and $\sigma_2 = \langle \mathcal{T}_2, \mathcal{C}_2, \mathcal{A}_2 \rangle$ be two semantic representations with $V_{\sigma_1} \cap V_{\sigma_2} = \emptyset$, and let f be a partial function from $\mathcal{A}_1$ to the set of free variables (without hole variables), labels and holes occurring in σ_2. Let f be defined for the set D_f.

The result of applying σ_1 to σ_2 under the assignment f (written $[\sigma_1(\sigma_2)]_f$) is the semantic representation $\sigma = \langle \mathcal{T}, \mathcal{C}, \mathcal{A} \rangle$ with

1. Terms $\mathcal{T}$:
   (a) For all $\tau_2 \in \mathcal{T}_2$ that do not contain any $v \in f(D_f)$: $\tau_2 \in \mathcal{T}$.
   (b) For all $\tau_1 \in \mathcal{T}_1$, a term τ_1' can be obtained as follows:
      (i) $\tau_1' := \tau_1$.
      (ii) For all $x \in D_f$ occurring in τ_1' such that $f(x)$ is a variable: $\tau_1' := [\lambda(x)\tau_1'](f(x))$.
      (iii) For all $x \in D_f$ occurring in τ_1' such that $f(x)$ is a label, and $[\lambda y \tau](f(x) : \tau_2')$ is in $\mathcal{T}_2$ (perhaps after β-reduction) with τ containing no further occurrence of $f(x) : \tau_2'$:
      $\tau_1' := [\lambda y \tau]([\lambda x \tau_1'](\tau_2'))$.

For the resulting τ_1', $\tau_1' \in \mathcal{T}$ holds.

(c) These are all terms in $\mathcal{T}$.

2. Constraints $\mathcal{C}$:
   (a) For all constraints in $c \in \mathcal{C}_1 \cup \mathcal{C}_2$ that do not contain an $s \in D_f$: $c \in \mathcal{C}$.
   (b) For all constraints in $c \in \mathcal{C}_1 \cup \mathcal{C}_2$ that contain an $s \in D_f$: $c[s/f(s)] \in \mathcal{C}$, where $c[a/b]$ stands for the result of replacing a with b in c.
   (c) These are all constraints in $\mathcal{C}$.
3. The argument set of σ is $(\mathcal{A}_1 \cup \mathcal{A}_2) \setminus D_f$.

As a special case of (iii), namely when $\lambda y \tau$ is the identical mapping, we get the following: if $f(x)$ is a label, and $f(x) : \tau_2'$ is in $\mathcal{T}_2$, then the new τ_1' is $[\lambda(x)\tau_1'](\tau_2')$, i.e. all occurrences of x are simply replaced by τ_2'.

We assume that all resulting formulas are transformed by β-reduction.

For this definition, we implicitly made the assumptions that f is such that the resulting semantic representation contains only well typed terms. This means first that for all $x \in D_f$, x and the corresponding $f(x)$ must be of the same type, and second that (iii) only occurs with modifiers τ_1', i.e. with $\lambda x \tau_1'$ of type $\langle T, T \rangle$ where x is of type T. The definition of suitable assignments f, given in Section 4.3, is such that these assumptions hold.

The case (iii) is similar to adjunction on the syntactic level: a labelled subformula is first removed, then a new subformula is inserted, and finally the removed subformula becomes an argument of the subformula that has been added. The label of the original subformula is removed. This is motivated by the observation that terms of the form $mod_1(mod_2(arg))$ can be built by first modifying arg by mod_2 and applying mod_1 to the result $mod_2(arg)$. Therefore after having added the first modifier, the label of arg can disappear since arg needs no longer to be accessable for further modification. In other words, more than one modification of a labelled term is not possible. This is similar to no adjunction constraints for foot nodes in auxiliary trees.

As an example for the case (iii) consider (25):

(25) Some former student loves every course.

(26) σ

$l_2 : \mathsf{some}(x, h_2, h_3), l_4 : \mathsf{every}(y, h_4, h_5),$ $l_1 : \mathsf{loves}(x, y),$ $l_3 : (q_1 : \mathsf{student})(x), l_5 : (q_2 : \mathsf{course})(y)$ $l_3 \leq h_2, l_1 \leq h_3, l_5 \leq h_4, l_1 \leq h_5, l_1 \leq h_1$
arg: –

σ_{mod}

$q_3 : (\mathsf{former}(p))$
arg: p

The semantic representation for (25) can be generated by applying a semantic representation σ_{mod} for *former* in (26) to the semantic representation σ of *some*

student loves every course. This application must be such that the assignment f maps the argument variable of σ_{mod} to the label of student. The application of σ_{mod} to σ with this assignment is shown in (27).

(27) $[\sigma_{mod}(\sigma)]_f$ with $f : \{p \to q_1\}$:

$l_2 : \mathsf{some}(x, h_2, h_3), l_4 : \mathsf{every}(y, h_4, h_5),$
$l_1 : \mathsf{loves}(x, y),$
$l_3 : (q_3 : (\mathsf{former}(\mathsf{student})))(x), l_5 : (q_2 : \mathsf{course})(y)$
$l_3 \leq h_2, l_1 \leq h_3, l_5 \leq h_4, l_1 \leq h_5, l_1 \leq h_1$
arg: –

In the resulting semantic representation, the new predicate former(student) is accessable via the label q_3.

Labels as q_3 in $[\sigma_{mod}(\sigma)]_f$ are sometimes left aside if they are not needed for any further derivation steps.

Other examples for the case (iii) were shown in Section 3.4.2 where recursive modification was considered.

Note that the result of a composition must be a semantic representation. In particular, all propositional labels occurring in the constraint set must also occur in the set of terms. This means that a propositional label l that occurs in a scope constraint cannot be removed because of a modification of the proposition labelled l. Instead of directly modifying this proposition, a term with a hole h is introduced together with a new constraint $l \leq h$.

4.2.3. *Disambiguation and Interpretation*

A disambiguation mapping for a given semantic representation σ is the same as a possible plugging in Bos (1995), namely a bijection from the set of holes in σ to the set of labels in σ that is such that the subordination constraints are respected. This means that after mapping all holes to labels, the transitive closure of the subordination in σ is a partial order. Furthermore, for each two labels there should be a label subordinated by both of them. In other words, the resulting structure must be a join semilattice. Additionally, for two holes or labels occurring in the same formula there must not be any hole or label that subordinates both of them. This last condition assures for example that nothing can be at the same time in the restriction and the body of a quantifier.

Disambiguation is done only when the derivation process is finished. Therefore, for the definition of a disambiguation mapping, we suppose that the argument set is empty. This means in particular that all hole variables have been replaced by holes.

DEFINITION 4.8 (Disambiguation mapping)
Let $\sigma = \langle \mathcal{T}, \mathcal{C}, \mathcal{A} \rangle$ be a semantic representation with $\mathcal{A} = \emptyset$. Let $L^{\sigma}_{\langle s,t \rangle}$ be the set of all propositional labels occurring in σ.

1. A bijection $\delta : H_\sigma \to L^{\sigma}_{\langle s,t \rangle}$ is a disambiguation mapping of σ iff
   for the homomorphism $\delta' : \langle H_\sigma \cup L^{\sigma}_{\langle s,t \rangle}, SUB_\sigma \rangle \to \langle L^{\sigma}_{\langle s,t \rangle}, SUB'_\sigma \rangle$ with $\delta'(h) := \delta(h)$ for all $h \in H_\sigma$ and $\delta'(l) := l$ for all $l \in L^{\sigma}_{\langle s,t \rangle}$,
   - the algebra $\langle L^{\sigma}_{\langle s,t \rangle}, SUB'^{*}_{\sigma} \rangle$ (with SUB'^{*}_{σ} being the transitive closure of SUB'_σ) is a join semilattice,
   - and for all $x_1, x_2 \in H_\sigma \cup L^{\sigma}_{\langle s,t \rangle}$: if there is an unlabelled term τ such that $\tau \in \mathcal{T}$ or $l : \tau \in \mathcal{T}$ for some label l and x_1 and x_2 occur in τ, then there is no $x_3 \in H_\sigma \cup L^{\sigma}_{\langle s,t \rangle}$ such that $\langle \delta'(x_3), \delta'(x_1) \rangle, \langle \delta'(x_3), \delta'(x_2) \rangle \in SUB'^{*}_{\sigma}$.

   Then we define $SUB_\delta := SUB'^{*}_{\sigma}$.
2. Let $\delta(\mathcal{T})$ be the result of replacing all holes h occurring in $\mathcal{T}$ by $\delta(h)$, and let $\delta(\mathcal{C})$ be the result of replacing all holes h occurring in $\mathcal{C}$ by $\delta(h)$. Then we define $\delta(\sigma) := \langle \delta(\mathcal{T}), \delta(\mathcal{C}) \rangle$.

For a disambiguated representation, we will give a model-theoretic semantics in such a way that, roughly, sets of terms are conjunctively interpreted. Such a model-theoretic semantics is defined only for semantic representations without any argument variables and where all terms in $\mathcal{T}$ are of propositional type.

DEFINITION 4.9 (Interpretation of semantic representations)
Let $\sigma = \langle \mathcal{T}, \mathcal{C}, \mathcal{A} \rangle$ be a semantic representation with $\mathcal{A} = \emptyset$ and $\mathcal{T}$ containing only terms of type $\langle s, t \rangle$, and let δ be a disambiguation mapping of σ with $\delta(\sigma) = \langle \mathcal{T}_\delta, \mathcal{C}_\delta \rangle$.

Let I be an interpretation function for constants and g an assignment for variables.

The interpretation of $\mathcal{T}_\delta$ under SUB_δ, I and g, written $[\![\mathcal{T}_\delta]\!]^{SUB_\delta}_{I,g}$, is recursively defined. For a given situation s,

- $[\![\mathcal{T}]\!]^{SUB_\delta}_{I,g}(s) = true$ iff $[\![\tau]\!]^{SUB_\delta, \mathcal{T}}_{I,g}(s) = true$ for all $\tau \in \mathcal{T}$ such that either τ is not labelled or its label is l_τ and there is no $l \in L_\sigma$ occurring inside some $\tau' \in \mathcal{T}$ with $\langle l_\tau, l \rangle \in SUB_\delta$.
- for all $l \in L_\sigma$ occurring inside some term: $[\![l]\!]^{SUB_\delta, \mathcal{T}}_{I,g}(s) = true$ iff $[\![\mathcal{T}_l]\!]^{SUB_\delta}_{I,g}(s) = true$
  where $\mathcal{T}_l := \{ l_\tau : \tau \mid l_\tau : \tau \in \mathcal{T} \text{ and } \langle l_\tau, l \rangle \in SUB_\delta \}$.
- for all types T and labelled terms $l_T : \tau_T$ of type T, $[\![l_T : \tau_T]\!]^{SUB_\delta, \mathcal{T}}_{I,g} := [\![\tau_T]\!]^{SUB_\delta, \mathcal{T}}_{I,g}$
- In all other cases, the interpretation is defined in the usual classical way with I giving the interpretations of constants, and g as an assignment for variables.

In general, $[\![\mathcal{T}]\!]^{SUB_\delta}_{I}(s) = true$ iff there is an assignment g such that $[\![\mathcal{T}]\!]^{SUB_\delta}_{I,g}(s) = true$.

The last part of this definition means that free variables are interpreted as existentially bound.

4.3. RELATION BETWEEN SYNTAX AND SEMANTICS

In this section, we will define the syntax-semantics interface, i.e. the way the syntactic tree sets are combined with semantic representations, and the way semantic representations are compositionally built depending on the corresponding syntactic derivation structure. In particular, the possible assignment functions f that are used when applying one semantic representation to another are defined depending on the corresponding derivation step in the syntactic TAG.

4.3.1. *The Grammar*

The syntax-semantics interface is a set of sets of triples, each of these triples consisting of an elementary tree γ from the syntactic MC-TAG, a semantic representation $\sigma(\gamma)$, and a relation ρ between the argument variables of $\sigma(\gamma)$ and positions of substitution nodes in γ. For each set of such triples, the trees occurring in this set form one elementary tree set in the MC-TAG.

DEFINITION 4.10 (Syntax-semantics interface)
The syntax-semantics interface is a pair $\langle G, \Sigma \rangle$ such that

1. G is a MC-TAG.
2. Σ is a set of sets of triples such that for each $S \in \Sigma$
   - for each $\langle \gamma, \sigma, \rho \rangle \in S$,
     - γ is an elementary tree in G,
     - $\sigma =: \langle \mathcal{T}, \mathcal{C}, \mathcal{A} \rangle$ is a semantic representation, and
     - $\rho \subset \mathcal{A} \times \{p \mid p$ is a position of a substitution node in $\gamma\}$ is a partial function, such that:
       if γ is an initial tree, then for all $x \in \mathcal{A}$ there is a position p with $\langle x, p \rangle \in \rho$.
   - the set $\{\gamma \mid$ there are σ and ρ such that $\langle \gamma, \sigma, \rho \rangle \in S\}$ is an elementary tree set in G.

Notation: For an elementary tree γ occurring in one of the triples, $\sigma(\gamma)$ is the corresponding semantic representation.

4.3.2. *Semantic Assignments and Derivation Edges*

The more interesting part of the syntax-semantics interface is the way the semantic representations combine with each other depending on the syntactic derivation structure. In order to obtain the semantic representation, it is not necessary to consider the specific syntactic trees or tree sets. The derivation structure is

sufficient to determine how to put the corresponding semantic representations together.

First, we will define the (partial) assignments f for the combination of two semantic representations that correspond to an edge in the derivation tree. We suppose that we have a derivation structure and a set of semantic representations corresponding to the elementary trees in the derivation tree. These semantic representations are such that without loss of generality the sets of variables (including hole variables) occurring in the semantic representations of two different elementary tree sets used in the derivation are disjoint.

Then, roughly, for each edge representing a substitution of some α for a node at position p in an elementary tree γ, $\sigma(\gamma)$ is applied to $\sigma(\alpha)$ with an assignment f that is defined for all argument variables in γ that are related to p. For each edge corresponding to an adjunction of some β to some elementary tree γ, $\sigma(\beta)$ is applied to $\sigma(\gamma)$ with an assignment f that is defined for all argument variables of β that are not related to any position in β.

The following definition specifies the assignments f of an edge in the derivation tree.

DEFINITION 4.11 (Semantic assignment of a derivation edge)
Let $\langle \gamma_1, \gamma_2 \rangle$ be an edge in a derivation tree that is labelled by p (p is a position in γ_1), and let $\sigma_1 := \sigma(\gamma_1)$ and $\sigma_2 := \sigma(\gamma_2)$ be the semantic representations of the two elementary trees. Without loss of generality suppose that $V_{\sigma_1} \cap V_{\sigma_2} = \emptyset$ holds. Let ρ_1 and ρ_2 be the relations with $\langle \gamma_1, \sigma_1, \rho_1 \rangle$, $\langle \gamma_2, \sigma_2, \rho_2 \rangle$ in Σ.

A semantic assignment of this edge is then a (total) function $f : D_f \to V_f$ such that:

1. If γ_2 is an initial tree, then
   - $D_f := \{x \in V_{\gamma_1} \mid \langle x, p \rangle \in \rho_1\}$, and
   - $V_f := V_{\sigma_2} \cup L_{\sigma_2} \cup H_{\sigma_2}$.
2. If γ_2 is an auxiliary tree, then
   - $D_f := \{x \in V_{\gamma_2} \mid x$ is argument of σ_2, and there is no p with $\langle x, p \rangle \in \rho_2\}$, and
   - $V_f := V_{\sigma_1} \cup L_{\sigma_1} \cup H_{\sigma_1}$.
3. For all $x \in D_f \cap V_H$, $f(x) \in H$ holds.
4. For all $x \in D_f$ such that there is a type T with $x \in V_T$, $f(x) \in V_T \cup L_T$ holds.
5. For all $x \in D_f$ such that there is a type T with $f(x) \in L_T$, and there is a subformula labelled by $f(x)$ in σ_1 or σ_2: There is exactly one term τ in σ_1 and σ_2 containing x, and for this τ holds that $\lambda x(\tau)$ is of type $\langle T, T \rangle$.

Note that an edge in a derivation tree may at least theoretically have more than one semantic assignment f: If there is more than one possible value for one of the argument variables in D_f, the choice of a semantic assignment for a derivation edge is not deterministic. However, in all the cases in this paper, there is just one possible assignment for each derivation edge.

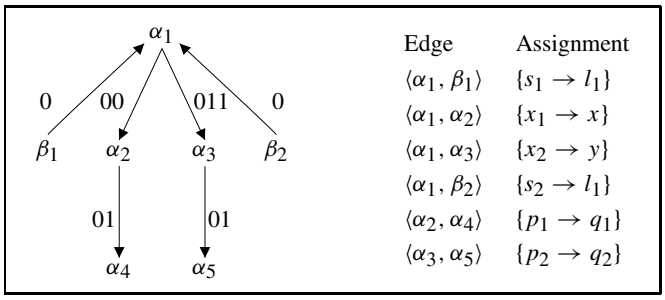


Figure 24. Derivation tree for (6) and corresponding assignments.

As an example of a derivation tree and the corresponding assignment, consider the derivation tree for (6) (*Some student loves every course.*) that was shown in Figure 22 and that is repeated in Figure 24 together with the corresponding assignments. (The elementary trees for (6) and their semantic representations are shown in (7)–(9) (see pp. 23–24).

4.3.3. *Semantic Representations of a Derivation Tree*

With this definition, the semantic representations corresponding to a whole derivation tree can be easily defined (there might be more than one for a specific derivation tree):

DEFINITION 4.12 (Semantic representation of a derivation tree)
Let T be a derivation tree, and without loss of generality let the set of semantic representations corresponding to the elementary trees in the derivation tree be such that for all sets $S_1, S_2 \in \Sigma$ involved in this derivation: $(\bigcup\{V_\sigma \cup L_\sigma \cup H_\sigma \mid \sigma \text{ occurs in } S_1\}) \cap (\bigcup\{V_\sigma \cup L_\sigma \cup H_\sigma \mid \sigma \text{ occurs in } S_2\}) = \emptyset$.

Suppose that for each edge in T, a semantic assignment has been chosen.

The semantic representation of T with respect to its semantic assignments, $\mathcal{S}(T)$, is defined as follows:

$$\mathcal{S}(T) := \begin{cases} \sigma(\gamma) \text{ if } & T \text{ is a trivial tree of one node labelled with the elementary tree } \gamma \\ \sigma(\gamma) \circ_{f_1} \mathcal{S}(T_1) \cdots \circ_{f_n} \mathcal{S}(T_n) & \\ \quad \text{else} & \text{where } T \text{ is a tree with root node } u \text{ labelled with the elementary } \gamma \text{ and with } n \text{ child subtrees } T_1, \ldots, T_n, \\ & f_i \text{ is the semantic assignment of the edge from } u \text{ to } T_i \text{ for } 1 \le i \le n, \\ & \text{and for all representations } \sigma, \sigma' \text{ and all } 1 \le i \le n: \\ & \sigma \circ_{f_i} \sigma' := \begin{cases} [\sigma(\sigma')]_{f_i} & \text{if the edge from } u \text{ to } T_i \text{ is a substitution edge} \\ [\sigma'(\sigma)]_{f_i} & \text{else} \end{cases} \end{cases}$$

The choice of the semantic assignments for the edges is not always deterministic. Therefore in general $\mathcal{S}(T)$ is not unique. But in all cases considered in this paper, $\mathcal{S}(T)$ is uniquely specified.

$\mathcal{S}(T)$ does not depend on the syntactic derivation order, i.e. on the linear precedence among the nodes of T. The reason is that each label can be assigned at most once to an argument variable occurring inside some term. After having performed the corresponding composition of semantic representations the label disappears.

5. Restrictions on Quantifier Scope

In this section we are concerned with constraints for possible scope orders of quantifiers. We only deal with constraints that are consequences of the specific structure of a sentence and that are strictly respected, independently from the specific quantifiers. Besides these constraints, there are several other factors that cause preferences of some readings over other readings, but these problems are left aside in this paper.

Furthermore, we do not consider referentially used quantifiers. It is well known that indefinites can always have wide scope (see for example Reyle, 1993) because they can be referentially used:

(28) John knows everybody who lives in a certain small town called XX.

Although relative clauses seem to constitute strict islands for quantifier raising, wide scope of the indefinite in (28) is possible. In order to account for this, one might adopt specific semantic representations for indefinites, at least for their referential use. However, in this paper, we will not consider these cases.

When looking more closely at examples of quantifier constraints, it becomes clear that two kinds of constraints can be distinguished, *logical* restrictions and *island* constraints.

5.1. LOGICAL RESTRICTIONS ON SCOPE ORDER

With the term "logical restriction" we mean constraints on quantifier scope that follow from the logical structure of the semantic representation. A well-known example is (29) (see also Hobbs and Shieber, 1987).

(29) Every representative of some company saw most samples.

There are three quantifiers in (29). Without restrictions on scope orders, (29) would therefore have $3! = 6$ different readings. However, at least one of these readings is excluded, namely the one where *every representative* has scope over *most samples*, and *most samples* outscopes *some company*.

The exclusion of this reading can be explained as follows: Suppose that x is the variable corresponding to the NP *every representative*, y corresponds to *some*

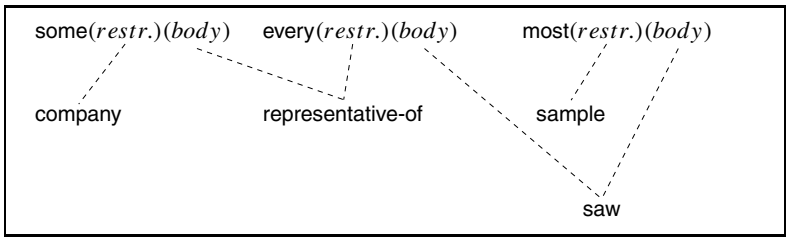


Figure 25. Scope restrictions for (29).

company and z corresponds to *most samples*. Then representative-of(x, y) is part of the restriction of *every*, and saw(x, z) must be part of the body of *every* and of the body of *most*. Consequently, if *most samples* is in the scope of *every*, it must be part of the body of *every*. Furthermore, if *some company* was outscoped by *most samples*, it also would be part of the body of *every*. But this is not possible, because representative-of(x, y) (which is part of the restriction of *every*) must be in the scope of *some company*, otherwise y would be a free variable in representative-of(x, y). (Here the term "free variable" is used in a sense where quantifiers like every bind variables.)

This constraint therefore can be considered as a result of the logical structure of quantifiers. Quantifiers have a restriction and a body and nothing can be part of the restriction and the body at the same time.

Sometimes it is claimed that in (29) *some company* outscoping *most samples* and *most samples* having scope over *every representative* is excluded. More generally, in inverse linking configurations (a quantifier phrase B inside a quantifier phrase A and B having scope over A) no other quantifier can be between B and A with respect to scope. This constraint however seems to be of a different nature than the constraints we examine in this paper and for the moment we leave it aside. We plan to deal with inverse linking in future work. Logically, such a scope order is possible since its semantic representation does not contain free variables:

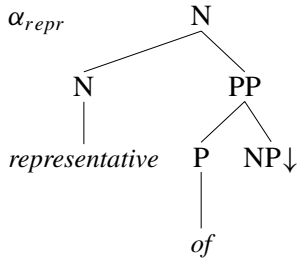
(30)
$$\begin{array}{llll} \mathsf{some}(y) & (\mathsf{company}(y)) & & \\ & (\mathsf{most}(z) & (\mathsf{sample}(z)) & \\ & & (\mathsf{every}(x) & (\mathsf{representative\text{-}of}(x)(y)) \\ & & & (\mathsf{saw}(x)(z)))) \end{array}$$

When considering *representative* as being part of the restriction of *every* and at the same time being part of the body of *some*, then we would obtain the scope constraints depicted in Figure 25. (A downward dashed edge stands for "has scope over", e.g. the restriction of some has scope over company and its arguments.) These constraints exclude exactly the reading we want to exclude.

With the semantic representations adopted above, we can get these constraints, since the two constraints added with each quantifier take care of the separation between restriction and body.

Representative of is a binary predicate that takes an NP as argument and returns a unary predicate. The elementary tree is the initial tree shown in (31) together with its semantic representation. The semantic representation is such that it contributes (1) a predicate p_1 that will be assigned to the predicate variable in the NP tree of *every* (similar to unary predicates as *sample*), and (2) a proposition labelled l_6 that will be part of the body of any quantifier added to *representative*, e.g., *some company* in this case. The constraint $l_6 \leq h_6$ makes sure that the new proposition is part of the predicate inserted in the proposition of the NP tree of *every*.

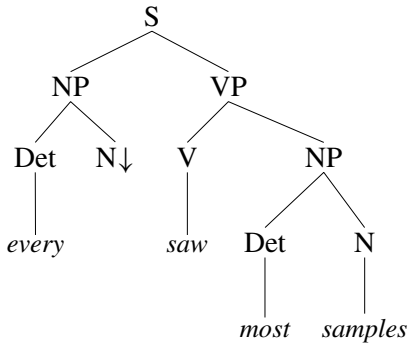
(31)



l_6 : representative-of(x_1, x_2) $p_1 : \lambda x_1 h_6$ $l_6 \leq h_6$
arg: $\langle x_2, 011 \rangle$

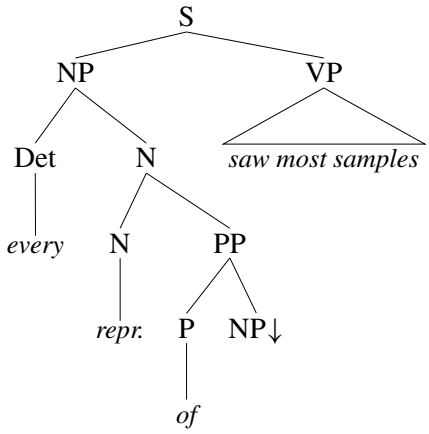
For the analysis of (29), first the syntactic tree and semantic representation for *every p saw most samples* shown in (32) are generated. In the next step, α_{repr} is added. This means that in the semantic representation p is replaced by the formula labelled p_1, namely $\lambda x_1 h_6$. This leads to (33).

(32)



l_2 : every(x, h_2, h_3) l_4 : most(y, h_4, h_5) l_1 : saw(x, y) $l_3 : p(x), l_5$: sample(y) $l_1 \leq h_1, l_3 \leq h_2, l_1 \leq h_3,$ $l_5 \leq h_4, l_1 \leq h_5$
arg: $\langle p, 01 \rangle$

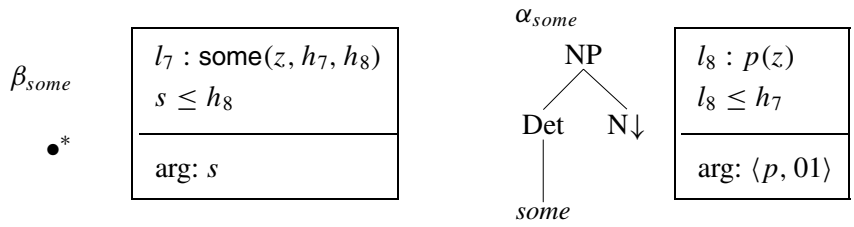
(33)



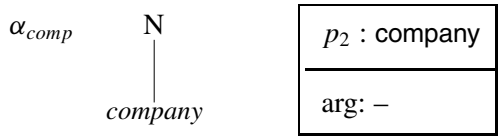
l_2 : every(x, h_2, h_3) l_4 : most(y, h_4, h_5) l_1 : saw(x, y) $l_3 : (\lambda x_1 h_6)(x)$ l_5 : sample(y) l_6 : representative-of(x_1, x_2) $l_1 \leq h_1, l_3 \leq h_2, l_1 \leq h_3,$ $l_5 \leq h_4, l_1 \leq h_5, l_6 \leq h_6$
arg: $\langle x_2, 011 \rangle$

The quantifier *some company* is not added to the matrix clause but to the noun of the subject NP (the root of α_{repr}). This shows that quantifiers can attach also to nodes with category N. More generally, they can attach to any node in an elementary tree γ as long as at the same time an NP tree is substituted for another node in γ. For *some* in this case, the elementary tree and semantic representation in (34) are adopted, and (35) shows the representation for *company*.

(34)



(35)



Adding β_{some} and α_{some} to α_{repr} and then adding α_{comp} to α_{some} gives the semantic representation (36).

(36)

l_7 : some(z, h_7, h_8), l_2 : every(x, h_2, h_3), l_4 : most(y, h_4, h_5) l_8 : company(z), l_3 : $(\lambda x_1 h_6)(x)$, l_5 : sample(y), l_1 : saw(x, y) l_6 : representative-of(x_1, z) $l_8 \leq h_7, l_6 \leq h_8, l_3 \leq h_2, l_1 \leq h_3, l_5 \leq h_4, l_1 \leq h_5, l_1 \leq h_1, l_6 \leq h_6$
arg: –

Wide scope of *some*, and *most* outscoping *every* (l_7>l_4>l_2) is possible with the disambiguation

$$\delta : \begin{cases} h_1 \rightarrow l_7 \\ h_2 \rightarrow l_3 \\ h_3 \rightarrow l_1 \\ h_4 \rightarrow l_5 \\ h_5 \rightarrow l_2 \\ h_6 \rightarrow l_6 \\ h_7 \rightarrow l_8 \\ h_8 \rightarrow l_4 \end{cases}$$

l_2>l_4>l_7 is excluded:

Suppose that a corresponding bijection δ exists. Then (in the join semilattice) $l_4 \leq \delta(h_2)$ or $l_4 \leq \delta(h_3)$ and $l_7 \leq \delta(h_4)$ or $l_7 \leq \delta(h_5)$.

Choosing a value for $\delta(h_1)$: Given $h_2, h_3 < l_2, h_4, h_5 < l_4 < l_2, h_6 \leq l_3, l_3 \leq h_2$ and $h_7, h_8 < l_7 < l_2, h_i < l_2$ for all $2 \leq i \leq 8$ holds and therefore $\delta(h_1) = l_2$.

Choosing a value for $\delta(h_2)$: Given that for all labels or holes x: $x \not\leq h_2$ or $x \not\leq h_3$:

- $l_1 \leq h_3, l_1 \leq h_5, h_5 < l_4 < l_2 \Rightarrow l_1, l_4 \leq h_3$
- $l_5 \leq h_4 < l_4 \Rightarrow l_5 \leq h_3$
- $l_6 \leq h_8 < l_7 < l_4 \Rightarrow l_6 \leq h_3$
- $l_7 < l_4 \Rightarrow l_7 \leq h_3$
- $l_8 \leq h_7 < l_7 < l_4 \Rightarrow l_8 \leq h_3$

$\Rightarrow \delta(h_2) = l_3$

$\Rightarrow h_6 \leq h_2$ and for all l_i with $i \in \{1, 4, 5, 6, 7, 8\}$: $l_i \leq h_3$. Consequently there is no possible value for $\delta(h_6)$.

The crucial constraints are the two constraints, introduced in the semantic representations of quantifiers, that express which of the two arguments of a quantifier is its restriction and which is its body. (The definition of a disambiguation mapping assures that nothing can subordinate two different holes of a term at the same time.) In other words, the fact that one reading is excluded, follows in a general way from the logical structure of quantifiers.

Note that the tree-locality of the grammar is important for the generation of adequate scope constraints. The scope bearing tree β_{some} must be adjoined to a node in α_{repr} because α_{some} is attached to α_{repr}. This leads to the constraint $l_6 \leq h_8$, i.e., to the constraint that representative-of(x_1, z) must be part of the body of the quantifying phrase *some*. Adjoining β_{some} to other trees, which would be possible in a non-local MCTAG, would lead to incorrect constraints. E.g., adjoining to the S node of the elementary tree for *saw* would lead to a constraint $l_1 \leq h_8$ instead of $l_6 \leq h_8$. However, any scope order of the three quantifiers would then be possible.

This shows that the locality of the grammar restricts the possible scope orders, i.e., it is crucial for obtaining the amount of underspecification that is adequate for scope ambiguities.

5.2. ISLAND CONSTRAINTS FOR SCOPE ORDER

Besides logical constraints there are also other restrictions on quantifier raising that seem to be very strict and that hold independently from the specific quantifiers. These are island constraints. Island constraints for quantifier raising are probably not exactly the same as island constraints for syntactical movement. But they seem to be parallel to a certain extent as noted, for example, in Fauconnier (1976) and Rodman (1976).

Relative clauses, in particular, are widely accepted to be islands for quantifier scope in the sense that a quantifier inside the relative clause cannot have scope

over the quantifier of the relativized NP (see Muskens, 1995; Muskens, 1998; Kallmeyer, 1999a).

(37) a. Every representative of most of the companies saw this sample.

b. Every person who represents most of the companies saw this sample.

There clearly is a contrast between (37)a. and (37)b. In (36)a. *most of the companies* can have wide scope, whereas in (37)b., wide scope of the embedded quantifier *most of the companies* is not possible. This can be explained by assuming that relative clauses constitute islands for quantifier scope.

Note that in some languages, particularly Scandinavian languages, relative clauses are not always islands for syntactic movement (see Engdahl, 1997). It still needs to be tested whether quantifiers extracted out of a relative clause (as it is possible in these languages) can have wide scope. However, even if they can have wide scope, this does not pose a problem for our analysis. In those cases where a quantifier is moved out of a relative clause, the syntactic analysis would already differ from the analysis of the relative clauses we are concerned with in this paper. But then it should be no problem to obtain a semantic analysis that allows for wide scope of the extracted quantifier. However, at this time, we will leave this issue for further research.

5.2.1. *Formalization of Island Constraints*

In most other approaches to underspecified semantics, island constraints are either not mentioned at all or they are explicitly stated as in Muskens (1995, 1998). Only in Kallmeyer (1999a, 1999b), do island constraints arise as consequences of more general properties of the grammar. We will partially follow these ideas in assuming that the difference between (37)a. and (37)b. follows from different kinds of derivations. The difference with respect to the derivation in TAG is that in (37)a., the elementary tree anchored by *representative* and *of* is an initial tree, whereas the tree for the relative clause with anchor *represents* in (37)b. is an auxiliary tree added by adjunction to the NP-tree of the quantifier *every*. This suggests that with respect to quantifier scope, auxiliary trees constitute an island, whereas initial trees do not. This observation coincides with the different dependencies we get in a derivation tree depending on whether a node is labelled by an initial or an auxiliary tree. In the dependency structure expressed by a derivation tree, auxiliary trees also constitute some kind of islands in the following sense: Suppose that the edges in a derivation tree are directed such that an edge goes from γ_1 to γ_2 iff γ_2 is an argument of γ_1. Then, as long as we have only initial trees, the edges will always be directed from the mother node to the daughters. But when a node has an auxiliary tree as its daughter, then the edge will go from the daughter to the mother. In this sense, when an auxiliary tree occurs, the chain of dependencies is interrupted and a new dependency tree begins. This is illustrated in the sample derivation tree in Figure 26.

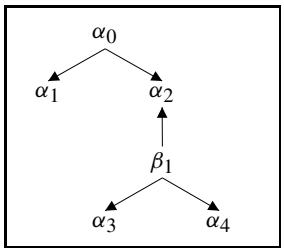


Figure 26. Sample directed derivation tree.

This suggests that the assumption that auxiliary trees create islands not only explains why relative clauses constitute islands but it really is a more general principle since it is a consequence of the semantic dependencies in a derivation structure.

In order to formalize this, we need the notion of the top of a semantic representation. Following Bos (1995), the top is defined as the topmost element with respect to subordination. In MRS (Copestake et al., 1999) the notion of a top is used in a similar way.

DEFINITION 5.1 (Top) Let σ be semantic representation with holes H_σ, propositional labels $L^\sigma_{\langle s,t\rangle}$ and free propositional or hole variables $V^\sigma_{\langle s,t\rangle}$. $top(\sigma) \in H_\sigma \cup L^\sigma_{\langle s,t\rangle} \cup V^\sigma_{\langle s,t\rangle}$ is called the top of σ iff for all $x \in H_\sigma \cup L^\sigma_{\langle s,t\rangle} \cup V^\sigma_{\langle s,t\rangle}$, $\langle x, top(\sigma)\rangle \in SUB_\sigma$ holds.

Now the additional constraints for quantifier scope arising from the derivation structure can be formulated in the following way:

DEFINITION 5.2 (Island constraint)
Let T be a derivation tree and $\beta \in A$. For all occurrences of β in T:

For all $\gamma_1, \ldots, \gamma_n \in I \cup A$ such that $\langle \beta, \gamma_1\rangle, \langle \gamma_1, \gamma_2\rangle, \ldots, \langle \gamma_{n-1}, \gamma_n\rangle$ are edges in T and $\sigma(\beta)$ and $\sigma(\gamma_n)$ both have a top, $top(\sigma(\gamma_n)) \leq top(\sigma(\beta))$ is an island constraint.

Here, $\sigma(\gamma)$ is a notation for the semantic representation of the specific occurrence of γ.

With these constraints, auxiliary trees block raising in the following way: everything that is added below an auxiliary tree β (by substitution or adjunction) is blocked by the top of $\sigma(\beta)$ (if β has a top at all), i.e. it cannot rise higher than β. However, it might still be the case that β itself can rise, there are no general constraints on the scope of the top of $\sigma(\beta)$. Whether β really constitutes an island or not depends on its specific semantic representation.[4]

This analysis predicts that relative clauses can constitute islands for quantifiers whereas argument NPs as *every representative of most of the companies* in

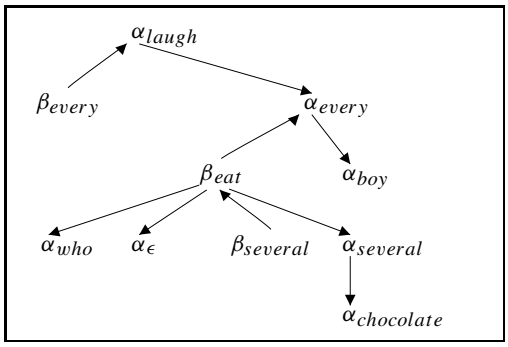


Figure 27. Derivation structure for (38).

(37)a. cannot. In other words, roughly said quantifiers can rise to higher trees in the derivation structure as long as there is a downwards predicate-argument dependency relation.

As an example consider (38).

(38) Every boy who eats several chocolates laughs.

Suppose that the derivation is done with the tree sets $\{\alpha_{laugh}\}$, $\{\beta_{every}, \alpha_{every}\}$, $\{\alpha_{boy}\}$, $\{\beta_{eat}\}$, $\{\alpha_{who}, \alpha_{\epsilon}\}$, $\{\beta_{several}, \alpha_{several}\}$, $\{\alpha_{chocolate}\}$. The tree β_{eat} is the tree for the anchor of the relative clause and it is adjoined to the NP tree α_{every} of *every boy*. Then the derivation structure for (38) is as shown in Figure 27.

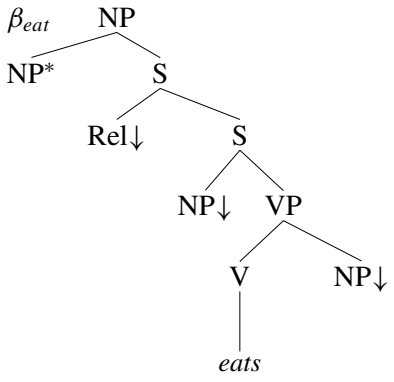
First, for *every boy laughs*, (39) is generated.

(39)

$l_2 : \mathsf{every}(x, h_2, h_3), l_3 : \mathsf{boy}(x), l_1 : \mathsf{laugh}(x)$ $l_3 \leq h_2, l_1 \leq h_3, l_1 \leq h_1$
arg: –

The relative clause must be adjoined to the NP tree of *every* because this gives us access to the variable corresponding to the NP.

(40)



$l_4 : \mathsf{eat}(p(x_1), x_2)$ $l_4 \leq h_4, h_4 \leq h$
arg: $x_1, h, \langle x_2, 01111\rangle, \langle p, 010\rangle$

The argument variable x_1 will be identified with the variable corresponding to the NP modified by the relative clause (in this case this is x) and x_2 is the second argument of eat.The predicate p will be contributed by the relative pronoun. In the case of (38) it is the identical mapping. But this is not always the case. If *who* in (38) was replaced by *whose friend*, p would be different.

The part that is crucial for making relative clauses act as islands is the constraint $h_4 \leq h$. The argument variable h will be identified with the hole in the semantic representation of the predicate argument part of the NP (in this case h_2, h_3 is part of the scope part of *every* and therefore not a possible value for h). This hole represents the restriction of the quantifier of this NP. In other words, if a relative clause is adjoined to a quantified NP, is must be in the scope of the restriction of the quantifier. h is the top of the semantic representation of the relative clause, i.e. everything added below the relative clause will be blocked by the value of h, namely by h_2.

(41)

$l_2 : \mathsf{every}(x, h_2, h_3), l_3 : \mathsf{boy}(x), l_1 : \mathsf{laugh}(x), l_4 : \mathsf{eat}(p(x), x_2)$ $l_3 \leq h_2, l_1 \leq h_3, l_1 \leq h_1, h_4 \leq h_2, l_4 \leq h_4$
arg: $\langle x_2, 01111\rangle$, $\langle p, 010\rangle$

(41) is the result of adding the relative clause to (39). h_2 now is an island for all quantifiers inside the relative clause.

(42)

α_{rel} Rel — *who*

$p_1 : \lambda x_1.x_1$
arg: –

α_{ϵ} NP — ϵ

arg: –

In the next step, the tree set in (42) is added to β_{eat}. The semantic assignment maps p to p_1. This means replacing $p(x)$ by x in (41). The resulting semantic representation is (43).

(43)

$l_2 : \mathsf{every}(x, h_2, h_3), l_3 : \mathsf{boy}(x), l_1 : \mathsf{laugh}(x), l_4 : \mathsf{eat}(x, x_2)$ $l_3 \leq h_2, l_1 \leq h_3, l_1 \leq h_1, h_4 \leq h_2, l_4 \leq h_4$
arg: $\langle x_2, 01111\rangle$

When adding the elementary tree and semantic representation for the quantifier inside the relative clause, its top must be below h_2 because this is the top of the elementary representation they are adjoined to. The result of the whole derivation is (44).

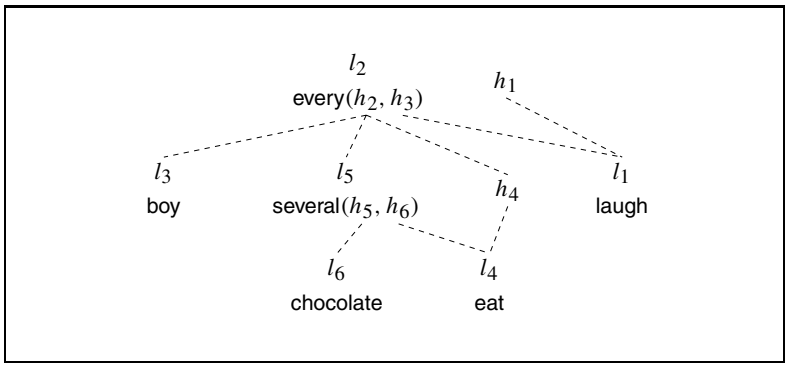


Figure 28. Scope constraints for (38).

(44)

l_2 : every(x, h_2, h_3), l_3 : boy(x), l_1 : laugh(x),
l_4 : eat(x, y), l_5 : several(y, h_5, h_6), l_6 : chocolate(y)
$l_3 \leq h_2, l_1 \leq h_3, l_1 \leq h_1, h_4 \leq h_2, l_4 \leq h_4, l_6 \leq h_5, l_4 \leq h_6, l_5 \leq h_2$
arg: –

The last constraint is an island constraint as defined in Def. 5.2. As a consequence of $l_5 \leq h_2$ the quantifier cannot raise out of the relative clause.

The picture in Figure 28 shows the scope constraints in (44).

The interpretation corresponds to the intersective interpretation of restrictive relative clauses: There is one possible disambiguation, namely

$$\delta : \begin{cases} h_1 \rightarrow l_2 \\ h_2 \rightarrow l_3 \\ h_3 \rightarrow l_1 \\ h_4 \rightarrow l_5 \\ h_5 \rightarrow l_6 \\ h_6 \rightarrow l_4 \end{cases}$$

With this disambiguation, the whole semantic representation (44) is true in some situation s iff every$(x, restr_x, body_x)$ is true in s, where

- $restr_x$ is true in s iff
  boy(x) is true in s and several$(y, restr_y, body_y)$ is true in s (with restriction chocolate(y) and body eat(x, y)), and
- $body_x$ is true in s iff laugh(x) is true in s.

In other words, the restriction of every is not only boy(x) but the conjunction of boy(x) and the interpretation of the relative clause with x as variable. This means that instead of the predicate boy, the predicate $\lambda x.$(boy$(x) \wedge$

$\mathsf{several}(y, \mathsf{chocolate}(y), \mathsf{eat}(x, y)))$ is considered, i.e. the intersection of boy and $\lambda x.\mathsf{several}(y, \mathsf{chocolate}(y), \mathsf{eat}(x, y))$ is built.

With the island constraints introduced in this section, the semantic representations of adverbial modifiers of the type of *usually* and *allegedly* in (11) must be slightly modified: Besides the argument variable s for the label of the modified proposition, a second argument variable h is needed for the hole of the modified proposition. This second argument makes sure that the top of the old proposition and the top of the modified proposition are the same.

(45)

$l_1 : \mathsf{usually}(h_1)$
$s \leq h_1, l_1 \leq h$
arg: h, s

(45) shows the revised elementary semantic representation for *usually*.

5.2.2. *Island Constraints and Logical Restrictions*

A more complex example involving logical restrictions and also island constraints is (46):

(46) Every man who thinks each girl loves some unicorn eats some fish.

(46) is taken from Muskens (1998). It has 12 readings because of the following two constraints:

- A logical constraint: *loves* is part of the restriction of *every* whereas *eats* belongs to the body of *every*. Therefore, if *every* has scope over *some fish*, then also *each girl* and *some unicorn* must outscope *some fish*.
- An island constraint: because of the relative clause, *each girl* and *some unicorn* both must be in the scope of *every*.

I.e., *some fish* either has wide scope or is in the scope of all the other quantifiers. In each of the cases, there are still 6 possibilities inside the relative clause: *each girl* can have scope over *thinks* or be inside the scope of *thinks*, and afterwards there are three possible scope positions for *some unicorn* (above, between or below *thinks* and *each girl*). Consequently there are $2 \cdot 2 \cdot 3 = 12$ readings.

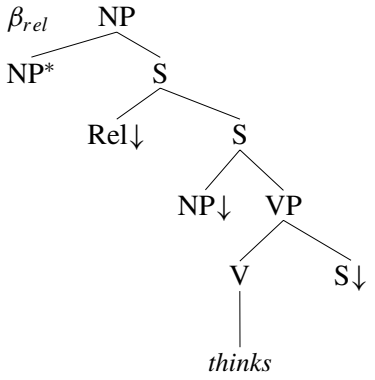
Going through the analysis of (46), we will show now how these island constraints together with the logical restrictions following from the structure of quantifiers give the desired 12 readings of (46). First, (47) is generated for *every man eats some fish*.

(47)

$l_2 : \mathsf{every}(x, h_2, h_3), l_4 : \mathsf{some}(y, h_4, h_5),$
$l_1 : \mathsf{eats}(x, y), l_3 : \mathsf{man}(x), l_5 : \mathsf{fish}(y)$
$l_3 \leq h_2, l_1 \leq h_3, l_5 \leq h_4, l_1 \leq h_5, l_1 \leq h_1$
arg: –

The elementary tree of the relative clause must be adjoined to the NP tree of *every*. For *thinks* as anchor of a relative clause as in (46), the elementary tree and semantic representation in (48) are adopted.

(48)



l_6 : think$(p(x_1), h')$ $h_6 \leq l_6, h_6 \leq h$
arg: $x_1, h, \langle p, 010\rangle, \langle h', 01111\rangle$

Here, x_1, h and p are the same kind of variables as in the relative clause in (38). The variable h' is linked to the substitution node of the complement clause of *thinks*, i.e. its value will be the hole of the complement clause.

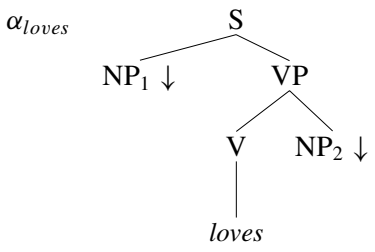
The top of this semantic representation is h. When adjoining this to the NP node of *every man*, i.e. when adjoining it to the initial tree in the tree set for *every*, h_2 will be assigned to h. Adjoining β_{rel} to the NP-node of *every man* and adding the trees and semantic representations shown in (42) for *who* generates the semantic representation (49).

(49)

l_2 : every$(x, h_2, h_3), l_4$: some(y, h_4, h_5), l_1 : eats$(x, y), l_3$: man$(x), l_5$: fish(y) l_6 : think(x, h') $l_3 \leq h_2, l_1 \leq h_3, l_5 \leq h_4, l_1 \leq h_5,$ $l_1 \leq h_1, l_6 \leq h_6, h_6 \leq h_2$
arg: $\langle h', 01111\rangle$

Then for *loves*, (50) is added to β_{rel} with h_7 assigned to h'. The resulting semantic representation is (51).

(50)



l_7 : loves(x_1, x_2) $l_7 \leq h_7$
arg: $\langle x_1, 00\rangle, \langle x_2, 011\rangle$

(51)

l_2 : every(x, h_2, h_3), l_4 : some(y, h_4, h_5),
l_1 : eats(x, y), l_3 : man(x), l_5 : fish(y)
l_6 : think(x, h_7), l_7 : loves(x_1, x_2)
$l_3 \leq h_2, l_1 \leq h_3, l_5 \leq h_4, l_1 \leq h_5, l_1 \leq h_1,$
$l_6 \leq h_6, h_6 \leq h_2, l_7 \leq h_7, h_7 \leq h_2$
arg: $\langle x_1, 00\rangle$, $\langle x_2, 011\rangle$

After having added the elementary trees and semantic representations for the two quantifiers inside the relative clause the semantic representation (52) is the result.

(52)

l_2 : every(x, h_2, h_3), l_4 : some(y, h_4, h_5),
l_1 : eats(x, y), l_3 : man(x), l_5 : fish(y)
l_3 : think(x, h_7), l_7 : loves(z, u)
l_8 : each(z, h_8, h_9), l_{10} : some(u, h_{10}, h_{11}),
l_9 : girl(z), l_{11} : unicorn(u)
$l_3 \leq h_2, l_1 \leq h_3, l_5 \leq h_4, l_1 \leq h_5, l_1 \leq h_1,$
$l_6 \leq h_6, h_6 \leq h_2, l_7 \leq h_7, h_7 \leq h_2$
$l_9 \leq h_8, l_7 \leq h_9, l_{11} \leq h_{10}, l_7 \leq h_{11}, l_8 \leq h_2, l_{10} \leq h_2$
arg: –

The last two constraints are island constraints. They cause the relative clause to be an island for the two quantifiers *each girl* and *some unicorn*. A picture of the scope constraints given by (52) is shown in Figure 29. The two kinds of constraints listed in the beginning of this section are correctly derived: The logical constraints $l_7 \leq h_2$ and $l_1 \leq h_3$ and the island constraints $l_8 \leq h_2$ and $l_{10} \leq h_2$.

Note that the two quantifiers inside the relative clause may have scope over *think*, i.e., inside the relative clause, *a unicorn* correctly has a de re as well as a de dicto reading. This is due to the fact that the elementary tree for *think* is an initial tree and therefore does not act as an island.

6. Conclusion

In this paper, we have presented a compositional semantics for LTAG based on the idea of factoring predicate argument and scope semantics. The principal characteristics of LTAG are the extended domain of locality of the elementary trees in a grammar and the factoring away of recursion that comes with the adjoining operation. These two properties allow the localization of the arguments of a lexical item

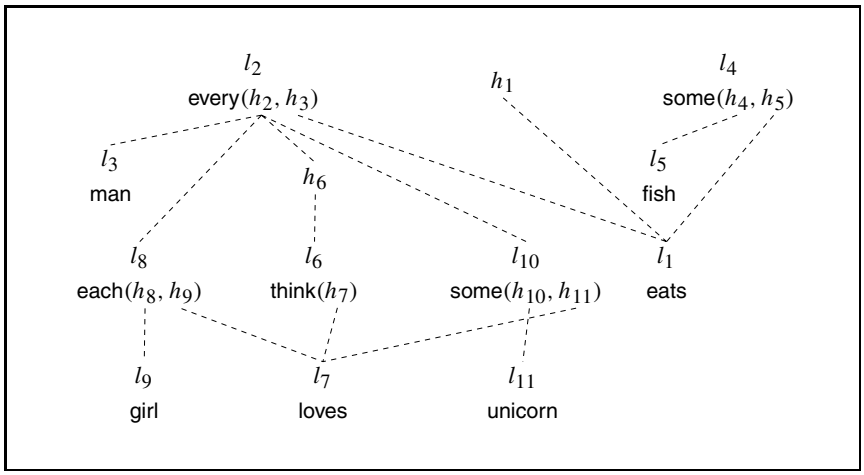


Figure 29. Scope constraints for (46).

within its elementary tree. Because of this localization of argument structures, it is appropriate to define a compositional semantics with respect to the derivation trees rather than the derived trees. In this system, each elementary tree is related to a semantic representation and the way these representations are combined depends only on the derivation structure. We have shown that the use of the derivation structure as interface between syntax and semantics leads to a more flexible relation (compared to traditional phrase structure based approaches) between syntax and semantics, which has advantages for the treatment of quantifiers.

One of the key ideas of our approach is to separate the contribution of a quantifier into a predicate argument part and a scope part. This leads to the use of tree-local MCTAG with at most two trees per elementary tree set. The choice of this extension of LTAG also arises out of an alternate perspective on adjoining, i.e., it constitutes a natural variant of LTAG. In order to deal with quantifiers, we allow multiple adjoinings in a very limited way. This does not influence the generative capacity of the formalism.

We have developed formal definitions of semantic representations, and formalized the way in which these representations combine with each oher depending on the specific derivation structure. Because of the extended domains of the elementary trees in LTAG, it was possible to abstract away from the specific internal structure of the syntactic trees and to use 'flat' semantic representations. In order to account for scope ambiguities, we enriched the semantic representations with metavariables and with a partial order on these variables and propositional labels. This partial order is interpreted as the scope order, and it enables the generation of adequate underspecified representations for scope ambiguities.

We investigated several scope phenomena, namely adjunct scope and quantifier scope, in particular restrictions on quantifier scope. We also showed that for

quantifier scope, the tree-locality of the grammar gives a way to obtain just the right amount of underspecification adequate for the analysis of scope ambiguities.

Acknowledgments

We are grateful to several people for fruitful discussions of the work presented in this paper and also for proof-reading, in particular to Tonia Bleam, Mark Dras, Robert Frank, Kim Gerdes, Chung-hye Han, Susan Heyner, Maribel Romero, Manfred Sailer and Anoop Sarkar. Furthermore, we would also like to thank two anonymous reviewers for valuable comments on the paper. This work was partially supported by NSF Grant SBR8920230 and by Deutsche Forschungsgemeinschaft DFG (SFB 441).

Notes

[1] The derivation trees of LTAG have a close relationship to the dependency trees, although there are some crucial differences, however, the semantic dependencies are the same.

[2] We should point out that this is not always the case. There are cases where the scope marking components of multi-component quantifier trees will attach to different trees, i.e., they will behave differently with respect to the derivation structure, although they show the same properties concerning scope. However, whenever the scope marking components behave the same way with respect to the derivation structure they have the same possibilities with respect to the scope.

Actually it is possible to develop a notion of flexible composition based on the observations that both substitution and adjoining can be thought of as operations of "attachments" and can be seen as going in either direction. We have seen this in the case of the alternate perspective on adjoining (Section 2.3). Such a notion of flexible composition was investigated in Joshi and Vijay-Shanker (1999). With this notion it is then possible to arrive at a derivation strucure such that the scope marking component from a lower clause rises above and attaches to the root node of the embedding clause along with another scope marking component in much the same way as in our previous example. However, we will not pursue this approach in this paper.

[3] The position of a node is its position in the elementary tree it belongs to. Positions do not refer to derived trees.

[4] If the top of β can rise arbitrarily high, β does not really represent an island. In this respect, the analysis presented here differs from the one in Kallmeyer and Joshi (1999) that seems to strong. In some cases, e.g. genitive NPs in German embedded into other NPs, quantifier raising out of these NPs (that are adjoined to other NPs) is possible. A similar case are quantifiers occurring in PP-adjuncts inside NPs. Our analysis allows the embedded quantifier in these cases to take wide scope.

References

Bos J. (1995) Predicate logic Unplugged. In Dekker P., Stokhof M. (eds.), *Proceedings of the 10th Amsterdam Colloquium*, pp. 133–142.

Bouma G., Malouf R., Sag, I. (1998) Adjunct Scope. Presented at the Workshop *Models of Underspecification and the Representation of Meaning*, 18–22 May, Bad Teinach.

Candito M.-H., Kahane S. (1998a) Can the TAG Derivation Tree Represent a Semantic Graph? an Answer in the Light of Meaning-Text Theory. In *Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks, IRCS Report 98-12*, University of Pennsylvania, Philadelphia, pp. 25–28.

Candito M.-H., Kahane S. (1998b) Defining DTG Derivations to Get Semantic Graphs. In *Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks, IRCS Report 98-12*, University of Pennsylvania, Philadelphia, pp. 25–28.

Copestake A., Flickinger D., Sag I. A. (1999) *Minimal Recursion Semantics. An Introduction*. Manuscript, Stanford University.

Engdahl, E. (1997) Relative Clause Extractions in Context. *Working Papers in Scandinavian Syntax*, 60, December.

Fauconnier G. (1976) *Etude de certains aspects logiques et grammaticaux de la quantification et de l'anaphore en français et en anglais*. PhD thesis, Université de Paris.

Hobbs J. R., Shieber S. M. (1987) An Algorithm for Generating Quantifier Scopings. *Computational Linguistics*, 13, pp. 47–63.

Joshi A. K. (1985) Tree Adjoining Grammars: How Much Context Sensitivity is Required ro Provide Reasonable Structural Descriptions? In Dowty D., Karttunen L., Zwicky A. (eds.), *Natural Language Parsing*, Cambridge University Press, pp. 206–250.

Joshi A. K. (1987) An Introduction to Tree Adjoining Grammars. In Manaster-Ramer A. (ed.), *Mathematics of Language*, John Benjamins, Amsterdam, pp. 87–114.

Joshi A. K. (1990) Processing Crossed and Nested Dependencies: An Automaton Perspective on the Psycholinguistic Results. *Language and Cognitive Processes*, 5, pp. 1–27.

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

Joshi A. K., Schabes Y. (1997) Tree-Adjoning Grammars. In Rozenberg G., Salomaa A. (eds.), *Handbook of Formal Languages*, Springer, Berlin, pp. 69–123.

Joshi A. K., Vijay-Shanker K. (1999) Compositional Semantics with Lexicalized Tree-Adjoining Grammar (LTAG): How Much Underspecification is Necessary? In Blunt H. C., Thijsse E. G. C. (eds.), *Proceedings ot the Third International Workshop on Computational Semantics (IWCS-3)*, Tilburg, pp. 131–145.

Kallmeyer L. (1999a) Synchronous Local TDGs and Scope Ambiguities. In Bouma G., Hinrichs E. W., Kruijff G-J., Oehrle R. T. (eds.), *Constraints and Resources in Natural Language Syntax and Semantics*, CSLI, pp. 245–262.

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

Kallmeyer L., Joshi A. K. (1999) Factoring Predicate Argument and Scope Semantics: Underspecified Semantics with LTAG. In Dekker P. (ed.), *12th Amsterdam Colloquium. Proceedings*, Amsterdam, December, pp. 169–174.

Kasper R. T. (1998) The Semantics of Recursive Modification. To appear in *Journal of Linguistics*, June.

Montague R. (1974) The Proper Treatment of Quantification in Ordinary English. In Thomason R. H. (ed.), *Formal Philosophy: Selected Papers of Richard Montague*, Yale University Press, New Haven, pp. 247–270.

Muskens R. (1995) Order-Independence and Underspecification. In Groenendijk J. (ed.), *Ellipsis, Underspecification, Events and More in Dynamic Semantics*. DYANA Report R2.2.C.

Muskens, R. (1998) Underspecified Semantics. Presented at the Workshop *Models of Underspecification and the Representation of Meaning*, 18–22 May, Bad Teinach.

Muskens R., Krahmer, E. (1998) Description Theory, LTAGs and Underspecified Semantics. In *Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks, IRCS Report 98-12*, University of Pennsylvania, Philadelphia.

Reyle U. (1993) Dealing with Ambiguities by Underspecification: Construction, Representation and Eduction. *Journal of Semantics*, 10, pp. 123–179.

Rodman R. (1976) Scope Phenomena, "Movement Transformations", and Relative Clauses. In Partee B. H. (ed.), *Montague Grammar*, Academic Press, pp. 165–176.

Schabes Y., Shieber S. M. (1994) An Alternative Conception of Tree-Adjoining Derivation. *Computational Linguistics*, 20(1), pp. 91–124, March.

Shieber S. M., Schabes Y. (1990) Synchronous Tree-Adjoining Grammars. In *Proceedings of COLING*, pp. 253–258.

The XTAG Research Group (1998) A Lexicalized Tree Adjoining Grammar for English. Technical Report 98-18, Institute for Research in Cognitive Science, Philadelphia.

Vijay-Shanker K. (1987) *A Study of Tree Adjoining Grammars*. PhD thesis, University of Pennsylvania.

Vijay-Shanker K., Joshi A. K. (1985) Some Computational Properties of Tree Adjoining Grammars. In *Proceedings of ACL*.

Vijay-Shanker K., Joshi A. K. (1988) Feature Structures Based Tree Adjoining Grammar. In *Proceedings of COLING*, Budapest.

Weir D. J. (1988) *Characterizing Mildly Context-Sensitive Grammar Formalisms*. PhD thesis, University of Pennsylvania.