GlueTag
Linear Logic based Semantics for LTAG
– and what it teaches us about LFG and LTAG –

Anette Frank

Language Technology Group
DFKI GmbH
Saarbrücken, Germany
frank@dfki.de

Josef van Genabith

Computer Applications
Dublin City University
Dublin, Ireland
josef@compapp.dcu.ie

Proceedings of the LFG01 Conference

University of Hong Kong, Hong Kong

Miriam Butt and Tracy Holloway King (Editors)

2001

CSLI Publications

http://csli-publications.stanford.edu

GlueTag *
Linear Logic based Semantics for LTAG
– and what it teaches us about LFG and LTAG –

Abstract

We review existing appoaches to semantics construction in LTAG (Lexicalised Tree Adjoining Grammar) based on the notion of *derivation (tree)s*. We argue that derivation structures in LTAG are not appropriate to guide semantic composition, due to a non-isomorphism, in LTAG, between the syntactic operation of adjunction on the one hand, and the semantic operations of complementation and modification, on the other.

Linear Logic based "glue semantics", as developed within the LFG framework (cf. Dalrymple (1999)), allows for flexible coupling of syntactic and semantic structure. We investigate application of glue semantics to LTAG syntax, using as underlying structure the *derived tree*, which is more appropriate for principle-based semantics construction. We show how Linear Logic based semantics construction helps to bridge the non-isomorphism between syntactic and semantic operations in LTAG. The glue approach captures non-tree local dependencies in control and modification structures, and extends to the treatment of scope ambiguity with quantified NPs and VP modifiers. Finally, glue semantics applies successfully to the adjunction-based analysis of long-distance dependencies in LTAG, which differs significantly from the f-structure based analysis in LFG.

1 Introduction

In this paper we review existing appoaches to semantics construction in LTAG (Lexicalised Tree Adjoining Grammar) based on the notion of *derivation (tree)s*. We argue that LTAG derivation trees are not appropriate to guide semantic composition, due to a non-isomorphism, in LTAG, between the syntactic operation of adjunction on the one hand, and the semantic operations of complementation and modification, on the other.

Linear Logic based "glue semantics", by now the classical approach to semantics construction within the LFG framework (cf. Dalrymple (1999)) allows for flexible coupling of syntactic and semantic structure. We investigate application of glue semantics to LTAG syntax,[1] using as underlying structure the *derived tree*, which seems more appropriate for principle-based semantics construction. We show how Linear Logic based semantics construction helps to bridge the non-isomorphism between syntactic and semantic operations in LTAG. Glue semantics captures non-tree local dependencies in control and modification structures, and extends to the treatment of scope ambiguity with quantified NPs and VP modifiers. Finally, glue semantics applies successfully to the adjunction-based analysis of long-distance dependencies in LTAG, which differs significantly from the f-structure based analysis in LFG in terms of functional uncertainty.

On a more general perspective, the exercise is instructive in that it elucidates the role that f-structure plays in LFG syntax and semantics, and helps clarify the similarities and differences between the two frameworks.

The paper is structured as follows. In Section 2 we review basic assumptions of the LFG and LTAG frameworks to set the stage for our investigations. Section 3 examines previous approaches to semantics construction in LTAG based on derivation (tree)s, namely Shieber and Schabes (1990), Schabes and Shieber (1994), Joshi and Vijay-Shanker (1999) and Kallmeyer and Joshi (1999). In

* We are grateful for valuable comments from the audiences of the LFG01 conference and the University of Konstanz, in particular Ron Kaplan, Josef Bayer and Ellen Brandner. Thanks go also to Dick Crouch and Mary Dalrymple for comments on earlier versions of this paper. Some interesting observations could not be given full justice in this paper, but provided important feedback for the overall conception of this work, which we hope to extend in future research. This research was partially funded by a BMBF grant to the DFKI project WHITEBOARD (FKZ: 01 IW 002).

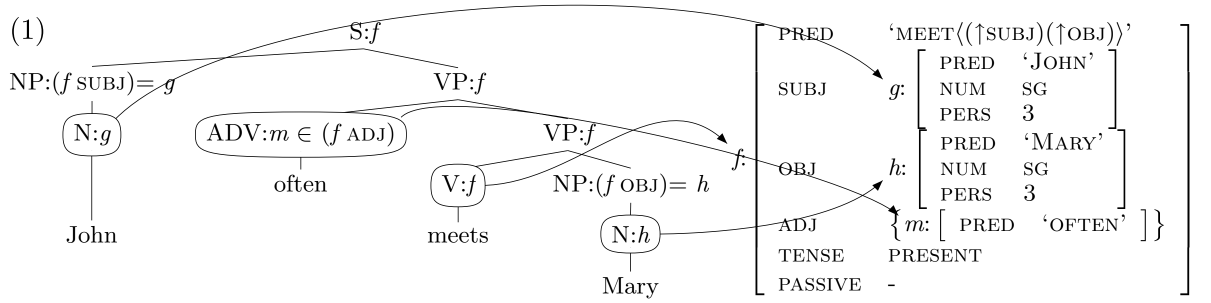
[1] Hepple (1999) sketches LL-based semantics for D-Trees, to overcome problems faced by categorial semantics in the analysis of quantification. Muskens (2001) develops a description-based syntax-semantics interface for LTAG, yet with extension to tree descriptions as used in D-Trees. We briefly discuss these related approaches in Section 4.7.

Section 4 we design LL-based semantics construction for LTAG on the basis of derived trees. In Section 4.1 we design labelling principles for LTAG elementary and derived trees as an interface to LL-based glue semantics. In Section 4.2 these principles are extended and refined to account for non-tree local dependencies and scope constraints, exemplified by modification structures. Section 4.3 summarises the specific assumptions introduced for glue semantics from LTAG derived trees. Section 4.4 shows that glue semantics successfully bridges the non-isomorphism between adjunction in syntax and corresponding operations in semantics. In Sections 4.5 and 4.6 we consider control and long-distance constructions which, in their syntactic analysis, differ considerably from the corresponding analyses in LFG. We show that LL-based semantics construction for LTAG straightforwardly extends to these more intricate cases. It is especially in the context of these constructions that differences and similarities between the two syntactic frameworks emerge most clearly. This is the topic of Section 5. Section 6 concludes.

2 Basic tenets of LFG and LTAG syntax and semantics

In LFG syntactic structure is represented in terms of two levels of syntactic description: c-structure and f-structure (1). Context-free PS rules with f-descriptions and lexical entries define the functional correspondence between c- and f-structure. Subcategorisation and long-distance dependencies are represented in f-structure, via functional descriptions. The correspondence function between c- and f-structure also accounts for word order variation. LFG semantics is driven by Linear Logic based meaning construction from f-structure, which allows for flexible coupling of syntax and compositional semantics. Lexical entries are associated with so-called meaning constructors. These consist of a "glue part", expressions in Linear Logic which refer to f-structure nodes, and a meaning part. The (instantiated) meaning constructors contributed by lexical entries are assembled as premises to a Linear Logic meaning derivation, based on the glue part. Following the Curry-Howard isomorphism, a meaning is computed, in parallel, on the meaning side. We assume familiarity with the glue semantics approach (see Dalrymple (1999), Dalrymple (2001) for more detail).

(1)



Lexical entries with associated meaning constructors

John	N	$(\uparrow \textsc{pred})=$ 'JOHN' $john : \uparrow_\sigma$
Mary	N	$(\uparrow \textsc{pred})=$ 'MARY' $mary : \uparrow_\sigma$
meets	V	$(\uparrow \textsc{pred})=$ 'MEET' $\lambda y,x.meet(x,y)$: $(\uparrow \textsc{obj})_\sigma \multimap ((\uparrow \textsc{subj})_\sigma \multimap \uparrow_\sigma)$
often	ADV	$(\uparrow \textsc{pred})=$ 'OFTEN' $\lambda P,x.often(P(x))$: $((\textsc{adj} \in \uparrow)\ \textsc{subj})_\sigma \multimap (\textsc{adj} \in \uparrow)) \multimap ((\textsc{adj} \in \uparrow)\ \textsc{subj})_\sigma \multimap (\textsc{adj} \in \uparrow))$

Instantiated meaning constructors

$john : g_\sigma$

$mary : h_\sigma$

$\lambda y,x.meet(x,y)$: $h_\sigma \multimap (g_\sigma \multimap f_\sigma)$

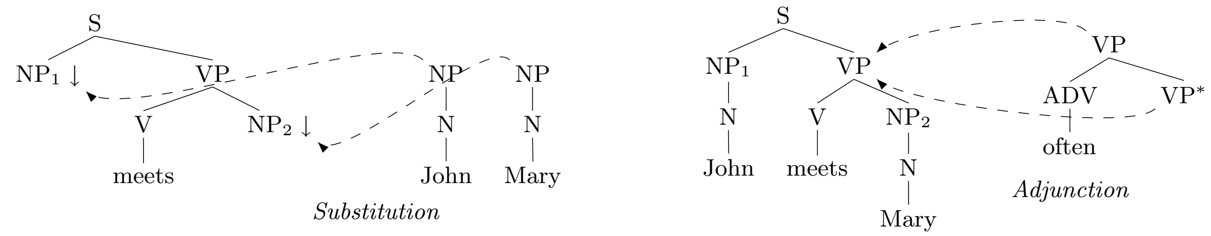
$\lambda P,x.often(P(x)) : (g_\sigma \multimap f_\sigma) \multimap (g_\sigma \multimap f_\sigma)$

Meaning derivation

$$\dfrac{\dfrac{\dfrac{\lambda y,x.meet(x,y) : h \multimap (g \multimap f) \qquad mary : h}{\lambda x.meet(x,mary) : g \multimap f} \qquad \lambda P,x.often(P(x)) : (g \multimap f) \multimap (g \multimap f)}{\lambda x.often(meet(x,mary)) : g \multimap f} \qquad john : g}{often(meet(john,mary)) : f}$$

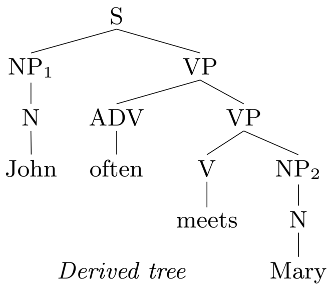
An LTAG grammar (Joshi 1987) consists of a set of lexicalised elementary trees (etrees), which are composed by two operations: substitution and adjunction (2). Elementary trees encode lexical syntactic properties: subcategorisation, agreement,[2] and syntactic variation in terms of tree families. The syntactic representation consists of the constituent tree (*derived tree*) built by substitution and adjunction of elementary trees (see (2), (3.a)), and a *derivation tree* (3.b), which records the dependencies between elementary trees as established by substitution and adjunction operations in parsing. LTAG semantics is traditionally based on the structure of *derivation trees*.

(2) Elementary trees, substitution and adjunction

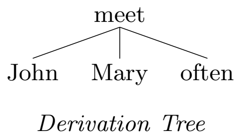


(3) Derived trees and derivation trees

(3.a)



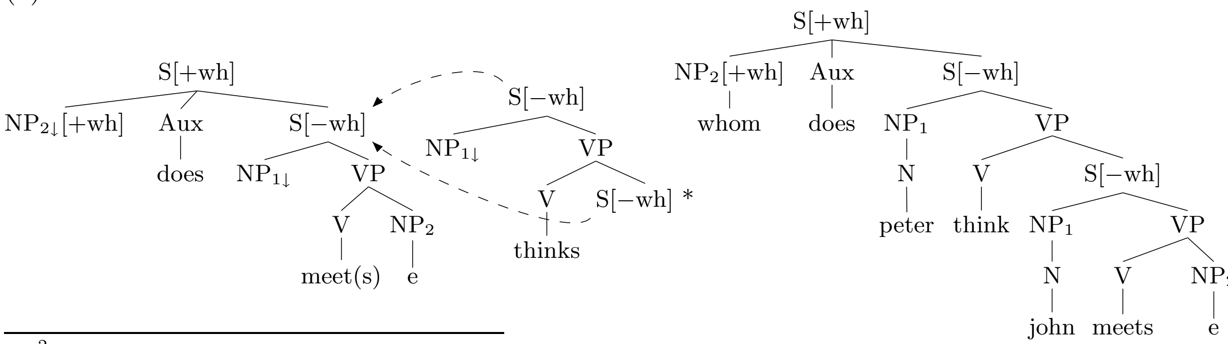
(3.b)



Extended Domains of Locality and Adjunction A central feature of LTAG syntax is *strict lexicalisation* combined with the concept of *Extended Domains of Locality*. Besides being *strictly lexicalised*, elementary trees *localise* all subcategorised arguments of the lexical head, representing them as substitution or foot nodes of the elementary tree. The joint assumptions of strict lexicalisation and localisation of arguments lead to *adjunction* as a major operation in syntactic composition. This is already evident in (2). Due to localisation of subject and object NPs in the etree *meets*, the derived tree for *John often meets Mary* can only be obtained by "folding in" the auxiliary tree for *often* into the etree of *meets* by use of the adjunction operation.

Besides optional (or recursive) modification structures, as in (2), localisation of arguments plays a central role in the analysis of long distance dependencies. In (4) the etree for *meets* locally encodes a fronted object wh-phrase. In order to derive sentence (4), the etree for *thinks* must again be "folded into" the etree of *meets* via adjunction. Thus, joint with the concept of localisation of arguments in strictly lexicalised elementary trees, the operation of adjunction naturally leads to the concept of *extended domains of locality*.

(4) *Whom does Peter think John meets?*



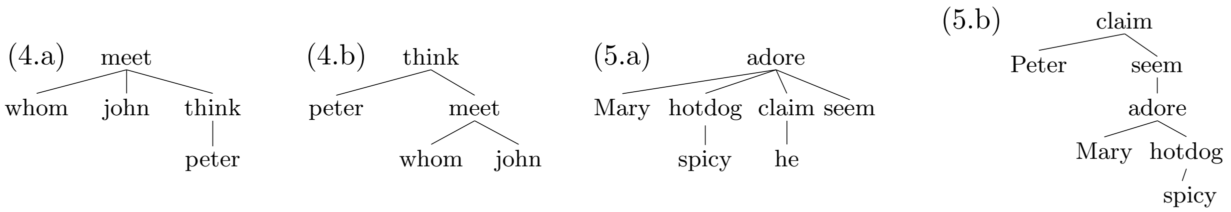
[2]Subject to feature constraints on nodes, not shown here.

Derivation trees are not dependency trees Derivation trees record the relations between elementary trees as established by substitution and adjunction operations in tree composition, and are traditionally used as the basis for semantics construction in LTAG.

Derivation trees do not in general correspond to well-formed dependency structures. This was observed by Rambow et al. (1995), and is illustrated below. In the derivation tree (4.a) for sentence (4) the dependence of *think* upon *meet* is in fact inverted, as evidenced by the correct dependencies displayed in (4.b). One could argue that dependencies established by adjunction could be specially marked to account for such inverted dependencies, but more complex cases prove that this cannot, in general, lead to a well-formed dependency tree. (5.a) displays the derivation tree for (5).[3] Since *claim* and *seem* independently adjoin to the S and VP nodes of the etree *adore*, the derivation tree cannot represent the dependence of *seem* upon *claim*, as given in the correct dependency tree (5.b).

Note further that due to the principle of localisation of arguments, the operation of adjunction applies both to modifiers in (3) and to complementation structures such as sentence embedding verbs in (4).[4] As we shall see, this constitutes an additional complication for principle-based semantics construction from derivation trees.

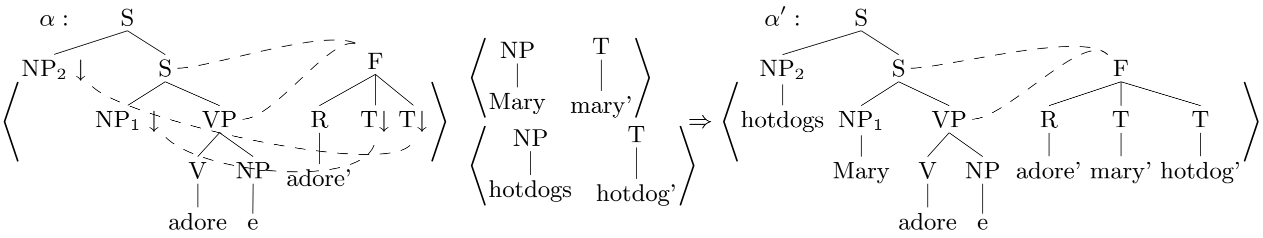
(4) Whom does Peter thinks John meets? (5) Spicy hotdogs he claims Mary seems to adore.



3 Semantics in LTAG

3.1 Shieber and Schabes: Semantics construction with Synchronous TAG

Shieber and Schabes (1990) associate LTAG syntax with a semantic representation in a synchronous TAG extension, where the grammar components are pairs of syntactic and semantic trees. The semantic representation, a tree-like logical form, is built in parallel with the syntactic derivation, making use of a specification of links between nodes in the paired tree components. On substitution of a tree t_1 into a substitution node n_1 in the syntactic tree, a parallel substitution takes place of the paired semantic tree t_2 into the node n_2 that n_1 is linked to. After substitution, the link being used is removed. This is illustrated for substitution of the trees for *Mary* and *hotdogs* into α below.

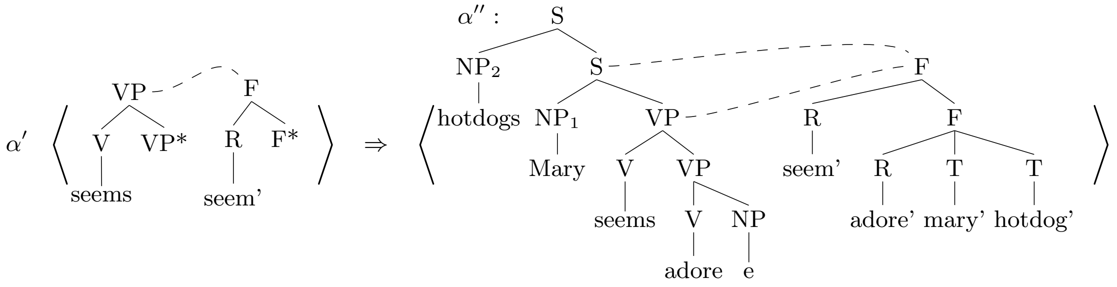


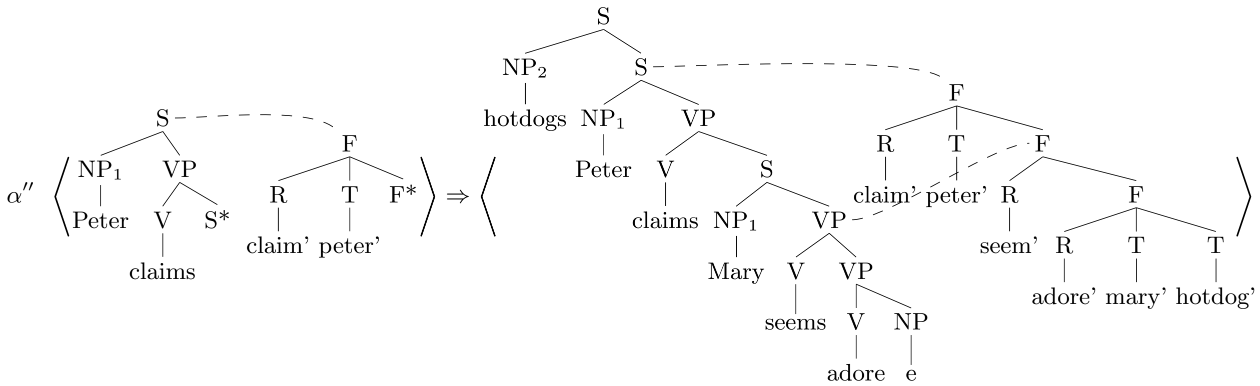
Crucial in this approach is the specification of links between paired syntactic and semantic trees, since they determine the attachment sites for the parallel semantic operations. Note in particular the link between the *tree internal* S and VP nodes of *adore* to the single root of the associated semantic tree. It is due to these links that the more complex cases of non-isomorphic derivations vs. dependencies as in (5) can be accounted for. The syntactic tree for *seems* can adjoin to the tree internal VP node of α', triggering a corresponding operation on the linked root node F in the paired semantic tree, which results in the correct scoping of *seem'* over *adore'*. Second, the syntactic tree

[3] The example is slightly changed from Rambow et al. (1995).

[4] As well as control constructions, see below.

for *claim* adjoins to the internal S node of the resulting syntactic tree in α'', leading to adjunction of its associated semantic tree to the root formula F with relation *seem*. The reader may verify that the same result is obtained for alternative derivations with *claim* being adjoined before *seem*.





However, as pointed out in Shieber and Schabes (1990), the order of derivations *can* have an effect on the semantic representation in that *different orders* of substitution of quantified NPs yields alternative scopings. Parsing must therefore compute all possible syntactic derivation histories, explicitly or implicitly (cf. Schabes and Shieber (1994)) in order to capture ambiguities that are essentially semantic. This is not only problematic conceptually, but in fact leads to spurious analyses in cases like (5) where the order of derivations is not distinctive for semantic interpretation.[5] Finally – in view of the following discussion – it is important to note that in this approach it is the linking of tree internal nodes in (elementary and derived) trees that accounts for cases of non dependency-like derivations, as in (5). This, however, characterises the approach as a hybrid one, in that semantics construction is essentially built on the structure of *derived trees*, while accounting for scope ambiguities in terms of *derivation histories* as determined by syntactic analysis.

3.2 Compositional Semantics from Derivation Trees

Joshi and Vijay-Shanker (1999) propose compositional semantics construction from derivation trees, focussing primarily on predicate-argument relations. Elementary trees are associated with tripartite semantic representations. The first part specifies the main variable of the predication, the second part states the predicate with argument variables, the third part associates variables with argument nodes in the elementary trees.[6] [7]

[5]Schabes and Shieber (1994) distinguish *modifier-type* from *predicative* auxiliary trees such as sentence embedding verbs (*say, claim*), where the foot node corresponds to an argument of the anchor. In contrast to the generally assumed notion of *standard derivations*, which excludes multiple adjunction to single nodes, Schabes and Shieber (1994) propose the notion of *extended derivations*, licensing multiple adjunction to single nodes. Allowing *extended derivations* for multiple adjunction of *modifier-type* auxiliary trees can yield alternative scopes in semantics construction, due to alternative derivation histories. For *predicative* auxiliary trees, however, *standard derivation* – i.e. the constraint against multiple adjunction at single nodes – is preserved. Scoping ambiguities are therefore correctly prohibited with cascaded sentence embeddings driven by adjunction.

[6]This association is not made explicit, but could be formalised by stating pairs of variables and the node addresses of the corresponding arguments in the elementary tree.

[7]For reasons of space we can only illustrate some selected entries (see continuation next page)

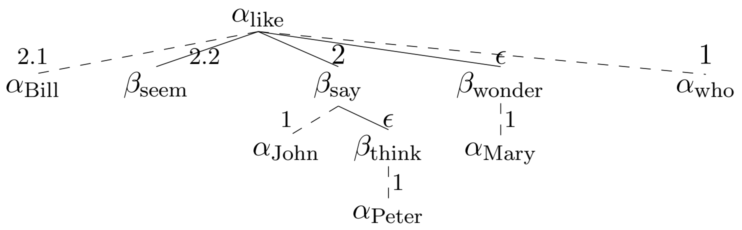
For substitution of NP arguments the binding of variables in the associated semantic representations is straightforward. For adjunction of predicative auxiliary trees (*say, think, wonder*), however, the derivation structure does not mirror semantic embedding: while in (8) *think* takes scope over *say*, this dependency is inverted in the derivation tree (8.a). Joshi and Vijay-Shanker (1999) propose a special treatment for predicative auxiliary trees, in that the adjunction node is basically processed as if it were a substitution node, during semantic composition. This allows for correct embedding of *say* by *think*, as well as *like* by *wonder* in the semantic representation for (8).

Yet, as in (5), multiple adjunction of predicative auxiliary trees to *distinct nodes* of a single elementary tree (here *wonder* and *seem* into *like*) leads to additional complication. While the relative scope of *think* and *say* can be correctly determined – in that the trees stand in a direct adjunction relation – the relative scope of *wonder* and *seem* cannot be determined through variable bindings at adjunction nodes: They adjoin to distinct nodes in the verb's elementary tree. In order to derive the correct relative scopes in such configurations Joshi and Vijay-Shanker (1999) impose an ordering constraint for processing multiple (predicative) adjunctions into etrees, in a bottom-up manner: since *seem* adjoins to a lower node in the etree of *like* than *wonder* (node 2.2 vs. ϵ in (8.a)), it is processed first in semantic composition, thereby taking narrow scope relative to *wonder*.

It seems conceptually problematic to resort to specially designed ordering constraints for the traversal of derivation trees in semantic composition, especially in view of language specific constituent structure properties which might not correspond to the structure of semantic composition. The analysis does also not explicitly deal with scope ambiguities induced by NP quantification.

(8) Mary wondered who Peter thought John said Bill seemed to like.

(8.a)



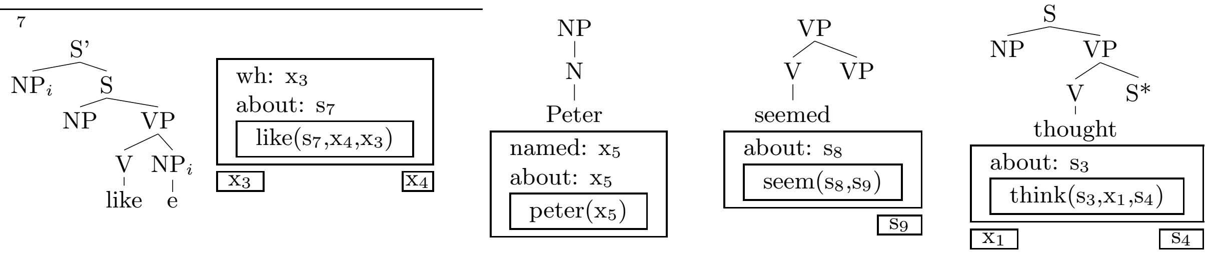
Kallmeyer and Joshi: Underspecified Semantics with MRS Kallmeyer and Joshi (1999) develop an account for underspecified semantics construction from LTAG derivation trees using MRS semantics. Very close to the architecture proposed in Joshi and Vijay-Shanker (1999) they associate flat semantic representations with elementary trees, now adopting the framework of Minimal Recursion Semantics to deal with scope underspecification. Semantic composition is again determined by the structure of derivation trees. The paper provides an underspecified analysis of quantification which accounts for quantifier scope ambiguities. It then focusses on examples of adjunct scope as in (9), where – given the assumption of *standard derivation* – *allegedly* must adjoin to *usually*, and is therefore restricted to take wide scope. Yet, given the assumption of *standard derivations*, scope ambiguities as in (10) can only be derived in terms of distinct derivations, i.e. *distinct* derivation trees. It is not clear in which way a *single underspecified* representation can be constructed from distinct derivation trees.

(9) Pat allegedly usually drives a cadillac.

(10) John intentionally knocked twice.

---

[7]



The paper is not really explicit about the distinction between semantic composition operations for adjunction versus substitution, in particular concerning the distinction between modifier and predicative auxiliary trees. We suppose that cascaded sentence embeddings can be handled along the lines of Joshi and Vijay-Shanker (1999)'s approach, by special conditions for variable binding on adjunction of predicative auxiliary trees. However, we see similar problems, in Kallmeyer and Joshi's account, to determine the correct embedding structure for multiple auxiliary trees adjoining to *distinct nodes* in a single elementary tree, as discussed for examples (5) and (8) above.

3.3 Discussion

We detailed the characteristics of LTAG syntax, in particular the structure of derivation trees, and the complexities that arise for semantics construction on the basis of LTAG derivation trees. We conclude that LTAG derivation trees do not provide an appropriate structure for principle-based semantics construction. The non-isomorphism between adjunction in syntax and modification in semantics introduces considerable complexity in semantics construction from derivation trees. The principle of extended domains of locality, in conjunction with the adjunction operation, yields semantically inappropriate dependencies in *derivation trees*, as these are imposed by purely syntactic operations in the composition of strictly lexicalised elementary trees. We therefore set out to investigate semantics construction in LTAG on the basis of the *derived tree*, which we consider more appropriate to guide principle-based meaning composition. We apply the framework of Linear Logic based glue semantics, which allows for considerable flexibility in the coupling of syntactic and semantic structure, while still remaining compositional in meaning construction.

4 Glue Semantics for LTAG

In applying glue semantics to LTAG we (i) define semantics on the basis of *derived* trees, which seems more appropriate for principle-based semantics construction. (ii), the loose coupling of syntactic and semantic structures with glue allows us to bridge the gap imposed by the aforementioned non-isomorphism in LTAG. (iii), we show that the glue approach captures non-tree local dependencies in modifier and control constructions. Finally, (iv) we propose a glue-based analysis of long-distance constructions, which in LTAG are driven by tree adjunction – as opposed to the f-structure based analysis in LFG with functional uncertainty.

4.1 Meaning constructors for LTAG initial elementary trees

To drive LL-based semantics construction, we need to associate meaning constructors with elementary trees, the lexical units of an LTAG grammar. As an interface to glue semantics we define principles for labelling nodes in elementary and derived trees with variables. These variables are referred to in the glue part of the associated lexical meaning constructors, and guide meaning composition.

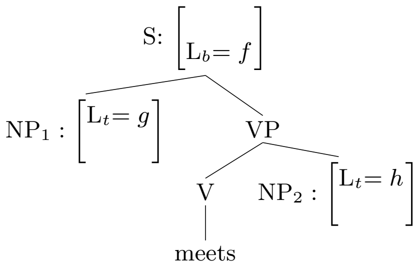
Tree Labelling Principle I, to be stepwise refined along the way, labels argument and root nodes in LTAG initial trees with atomic features L_t and L_b, which we will call *upper and lower labels*:

Tree Labelling Principle I
Assign variables $f, g, h \in VAR$ to top/bottom labels $\mathrm{L}_t/\mathrm{L}_b$ of nodes n in LTAG initial etrees α

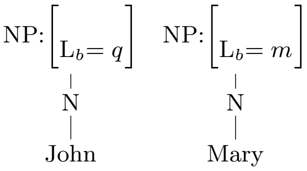
- Root nodes $\mathrm{root}(\alpha)$: $\mathrm{L}_b(\mathrm{root}(\alpha)) = x$, x a new variable from VAR
- Argument nodes $\mathrm{arg}_i(\alpha)$: $\mathrm{L}_t(\mathrm{arg}_i(\alpha)) = x$, x a new variable from VAR

(11.a) displays sample etrees with associated meaning constructors. For *John meets Mary* we obtain the labelled derived tree (11.b). On substitution, feature bundles on substitution nodes and inserted root nodes are unioned, the resulting nodes display both upper and lower labels L_t and L_b.

(11.a) Labelled elementary trees with associated meaning constructors

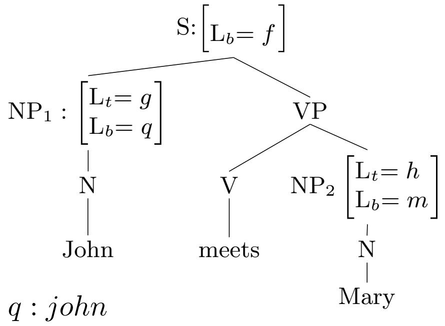


$h \multimap (g \multimap f) : \lambda y, x.meet(x, y)$



$q : john$ $\quad$ $m : mary$

(11.b) Derived tree with assembled meaning constructors



$q : john$
$m : mary$
$h \multimap (g \multimap f) : \lambda y, x.meet(x, y)$

The meaning constructors of the etrees used in tree composition are assembled (11.b), but in their present form do not yield a successful proof in meaning derivation, since the variables in the glue parts are not connected. The missing equalities between variables are determined by tree composition, along the following lines.

Variable Equations in Tree Composition

- **Substitution:** when substituting β into α at node n_α, add equation: $\mathrm{L}_t(n_\alpha) = \mathrm{L}_b(\mathrm{root}(\beta))$
- **Adjunction:** see below

Using this information about variable equations, we could either resolve the equation system globally, or else trigger systematic variable substitutions in the set of assembled meaning constructors. We choose the latter option here, by adopting the following convention: for all substitution nodes n, and all assembled meaning constructors mcs, we replace (all occurrences of) the lower label variable $\mathrm{L}_b(n) = t$ by n's upper label variable $\mathrm{L}_t(n) = t'$. In (11.b), this triggers the substitutions $q \rightarrow g$ and $m \rightarrow h$.

Variable Substitution in (glue part of) meaning constructors *mcs*

- $\forall n \forall mcs.((L_b(n) = t \wedge L_t(n) = t' \wedge mcs[t]) \rightarrow mcs[t \rightarrow t'])$

LL-derivations for meaning construction With this in place we obtain the set of meaning constructors in (12) which yields a successful proof of the meaning associated with the tree's root variable, based on the Curry-Howard isomorphism.

(12) $g : john$
$h : mary$
$h \multimap (g \multimap f) : \lambda y, x.meet(x, y)$

$$\frac{g : john \qquad \dfrac{h \multimap (g \multimap f) : \lambda y, x.meet(x, y) \quad h : mary}{g \multimap f : \lambda x.meet(x, mary)}}{f : meet(john, mary)}$$

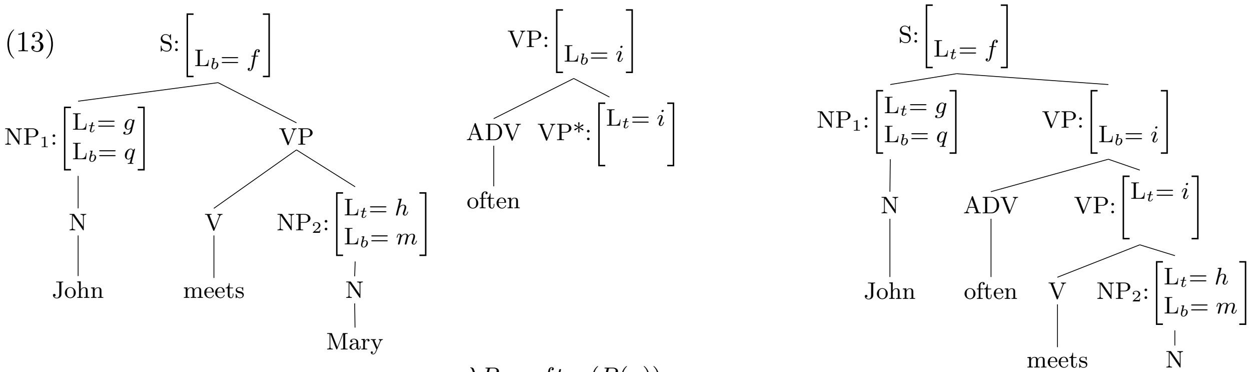
4.2 Non-tree local dependencies I: VP modification

Up to now we were only looking at constructions involving tree-local dependencies identified by argument substitution nodes. We now turn to constructions involving non-tree local dependencies, i.e. dependencies which are not identified by tree-local nodes in elementary trees. Constructions that fall into this class are modifiers and control constructions.

Let us first consider a VP modifying adverb like *often*. In LTAG, it is represented as an auxiliary tree that adjoins to VP, as displayed in (2). We extend our tree labelling principle to *modifier-type* auxiliary trees, assigning identical bottom and top labels to root and foot nodes: $\mathrm{L}_b(\mathrm{root}(\beta)) = \mathrm{L}_t(\mathrm{foot}(\beta)) = x$, x a fresh variable from the set of variables VAR.

For *often*, we obtain a labelled tree β as in (13). The meaning constructor for *often*, as a VP modifying adverbial, should consume and produce a VP meaning, which is characterised by consuming the subject's glue variable $\mathrm{L}_t(\mathrm{NP}_{1\alpha})$ of the tree α that β adjoins to, to produce the glue variable $\mathrm{L}_b(root(\alpha))$ of α's root node, as sketched below. But neither of these is local to the auxiliary tree β (*often*), and can therefore not be referred to in its associated meaning constructor.

(13)



$h \multimap (g \multimap f) : \lambda y, x.meet(x, y)$

$\lambda P, x.often(P(x))$:
$(\mathrm{L}_t(\mathrm{NP}_{1\alpha}) \multimap \mathrm{L}_b(root(\alpha))) \multimap (\mathrm{L}_t(\mathrm{NP}_{1\alpha}) \multimap \mathrm{L}_b(root(\alpha)))$

4.2.1 Labelling tree internal nodes: head projections

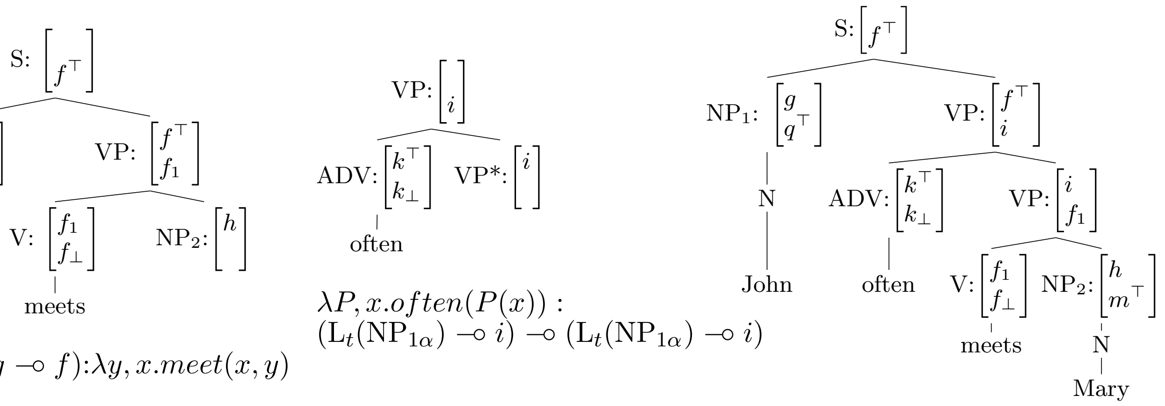
To capture such non-tree local dependencies, we revise our Tree Labelling Principle in two ways: (i) instead of root nodes, we label the lexical anchor (head) node with some variable $x_\perp$. (ii) this anchor node label is projected to all non-labelled tree internal nodes, introducing a chain of variables with intermediate projection labels $x_\perp, x_1, x_2, \ldots, x^\top$, as seen in (14).[8] Yet, given LTAG's concept of lexicalised elementary trees, with direct encoding of subcategorised argments, we will keep these projection variables distinct, triggering variable substitutions only locally, i.e. at local adjunction nodes, as opposed to unification of f-structure nodes in head projection chains in LFG.

Tree Labelling Principle II

- Anchor nodes: $\mathrm{L}_b(anchor(\alpha)) = x_\perp$, x new variable from VAR
- Argument nodes $\mathrm{arg}_N(\alpha)$: $\mathrm{L}_t(\mathrm{arg}_N(\alpha)) = x$, x new variable from VAR
- Modifier-type auxiliary trees β: $\mathrm{L}_b(\mathrm{root}(\beta)) = \mathrm{L}_t(\mathrm{foot}(\beta)) = x$, x new variable from VAR
- Projecting anchor node variable to all non-labelled tree internal nodes, introducing intermediate projection variables $\mathrm{x}_\perp$ x_1 x_2 ... $\mathrm{x}^\top$

On adjunction of some auxiliary tree β to a node n in α, n's label features are split: the top label feature L_t of n is assigned as the top label feature L_t of $root(\beta)$, and the bottom label feature L_b of n is assigned as the bottom label feature L_b of β's foot node. We obtain the derived tree in (14).

(14)



$\lambda P, x.often(P(x))$:
$(\mathrm{L}_t(\mathrm{NP}_{1\alpha}) \multimap i) \multimap (\mathrm{L}_t(\mathrm{NP}_{1\alpha}) \multimap i)$

$h \multimap (g \multimap f)$:$\lambda y, x.meet(x, y)$

[8] In the following we omit the feature names L_t and L_b, to avoid confusion with the upper and lower bounds of projected labels $\mathrm{x}_\perp$ x_1 x_2 ... $\mathrm{x}^\top$.

We now extend the conditions for equating variables in tree composition to the case of adjunction, and generalise the conditions for variable substitutions in meaning constructors to account for both substitution and adjunction. With these extensions and the labelling of tree internal nodes by projected anchor variables, the variable i in the meaning constructor for *often* will – after variable substitution – successfully refer to the sentence's root node variable $f^{\top}$.[9]

Variable Equations in Tree Composition
- **Substitution:** when substituting β into α at node n_α, add equation: $\mathrm{L}_t(n_\alpha)$=$\mathrm{L}_b(\mathrm{root}(\beta))$
- **Adjunction:** when adjoining β to α at node n_α, add equation: $\mathrm{L}_t(n_\alpha) = \mathrm{L}_b(\mathrm{root}(\beta))$

Variable Substitutions in meaning constructors mcs **using set of equations** EQ **(final)**
- $\forall n$ (($\mathrm{L}_t(n) = x$ and $x = y \in EQ) \rightarrow \forall mcs : mcs[y \rightarrow x]$)

4.2.2 Labelling arguments as arguments of lexical heads

With these changes it is still not possible to refer to the non-tree local variable for the subject $\mathrm{L}_t(\mathrm{NP}_{1\alpha})$ in the meaning constructor of *often*. We therefore further revise Tree Labelling Principle (II) by encoding argument nodes in elementary trees as *arguments of their lexical head*, using the local tree's anchor label and a grammatical function identifier. Rather than using identifiers like NP_1, NP_2, etc., which in LTAG encode grammatical functions, we make use of grammatical function labels *subj*, *obj*, *comp*, etc., similar to those used in LFG.[10] So, if $f_\perp$ is the label of the lexical anchor, the subject node NP_1 will be labelled $f_\perp$:subj, the object node NP_2 $f_\perp$:obj.

As a result, the labelled trees look more LFG-like, but do not introduce additional linguistic assumptions into LTAG: Note that indices on NPs, such as NP_1, NP_2 do in fact encode grammatical functions in LTAG. This is brought out by looking at pairs of trees in the passive relation change.[11] Lexical heads are identified as primary anchors of elementary lexicalised trees, and head projection lines from these anchors emerge naturally as the complement of the set of nodes which are marked as argument or paired root/foot nodes of the local tree. Finally, LTAG's principle of extended domains of locality requires all arguments of a lexical item to be encoded within its elementary tree. Encoding arguments as *arguments of their head* is thus in line with LTAG's basic assumptions.

Tree Labelling Principle III (clause (ii)) (revised from (II))
- Argument nodes: $\mathrm{L}_t(\mathrm{arg}_N(\alpha)) = \mathrm{L}_b(\mathrm{anchor}(\alpha))$:$\mathrm{GF}_N$,
  where GF_N is the grammatical function corresponding to arg_N

Elementary trees and meaning constructors for *John often sees Mary* are now revised according to the new conventions (see (15)). The meaning constructor for *often* refers to the non-tree local subject NP variable in terms of the local variable i: (i:subj $\multimap i$) $\multimap$ (i:subj $\multimap i$).

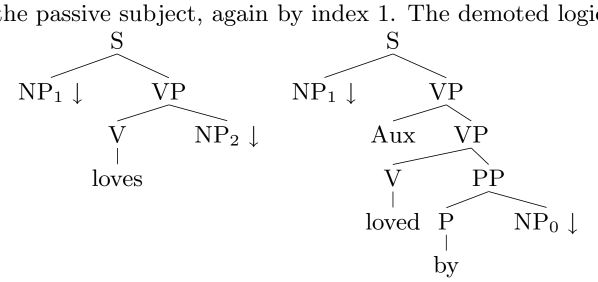
In tree composition we establish variable equations, which trigger variable substitutions in the assembled set of meaning constructors (16.a). The resulting set of (resolved) meaning constructors is (16.b). The glue formula contributed by *often* does now refer to the verb's subject node label as $f^{\top}$:subj and to the root node label as $f^{\top}$.

---

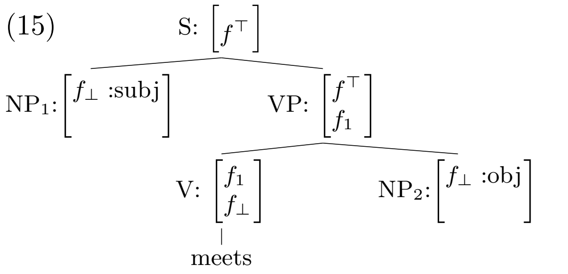
[9]We will display the analysis in (15) and (16), after a further revision to our Tree Labelling Principle.

[10]See also extensions in FTAG, which makes use of grammatical function features like SUBJ, OBJ, etc.

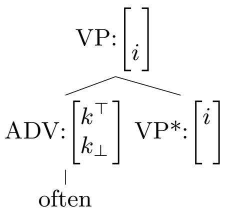
[11]The subject NP in the active tree is marked NP_1, the object NP_2. In the passive, the logical object is marked as the passive subject, again by index 1. The demoted logical subject, if present, is labelled NP_0.



(15)



$f_\perp$:obj $\multimap$ ($f_\perp$:subj $\multimap$ $f^\top$) : $\lambda y, x.meet(x, y)$

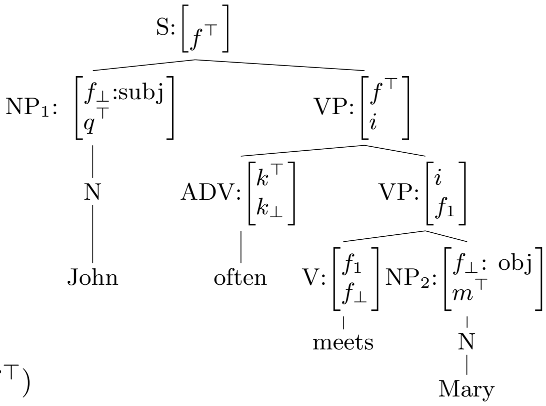


$\lambda P, x.often(P(x))$:(i:subj $\multimap$ i) $\multimap$ (i:subj $\multimap$ i)

(16.a)
$john$: $q^\top$
$mary$: $m^\top$
$\lambda y, x.meet(x, y)$: $f_\perp$:obj $\multimap$ ($f_\perp$:subj $\multimap$ $f^\top$)
$\lambda P, x.often(P(x))$: (i :subj $\multimap$ i) $\multimap$ (i :subj $\multimap$ i)

Substitutions: $q^\top \rightarrow f_\perp$:subj, $m^\top \rightarrow f_\perp$:obj, $i \rightarrow f^\top$

(16.b)
$john$: $f_\perp$:subj
$mary$: $f_\perp$:obj
$\lambda y, x.meet(x, y)$: $f_\perp$:obj $\multimap$ ($f_\perp$:subj $\multimap$ $f^\top$)
$\lambda P, x.often(P(x))$: ($f^\top$:subj $\multimap$ $f^\top$) $\multimap$ ($f^\top$:subj $\multimap$ $f^\top$)



However, the set of meaning premises (16.b) does not yield a successful meaning derivation. The resource $f_\perp$:obj (from *Mary*) can be consumed by $f_\perp$:obj $\multimap$ ($f_\perp$:subj $\multimap$ $f^\top$), to produce the resource $f_\perp$:subj $\multimap$ $f^\top$ corresponding to the VP meaning $\lambda x.meet(x, mary)$. But this latter resource cannot be consumed by the meaning constructor for *often*, which expects a VP constructor ($f^\top$:subj$\multimap f^\top$). This latter glue formula resulted from variable substitutions in the meaning constructor of *often*, which can only refer to its local variable i in root and foot node. In tree composition we established equality of this variable with the connecting adjunction node's top label $f^\top$.

4.2.3 Labelling arguments as arguments of local head projection

The LFG instructed reader will now suggest that we give up the distinction between head projection variables $x^\top \ldots x_\perp$, by triggering global variable substitutions on head projection labels. However, we want to introduce as little additional assumptions in LTAG-based semantics construction as needed. We will show that we can continue to restrict ourselves to *local* variable substitutions at adjunction nodes, by a weaker extension of our labelling principle, which will encode the attachment of arguments as *attachments to the respective level of the local head projection*. And as we shall see later, in the discussion of long-distance dependencies, it is by this move – as opposed to global variable substitutions in head projections – that we can correctly define scope constraints in long-distance constructions, given LTAG's principle of *extended domains of locality*.

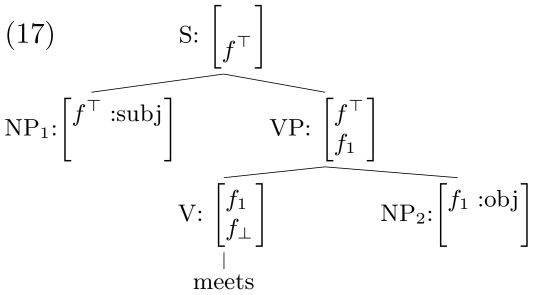
Our final version of the Tree Labelling Principle does now encode argument nodes $\text{arg}_N(\alpha)$ as *arguments of their local head projection*, by referring to the projection label of the argument's mother node (a tree internal node). That is, the projection index y in labels x_y:GF refers to the projection variable of the argument's mother (head projection) node.

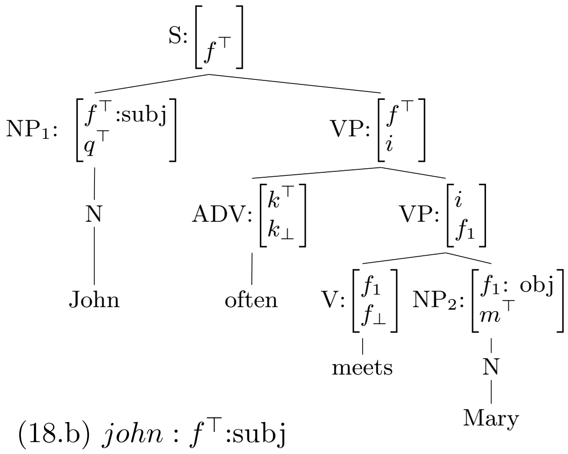
Tree Labelling Principle (Final Version)

- Anchor nodes: $\text{L}_b(anchor(\alpha)) = x_\perp$, x new variable from VAR
- Modifier-type auxiliary trees: $\text{L}_b(\text{root}(\beta)) = \text{L}_t(\text{foot}(\beta)) = x$, x new variable from VAR
- Projecting anchor node variable to all non-labelled tree internal nodes, introducing intermediate projection variables $\text{x}_\perp$ x_1 x_2 ... $\text{x}^\top$
- Argument nodes $\text{arg}_N(\alpha)$: $\text{L}_t(\text{arg}_N(\alpha)) = \text{L}_b(\text{mother}(\text{arg}_N(\alpha)))$: GF_N

The labelling of elementary trees differs only slightly from the previous version, the subject NP of *meets* being labelled $f^\top$:subj as before, since it attaches to the highest projection of the elementary tree, whereas the object NP attaches to projection level f_1, and is thus labelled f_1:obj. With variable substitutions $q^\top \rightarrow f^\top$:subj, $m^\top \rightarrow f_1$:obj, and $i \rightarrow f^\top$ the premises to meaning construction in (18.b) yield a successful proof in meaning derivation.

(17)





f_1 :obj $\multimap (f_\top$:subj $\multimap f^\top) : \lambda y, x.meet(x, y)$

(18.a) $john : q^\top$
$mary : m^\top$
$\lambda y, x.meet(x, y)$:f_1:obj $\multimap (f_\top$:subj $\multimap f^\top)$
$\lambda P, x.often(P(x))$:
$(i$:subj $\multimap i) \multimap (i$:subj $\multimap i)$

(18.b) $john : f^\top$:subj
$mary : f_1$:obj
$\lambda y, x.meet(x, y)$:f_1:obj $\multimap (f^\top$:subj $\multimap f^\top)$
$\lambda P, x.often(P(x))$:
$(f^\top$:subj $\multimap f^\top) \multimap (f^\top$:subj $\multimap f^\top)$

$$\dfrac{\dfrac{\dfrac{\lambda y, x.meet(x,y){:}f_1{:}\mathrm{obj} \multimap (f^\top{:}\mathrm{subj} \multimap f^\top) \quad mary{:}f_1{:}\mathrm{obj}}{\lambda x.meet(x, mary){:}f^\top{:}\mathrm{subj} \multimap f^\top} \quad \lambda P, x.often(P(x)){:}(f^\top{:}\mathrm{subj} \multimap f^\top) \multimap (f^\top{:}\mathrm{subj} \multimap f^\top)}{\lambda x.often(meet(x, mary)){:}f^\top{:}\mathrm{subj} \multimap f^\top} \quad john{:}f^\top{:}\mathrm{subj}}{often(meet(john, mary)){:}f^\top}$$

4.2.4 Deriving scope ambiguities

With our Tree Labelling Principle in place, we will now illustrate that LTAG semantics construction based on derived trees accounts for scope ambiguities induced by modifiers and NP quantifiers.

By identifying root and foot labels of modifiers, and due to local variable substitutions in tree composition, we account for the scoping behaviour of modifiers to take scope over other modifiers within their clausal projection. In particular, proper labelling conditions of argument nodes ensures that the meaning constructor's non-local variable stays local to the clause nucleus.[12]

In (19) we consider a case of modifier scope ambiguity, with one of the adverbs adjoining to S, the other to VP.[13] After substitutions, the set of premises allows for derivation of alternative meanings, by either first consuming the meaning constructor for *twice*, and then *intentionally*, or vice versa. That is, from the single derived tree we obtain ambiguous semantic analyses, with alternative modifier scopes: $intentionally(john, twice(call(john, mary)))$, and $twice(intentionally(john, call(john, mary)))$.

[12] See Section 4.4 for the analysis of predicative auxiliary trees.

[13] Example (19) brings us to the special case where modifier adjunction applies to the root node, which doesn't specify an upper label. For this case we need to refine the conditions for adjunction in tree composition by adopting clause (i), which provides an upper label for modifier adjunction to root nodes. Clause (ii) can then apply as before. Clause (ii) will also cover predicative auxiliary trees (see below).

(i) On adjunction of a modifier auxiliary tree β to a node n in α, if n does not specify an upper label, we instantiate an upper label, assigning it the value of n's lower label.

(ii) On adjunction of some auxiliary tree β to a node n in α, n's label features are split: the top label feature L_t of n, if instantiated, is assigned as the top label feature L_t of $root(\beta)$, and the bottom label feature L_b of n is assigned as the bottom label feature L_b of β's foot node.

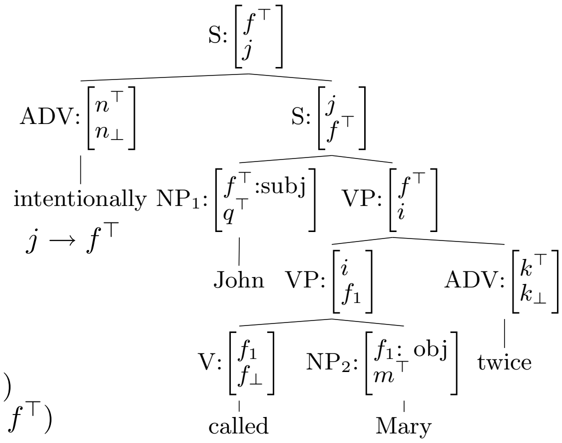
We further assume that for meaning derivation it is the lower label of the sentence's root node that constitutes the target of the proof in meaning derivation (resp. the variable it is substituted with).

(19)

$john : q^\top$
$mary : m^\top$
$\lambda y, x.call(x, y)$: f_1:obj $\multimap$ ($f^\top$:subj $\multimap$ $f^\top$)
$\lambda P, x.intent(x, P(x))$: ($j$:subj $\multimap$ j) $\multimap$ (j:subj $\multimap$ j)
$\lambda P, x.twice(P(x))$: (i:subj $\multimap$ i) $\multimap$ (i:subj $\multimap$ i)

Substitutions: $q^\top \rightarrow f^\top$:subj, $m^\top \rightarrow f_1$:obj, $i \rightarrow f^\top$, $j \rightarrow f^\top$

$john : f^\top$:subj
$mary : f_1$:obj
$\lambda y, x.call(x, y)$: f_1:obj $\multimap$ ($f^\top$:subj $\multimap$ $f^\top$)
$\lambda P, x.twice(P(x))$: ($f^\top$:subj $\multimap$ $f^\top$) $\multimap$ ($f^\top$:subj $\multimap$ $f^\top$)
$\lambda P, x.intent(x, P(x))$: ($f^\top$:subj $\multimap$ $f^\top$) $\multimap$ ($f^\top$:subj $\multimap$ $f^\top$)



Yet, for left and right adjoining VP modifiers as in *John intentionally called Mary twice*, assuming *standard derivation*, we obtain alternative derived trees with identical, ambiguous meanings. We therefore adopt Schabes and Shieber (1994)'s *extended derivations* for modifier-type auxiliary trees, allowing multiple adjunction to single nodes. Assuming further that simultaneous left and right adjunction to a single node produces a shared adjunction root node, as described in (Schabes and Waters 1995), we obtain a *single* derived tree, which yields the same ambiguity as in (19).

Finally, we need to account for scope restrictions. In Crouch and van Genabith (1999) scope restrictions are defined by *scope constaints*, which restrict the order of derivations in meaning construction, to yield corresponding scope meanings. Scope constraints refer to variables in glue expressions. But modifiers attaching to the same clause exhibit *identical* glue variables (after equality resolution) – they cannot be distinguished in the glue part. This problem could be solved by allowing scope constraints to refer to *glue : meaning pairs*, i.e. by exploiting the Curry-Howard isomorphism. In *John called Mary intentionally twice* the governing modifier could introduce a constraint *intentionally* $\prec$ *twice* – an instruction to *consume/apply* the *glue : meaning* pair associated with *intentionally* before the one associated with *twice*, corresponding to wide scope of *twice*.

Scope ambiguities with NP quantifiers are accounted for in the definition of the associated meaning constructors, in line with standard glue semantics (cf. Dalrymple (1999)). In (20) we display the entries for quantified pronominal NPs. For *Everyone meets someone*, we obtain – after substitutions – a set of instantiated meaning constructors which allows for alternative meaning derivations, corresponding to alternative quantifier scopes.

(20)

NP: $[g^\top]$ — everyone
$\lambda P.\forall x(person(x) \rightarrow P(x))$: $(g^\top \multimap X) \multimap X$

NP $[h^\top]$ — someone
$\lambda P.\exists x(person(x) \wedge P(x))$: $(h^\top \multimap X) \multimap X$

Instantiated mcs for *Everyone meets someone*
$\lambda P.\forall x(person(x) \rightarrow P(x))$:($f^\top$:subj $\multimap X) \multimap X$
$\lambda P.\exists x(person(x) \wedge P(x))$:($f_1$:obj $\multimap X) \multimap X$
$\lambda y, x.meet(x, y)$:f_1:obj $\multimap f^\top$:subj $\multimap f^\top$

4.3 Taking stock

Let us review the basic assumptions in our design of LL-based semantics construction for LTAG. As in glue semantics applied to LFG, semantics construction is lexicon driven, here by associating meaning constructors with lexicalised elementary trees. While glue semantics in LFG is based on f-structure – variables in the glue part of meaning constructors refer to instantiated f-structure nodes (or their semantic σ–projections) – glue semantics for LTAG is based on *labelled* derived trees. The principle for labelling elementary trees was introduced stepwise, in order to introduce as little additions to the LTAG framework as needed, and to motivate specific conceptual moves and machinery. Labelling of trees with features assigning upper and lower variables is a necessary extension to LTAG if semantics construction is based on *derived* as opposed to *derivation trees*, and we have presented evidence for serious shortcomings of the latter approach in Section 3. It is especially the labelling of *tree internal nodes by (projected) anchor labels* that extends basic assumptions of LTAG, which in its basic form only "talks about" root, substitution and adjunction nodes in elementary trees. As we have seen, in semantics construction from derived trees, reference to tree internal nodes is crucial for connecting variables in adjunction structures for modification. Further we have seen that non-tree local dependencies as in the case of VP modifying adverbs (and similarly for control verb constructions, see below) can only be captured by (i) labelling arguments *as arguments of their lexical head*, and (ii) association with grammatical function labels (or similar naming conventions). Finally, (iii), the labelling of arguments needs to specify the head projection level where attachment takes place in order to allow for a principled solution to the scoping of adverbs and – as we shall see – wh-elements in long-distance constructions.

In tree composition we establish equations between variables in upper and lower labels, based on two basic principles for substitution and adjunction. Variable substitutions in meaning constructors are uniquely based on these equations. That is, only those variables that are *local to substitution or modifier root nodes* play a role in establishing connections between the isolated variables assigned in elementary trees. Variables of intermediate projection nodes which are not touched by adjunction or substitution operations can be safely ignored in meaning construction.

We consider *head projection labelling* as the most crucial addition to LTAG's standard assumptions. It is a necessary extension for semantics construction from derived trees, and a crucial assumption to account for the most characteristic and difficult aspects of LTAG syntax and its syntax-semantics interface, namely adjunction and its interplay with semantic composition and the definition of scope. This is evident in the analysis of modifiers and scope ambiguity, and required for long distance dependencies in Section 4.6.

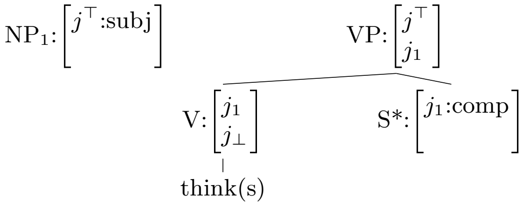
4.4 Bridging the syntax-semantics non-isomorphism in LTAG

As outlined in Section 2, a main characteristic of LTAG syntax is that – due to the principle of extended domains of locality and the existence of long distance phenomena – sentential embedding is necessarily driven by adjunction in syntax, that is, by the same syntactic operation that applies to optional (and recursive) modifiers. This results in a non-isomorphism between adjunction in syntax on the one hand, and the semantic operations of complementation vs. modification on the other. As we saw in discussion of Joshi and Vijay-Shanker (1999), this generates inverted embedding structures in derivation trees, as opposed to correct embeddings in derived tree structures, and leads to complications in semantics construction from derivation trees.

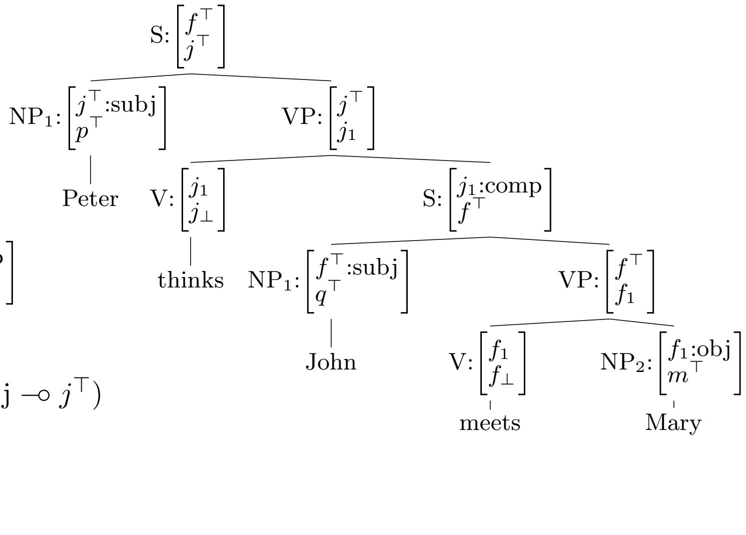
One would therefore expect that in our approach, where the syntax-semantics interface is built on the derived tree, the problem of this non-isomorphism is automatically circumvented. But this is not the case. The problem arises in our approach as well, yet in a slightly different way.

Let us take a look at a sentence embedding verb like *think* in a simple embedding configuration (21). Sentence embedding verbs (and similarly control verbs) are represented as auxiliary trees, yet differ from *modifier-type* auxiliary trees in that their foot node corresponds to an argument of the

(21)



$\lambda P, x.think(x, P)$: j_1:comp $\multimap$ ($j^\top$:subj $\multimap$ $j^\top$)



lexical head. In LTAG this difference is captured by the distinction between *modifier* and *predicative* auxiliary trees. In our Tree Labelling Principle labelling of root and foot nodes with identical variables was restricted to *modifier auxiliary trees*. The foot node of a *predicative auxiliary tree*, by contrast – as argument node of the lexical anchor – falls under the conditions for argument nodes, and is assigned a composed label, consisting of a projection variable and a grammatical function label.[14] For the *predicative* auxiliary tree *think* we thus obtain a labelled elementary tree as displayed in (21). In contrast to modifier trees, root and foot node labels are distinct.

In tree composition, the bottom label of the foot node is instantiated with the bottom label of the S node to which adjunction applies ($f^\top$ in (21)), in accordance with our principles for assigning top and bottom labels in tree composition.[15]

Although the derived tree does not reflect any crucial difference between adjunction and substitution, we need to adjust the conditions for *variable equations* in tree composition to account for the special aspects of semantic composition with *predicative auxiliary trees*, as opposed to *modifier auxiliary trees*. On adjunction of modifiers root and foot node variables need to be equated with the projection variable of the adjunction node, to allow modifiers to take scope within the maximal projection. This we obtained by equating upper and lower labels at the modifier's root node in the derived tree. For *predicative auxiliary trees*, by contrast, we need to link, i.e. equate, the labels of argument foot node and adjunction node, parallel to the standard case of argument *substitution*.

All other things being equal, then, we account for the non-isomorphism of adjunction in semantics construction through refinement of variable equations in tree composition: We distinguish between adjunction of *modifier* and *predicative* auxiliary trees, the latter being defined along the lines of standard cases of argument substitution. Thus, in variable equation, on adjunction of *predicative auxiliary trees* the foot node plays the role the substitution node n_α plays in substitution, and the adjunction node n_α plays the role of the root node of the inserted tree in substitution.

Variable Equations in Tree Composition (Final Version)

- **Substitution:** When substituting β into α at node n_α, add equation: $\mathrm{L}_t(n_\alpha)=\mathrm{L}_b(\mathrm{root}(\beta))$
- **Adjunction:** When adjoining β to α at node n_α,
  if β is a *modifier auxiliary tree*, add equation: $\mathrm{L}_t(n_\alpha) = \mathrm{L}_b(\mathrm{root}(\beta))$
  if β is a *predicative auxiliary tree*, add equation: $\mathrm{L}_t(\mathrm{foot}(\beta)) = \mathrm{L}_b(n_\alpha)$

With these adjustments we trigger substitution of $f^\top$ by j_1:comp in the set of meaning constructors (22) assembled for (21).[16] After substitution of variables, complex terms such as j_1:comp:subj are treated as atoms in matching producers and consumers in Linear Logic meaning derivation.

(22) Substitutions: $p^\top \to j^\top$:subj, $q^\top \to f^\top$:subj, $m^\top \to f_1$:obj, $f^\top \to \mathrm{j}_1$:comp

$peter : p^\top$	$peter : j^\top$:subj
$john : q^\top$	$john : j_1$:comp:subj
$mary : m^\top$	$mary : f_1$:obj
$\lambda y, x.meet(x, y)$:f_1:obj $\multimap$ ($f^\top$:subj $\multimap f^\top$)	$\lambda y, x.meet(x, y)$:f_1:obj $\multimap$ (j_1:comp:subj $\multimap j_1$:comp)
$\lambda P, x.think(x, P)$:j_1:comp $\multimap$ ($j^\top$:subj $\multimap j^\top$)	$\lambda P, x.think(x, P)$:j_1:comp $\multimap$ ($j^\top$:subj $\multimap j^\top$)
	$\vdash think(peter, meet(john, mary))$

[14] We assume a mapping from argument foot nodes S with discriminating features (such as [$\pm fin$] etc.) to corresponding grammatical functions COMP and XCOMP.

[15] Since the upper label of the adjunction node is not instantiated, the upper label of the root node is instantiated to $f^\top$ (see the refined definition for adjunction in fn. 13). The target variable in the meaning proof is $\mathrm{L}_b(\mathrm{root}) = j^\top$.

[16] Note that in this example the order in which substitutions are triggered is critical. In general, though, we can avoid specification of some canonical order for substitutions by *repeatedly* applying substitutions to the modified premises in the same, arbitrarily fixed order, until no more modifications are triggered in a complete run. Alternatively, one could adopt a general resolution algorithm, along the lines of Kaplan and Bresnan (1995).

4.5 Non-tree local dependencies II: Control constructions

We now show how LL-based semantics construction on the basis of labelled LTAG trees accounts for non-tree local dependencies in control constructions.

4.5.1 Control constructions in LFG

We first take a quick look at the treatment of control in LFG syntax and LL-based glue semantics. Lexical entries of control verbs specify the relation between controller and embedded subject in terms of functional equations. In the resulting f-structure the matrix SUBJ of *tries* is unified with the SUBJ of the embedded verb, that is, they are instantiated to some unique f-structure node (g in (23)). The meaning constructor associated with *tries*[17] consumes the VP meaning of the embedded verb $(\uparrow\text{XCOMP SUBJ}_\sigma) \multimap (\uparrow\text{XCOMP}_\sigma)$ to produce the VP meaning of *tries*, which then consumes the overt matrix subject to produce the sentence meaning $((\uparrow\text{SUBJ}_\sigma) \multimap \uparrow_\sigma)$. Parallel to linear logic derivation based on the glue formulae the meaning is computed in the associated meaning terms.

(23) tries V,
$(\uparrow \text{PRED})= \text{`TRY}\langle (\uparrow \text{SUBJ})\ (\uparrow \text{XCOMP})\rangle\text{'}$
$(\uparrow \text{SUBJ})= (\uparrow \text{XCOMP SUBJ})$
$\lambda P, x.try(x, P)$:
$((\uparrow\text{XCOMP SUBJ}_\sigma) \multimap (\uparrow\text{XCOMP}_\sigma)) \multimap ((\uparrow\text{SUBJ}_\sigma) \multimap \uparrow_\sigma)$

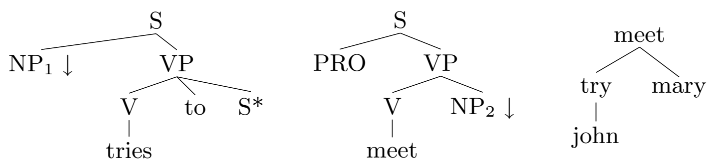
Mcs for: *John tries to meet Mary*
$john : g$
$mary : m$
$\lambda y, x.meet(x, y){:}m \multimap (g \multimap h)$
$\lambda P, x.try(x, P){:}(g \multimap h) \multimap (g \multimap f)$

$$\dfrac{\dfrac{\dfrac{\lambda y, x.meet(x, y) : m \multimap (g \multimap h) \qquad mary : m}{\lambda x.meet(x, mary) : g \multimap h} \qquad \lambda P, x.try(x, P) : (g \multimap h) \multimap (g \multimap f)}{\lambda x.try(x, \lambda x.meet(x, mary)) : g \multimap f} \qquad john : g}{try(john, \lambda x.meet(x, mary)) : f}$$

4.5.2 Control constructions in LTAG

In LTAG syntax control verbs are encoded as *predicative auxiliary trees*, since they can occur in long-distance dependencies. Infinite verbs embedded under control verbs do not display an overt subject. The corresponding elementary tree therefore encodes a PRO subject node. Due to adjunction, the *derivation tree* for *John tries to meet Mary* incorrectly represents the control verb as dependent on the embedded verb, *meet*. Moreover, since no substitution takes place into the subject argument position of *meet* the derivation tree does not represent *John* as an argument of *meet*.

(24)

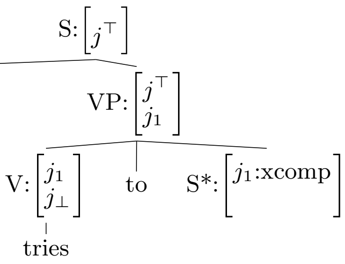


4.5.3 Glue Semantics for LTAG control contructions

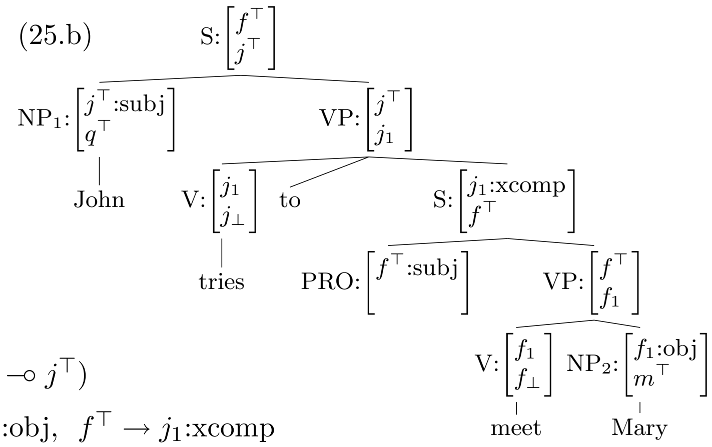
Applying our Tree Labelling Principle to the *predicative* auxiliary tree *tries* (25.a), we label the subject NP and the adjunction foot node S* as subj and xcomp arguments, anchored to the respective head projection variables $j^\top$ and j_1. Labelling and meaning constructor for the embedded infinitive etree *meet* (see (24)) is identical to (17), except for the presence of a subject PRO argument, which is not a substitution node. For *John tries to meet Mary* we obtain the labelled derived tree (25.b). With variable substitutions in the set of assembled meaning constructors, we obtain a successful proof of the sentence meaning, similar to the meaning derivation in the LFG analysis.

[17]We follow the analysis given in Asudeh (2000) and Asudeh (2001).

(25.a)



(25.b)



$\lambda P, x.try(x, P)$:
$(j_1\text{:xcomp:subj} \multimap j_1\text{:xcomp}) \multimap (j^\top\text{:subj} \multimap j^\top)$

Substitutions: $q^\top \to j^\top\text{:subj}$, $m^\top \to f_1\text{:obj}$, $f^\top \to j_1\text{:xcomp}$

$john : q^\top$	$john : j^\top\text{:subj}$
$mary : m^\top$	$mary : f_1\text{:obj}$
$\lambda y, x.meet(x, y) : f_1\text{:obj} \multimap (f^\top\text{:subj} \multimap f^\top)$	$\lambda y, x.meet(x, y)$: $f_1\text{:obj} \multimap (j_1\text{:xcomp:subj} \multimap j_1\text{:xcomp})$
$\lambda P, x.try(x, P)$: $(j_1\text{:xcomp:subj} \multimap j_1\text{:xcomp}) \multimap (j^\top\text{:subj} \multimap j^\top)$	$\lambda P, x.try(x, P)$: $(j_1\text{:xcomp:subj} \multimap j_1\text{:xcomp}) \multimap (j^\top\text{:subj} \multimap j^\top)$

$$\dfrac{\dfrac{\dfrac{\lambda y, x.meet(x,y) : f_1\text{:o} \multimap (j_1\text{:xcmp:s} \multimap j_1\text{:xcmp}) \quad mary : f_1\text{:o}}{\lambda x.meet(x, mary) : j_1\text{:xcmp:s} \multimap j_1\text{:xcmp}} \quad \lambda P, x.try(x,P) : (j_1\text{:xcmp:s} \multimap j_1\text{:xcmp}) \multimap (j^\top\text{:s} \multimap j^\top)}{\lambda x.try(x, \lambda x.meet(x, mary)) : j^\top\text{:s} \multimap j^\top} \quad john : j^\top\text{:s}}{try(john, \lambda x.meet(x, mary)) : j^\top}$$

On closer inspection, though, an interesting difference emerges between the LFG and LTAG analyses. In LFG coreference of controller and controlled subject is represented in f-structure by mapping them to a single f-structure node, due to the control relation in the lexical entry of the control verb. This is explicit in the instantiated meaning constructor of *tries*. In the LTAG analysis this control relation is not expressed in terms of tree labelling. This is seen in the meaning constructor of *tries*, where embedded and matrix subject are referred to by distinct labels, j_1:xcomp:subj and $j^\top$:subj, both before and after variable substitutions. Despite the missing identification of controller and controlled argument, meaning derivation in the LTAG analysis yields the same sentence meaning.

This fact is interesting in that it points to an important difference between the LFG and LTAG frameworks. LFG's syntactic f-structure representation effectively encodes predicate-argument dependencies. In LTAG syntax – even if augmented with grammatical relation labelling – neither labelled derived trees, nor derivation trees reflect full-fledged predicate-argument structures. While the embedded subject in (25.b) displays an upper label $f^\top$:subj, there is no equivalent to a control relation to establish coreference of controller ($j^\top$:subj) and controlled argument. It is even more compelling, then, that in coupling LTAG syntax with glue semantics construction from labelled derived trees, we obtain a well-formed meaning, i.e., the same meaning we obtain from fully specified dependencies in LFG f-structures.[18] We will come back to this issue in Section 5.

[18] The reader might object here, since the meaning of controlled infinitives is a property, which does not express the control relation in question. This is most evident with transitive control verbs, where the derived meaning does not encode a subject vs. object control reading. For *John promised Mary to leave*, e.g. we obtain the meaning $promise(john, mary, \lambda x.leave(x))$. Meaning postulates need to encode which of the matrix arguments satisfies the property.

A slight revision of the meaning of control verbs proposed in Asudeh(2000, 2001) solves this problem. Instead of a property meaning, *tries* could select an open proposition $\lambda P, x.try(x, P(x))$, a verb like *promise* could be assigned the meaning $\lambda P, y, x.promise(x, y, P(x))$. In meaning derivation, then, consumption of the matrix subject will appropriately instantiate the embedded open proposition ($promise(john, mary, leave(john))$ in the above example). Note that an open proposition analysis still accounts for the well-known inference problems with control verbs, if we

4.6 Long-distance dependencies and wh-scope

We finally turn to long-distance constructions, which turned out to be particularly difficult for LTAG semantics construction from derivation trees. Wh-constructions involve the additional complexity of defining an operator semantics for wh-phrases such as *who(m)*, and defining the scope of the wh-operator. As we shall see now, it is the treatment of scope and wh-semantics in long-distance constructions which motivated much of the architecture we developed in the previous sections.

Labelling etrees for long extraction (26) displays an etree for wh-object extraction, with a fronted wh-object NP and a corresponding (unlabelled) trace. Due to object fronting and the additional projection level, argument labelling differs from (17): the object attaches to the highest projection, while the subject NP attaches to the f_2 projection level. Note further that the meaning constructor is specified to produce $f2$ as (the glue equivalent of) the meaning of the clause headed by *meet*, which is the label of the tree internal node that is the target for adjunction for clause embedding auxiliary trees, such as *think*. The assignment of f_2 as the target of sentence meaning reflects – on the semantic side – the split that is triggered – in syntax – by adjunction of sentence embedding trees, which split the basic elementary tree into a long-distance extracted projection level ($f^\top$), and the clause's basic tree trunk, cut at projection level f_2. This is seen in (27): on adjunction of *think* to S[−wh], the foot node's argument label j_1:comp is linked to f_2.[19]

Deriving meanings for long extractions Ignoring the role of the wh-phrase for a moment, by looking at the lower projections in (27), a meaning for f_2 can only be derived by *hypothesising*, or *assuming* a resource $z : f^\top$:obj, z an unbound variable, to fill the object argument position of *meet*. This kind of hypothetical reasoning we observe in the upper part of the meaning derivation (28), which leads to the partial meaning $meet(john, z) : j_1$:comp, z the unbound variable from the assumed premise $[z : f^\top$:obj$]$. Due to variable equations, j_1:comp $(= f_2)$ is consumed by *think*, which after consumption of its subject (j_1:subj) delivers the partial meaning $think(peter, meet(john, z)) : f^\top$. Since this meaning was obtained on the *assumption* of $[z : f^\top$:obj$]$, the assumption has to be discharged in order to yield a valid proof. This is done in the implication introduction step (I) in (28), which yields the partial meaning $\lambda z.think(peter, meet(john, z)) : f^\top$:o $\multimap f^\top$. This basically says that *had we found* a premise $z : f^\top$:o, whatever the meaning of z, it is of this z that Peter thinks that John meets z. The meaning constructor associated with *whom* in (28) is now easily understood: it combines with a (property) meaning which was built on the assumption of an object-NP meaning for *meets* ($f^\top$:o $\multimap f^\top$), and should deliver the meaning of the root node ($f^\top$) to which *whom* attaches, by applying the function $\lambda v.who(v)$ to this partial meaning $\lambda z.think(peter, meet(john, z))$: ($f^\top$:o $\multimap f^\top$) $\multimap f^\top : \lambda v.who(v)$.

Defining wh-scope The difficult aspects in meaning construction for long wh-extractions are the following: (i) the scope of the wh-operator is constrained by the position of the wh-phrase in the *derived* tree. That is, we are not allowed to derive narrow scope of the wh-operator in (27) and (28). (ii) from the local elementary tree of *whom* we cannot access the nodes S[−wh] or S[+wh] of the verb's tree, which are crucial to define these scope constraints. The labelling of the etree *whom* captures the non-tree local variable $f^\top$ of the tree it is substituted in by assigning its root node the label $X^\top$:obj, X a metavariable. The meaning constructor is defined by reference to $X^\top$. In tree composition, substitution triggers the equality $X^\top$:obj $= f^\top$:obj, and at the same time instantiates $X^\top$ to $f^\top$. With global substitutions as given in (27) we obtain a set of premises that yield the

---

assume quantification to range over a variable P of type property, just as in the classical property analysis.

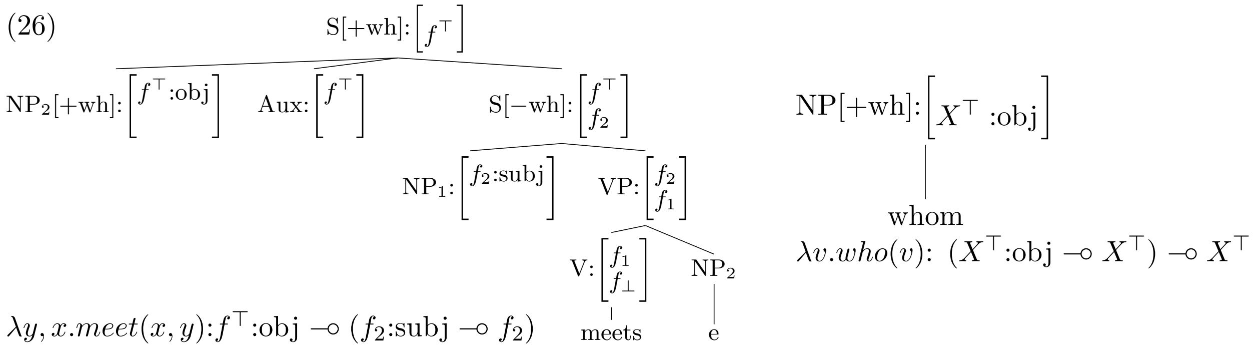
Peter tries to leave.	try(peter,λx.leave(x)(peter))
John tries everything Peter tries.	∀P [try(peter,P(peter))→try(john,P(john))] where P = λx.leave(x)
⊢ John tries to leave.	⊢ try(john,λx.leave(x)(john))
⊬ John tries for Peter to leave.	

With these additional assumptions we can safely argue that Linear Logic based semantics construction is in fact able to bridge the missing control relation in LTAG syntax.

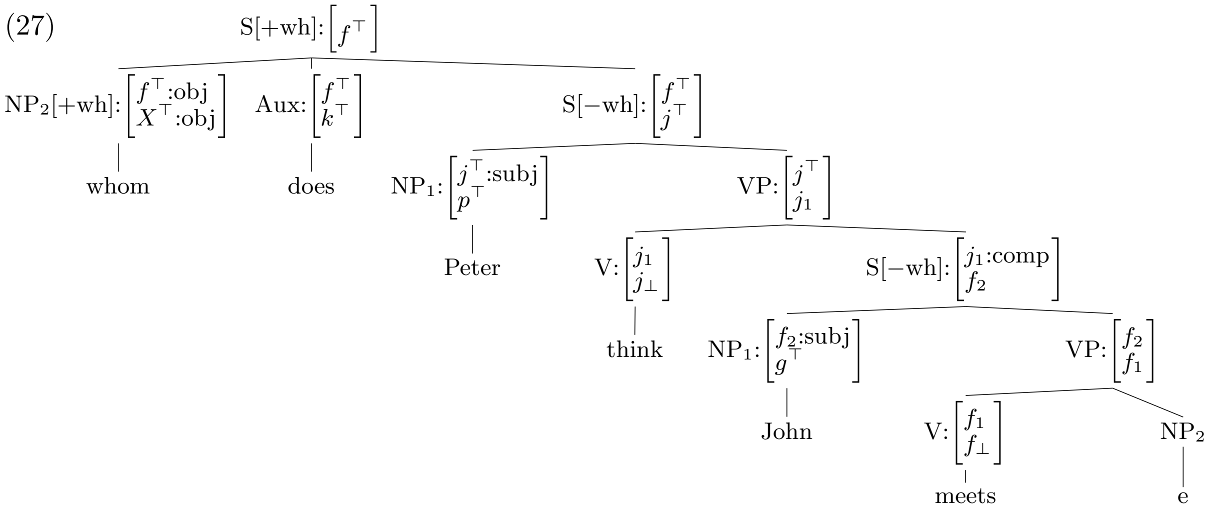
[19] See the variable equation conditions for predicative auxiliary trees (p. 15).

desired meaning in (28), with wide scope of the wh-operator over the clause headed by *think*. Wide scope is enforced by the definition of *whom*, which is constrained to consume $f^\top$:obj $\multimap f^\top$, where $f^\top$ is the clause label of the projection *whom* is attached to. Narrow scope is prohibited due to the labelling of the etree *meets*, which produces the resource f_2, and thus a partial meaning $meet(jn, z) : f_2$, where $f_2 = j_1$:comp by variable substitution. If implication introduction (I) were to apply to this partial meaning, the resulting meaning term $\lambda z.meet(john, z) : f^\top$:obj $\multimap j_1$:comp would not provide an adequate resource for *whom* to define narrow scope under *think*.

(26)



(27)



Substitutions:

$g^\top \to f_2$:subj, $\quad f_2 \to j_1$:comp, $\quad p^\top \to j^\top$:subj, $\quad j^\top \to f^\top$, $\quad X^\top$:obj$\to f^\top$:obj, $\quad X^\top \to f^\top$

$\lambda v.who(v) : (X^\top$:obj $\multimap X^\top) \multimap X^\top$
$peter : p^\top$
$john : g^\top$
$\lambda y, x.meet(x, y)$:$f^\top$:obj $\multimap$ (f_2:subj $\multimap f_2$)
$\lambda P, x.think(x, P)$:j_1:comp $\multimap$ ($j^\top$:subj $\multimap j^\top$)

$\lambda v.who(v) : (f^\top$:obj $\multimap f^\top) \multimap f^\top$
$peter : f^\top$:subj
$john : j_1$:comp:subj
$\lambda y, x.meet(x, y)$:$f^\top$:obj $\multimap$ (j_1:comp:subj $\multimap j_1$:comp)
$\lambda P, x.think(x, P)$:j_1:comp $\multimap$ ($f^\top$:subj $\multimap f^\top$)
$\vdash who(\lambda z.think(peter, meet(john, z)))$

(28)

$$\dfrac{\lambda v.who(v) :(f^\top{:}o \multimap f^\top) \multimap f^\top \qquad \dfrac{\dfrac{pt : f^\top{:}s \qquad \dfrac{\lambda P, x.think(x, P) : j_1{:}comp \multimap (f^\top{:}s \multimap f^\top) \qquad \dfrac{jn : j_1{:}comp{:}s \qquad \dfrac{\lambda y, x.meet(x, y) : f^\top{:}o \multimap (j_1{:}comp{:}s \multimap j_1{:}comp) \quad [z : f^\top{:}o]}{\lambda x.meet(x, z) : j_1{:}comp{:}s \multimap j_1{:}comp}}{meet(jn, z) : j_1{:}comp}}{\lambda x.think(x, meet(jn, z)) : f^\top{:}subj \multimap f^\top}}{think(pt, meet(jn, z)) : f^\top}}{\lambda z.think(pt, meet(jn, z)) : f^\top{:}o \multimap f^\top}\,I}{who(\lambda z.think(pt, meet(jn, z))) : f^\top}$$

In sum, by tree internal projection labelling, meaning constructors in extraction contexts can refer to the adjunction node label (f_2) which defines the upper bound of the embedded clause trunk, and thereby correctly constrains the scope of the extracted wh-phrase, which by reference to $X^\top$ ($=f^\top$) is forced to take scope over the projection level to which it attaches in the derived tree.

Note that in the labelled derived tree (27), similar to the case of control constructions, substitution of equated variables does not yield a well-formed dependency structure. The object *whom* labelled $(f^\top\text{:obj} \multimap f^\top) \multimap f^\top$ is not linked to *meets*, which is looking for a premise labelled $f^\top$:obj. So, similar to what we observed in control constructions, it by Linear Logic meaning derivation that the missing dependency link is established, here in terms of hypothetical reasoning. This contrasts again with the architecture of LFG, where extracted phrases are represented in their basic argument position in the f-structure, local to their governing predicate, by the interplay of functional uncertainty equations and completeness and coherence constraints.

Having dealt with long-distance dependencies, we can resume our discussion of semantics construction from derivation trees in Section 3. These approaches were shown to have difficulties in representing the embedding structure of long distance constructions involving adjunction of sentential embedding and raising verbs to *distinct* nodes in the basic verb's elementary tree, as in examples (5) and (8). For reasons of space we cannot go through the examples in detail, but we sketch the relevant aspects here.[20]

For long object topicalisation in (5) and long object extraction in (8) we assume transitive verb trees along the lines of *meet* in (26), without auxiliary insertion at the root node. Raising verbs like *seem* are represented as modifier-type VP auxiliary trees, with a meaning constructor $\lambda P.seem(P)$: $i \multimap i$, where i the variable assigned to root and foot node labels. Variable substitutions in tree composition ensure that in the derived meaning for (5) (*Spicy hotdogs he claims Mary seems to adore*) *claim* takes scope over *seem*, which in turn takes scope over *adore*. Given appropriate meaning constructors for question embedding verbs, our account provides the correct semantics and embedding structure in cases like (8).

4.7 Related approaches: Semantics construction for D-Tree Grammars

Finally we briefly mention two approaches which are related to our work, yet at the same time need to be differentiated. Hepple (1999) proposes categorial-style semantics construction from derived trees of D-Tree Grammar (DTG, Rambow et al. (1995)), to dispense with the more problematic, process-based interpretation model provided by derivation structures. To overcome problems faced in the analysis of NP quantification he then provides a small fragment for glue semantics from DTG derived trees,[21] exploiting the greater flexibility of glue semantics in meaning assembly. Yet, while closely related to LTAG, DTG elementary trees (d-trees) can provide a larger syntactic context than LTAG etrees[22] and therefore lead to a different set-up for tree labelling, as compared to the LTAG framework we were dealing with. Still, we believe that comparison of our LTAG-based approach to (an extended fragment of) Hepple's account in DTG could lead to interesting insights into the special aspects of the respective frameworks in the syntax-semantics interface.

Muskens (2001) develops a very elegant description-based model of syntax and the syntax-semantics interface, which allows syntactic and semantic representations to be highly underspecified. The model is applied to LTAG syntax, yet again coupled with extension to tree descriptions as used in D-Tree Grammar. As in Hepple's and our approach, semantics construction is not guided by derivation structures. Elementary tree descriptions are enriched with semantic expressions in predicate logic, and instructions for meaning composition. The design of the syntax-semantics differs from our approach in using D-Tree Grammar and a different model for semantic composition. Detailed discussion and comparison cannot be provided within the scope of this paper.

[20] The reader is invited to go through the analysis in detail.

[21] The fragment in Hepple (1999) does not cover VP modification, control or long-distance dependencies.

[22] Especially for operators in long-distance dependencies and quantifiers.

5 What we learn about the relation between LTAG and LFG

We defined a Linear Logic-based semantics interface for LTAG syntax which is lexicon driven, and builds on the structure of derived trees. We argued that a major extension to the LTAG framework is *labelling of tree internal nodes by projected anchor variables*. Since we operate on derived trees, tree composition records variable equations which are restricted to variables of *substitution and adjunction nodes*, and allow for coherent meaning composition from assembled meaning constructors.

We noted a striking difference of LTAG tree labelling as opposed to the LFG framework: while in LFG all levels in head projection chains are unfied in terms of $\uparrow=\downarrow$ equations, LTAG tree labelling can restrict itself to equation of variables at substitution and adjunction nodes. This is of course due to the principle of strict lexicalisation and localisation of arguments in elementary trees.

We observed that labelled LTAG derived trees are not merely a mirror image of LFG c-structures annotated with f-descriptions. LTAG labelled trees do not encode control relations, nor are extracted arguments represented as local to their governing predicates. That is, we could conceive of labelled LTAG trees as an impoverished version of annotated c-structure trees in LFG: impoverished in that they lack functional path and uncertainty equations, and only allow for restricted variable equations, at local substitution and adjunction nodes. As a consequence, labelled derived trees (including variable equations) do not in general encode dependency structures, yet they provide semantically valid embedding structures, as opposed to LTAG derivation trees. An interesting observation emerging from this comparison is that this lack is compensated by LL-based semantics construction, which establishes the correct dependencies at the level of meaning representation.

It is especially with respect to long distance constructions that semantics construction from constitutent trees proved to be superior to meaning composition from derivation trees. This we achieved by a major addition to the LTAG framework, in that we label arguments to record the local projection level to which they attach. This addition is critical for the definition of scope constraints, as well as the correct definition of embedding structures, in particular in long extraction contexts with multiple adjunctions into single elementary trees. Moreover, we could show that glue semantics from labeled derived trees accounts for modifier and quantifier scope ambiguities.

Should we conclude from this exercise, then, that the additional power that LFG provides – the projection of f-structure by functional descriptions (in particular functional uncertainty, and other path equations), completeness and coherence constraints in f-structure, and global variable equations of head projection chains – is a luxury, or formal overhead which we could dispense with on the basis of a sparser syntactic formalism? The answer is certainly – No.

It is, first of all, a matter of conceptual clarity and modularity of linguistic representation which calls for an adequate *syntactic* representation of predicate-argument relations, as it is provided by dependency, or f-structure representations. Second, it is a well-known fact that tree composition operations in LTAG syntax are restricted in formal power, and do not extend to languages with special word order properties. Example (29) from Kashmiri, discussed in Rambow et al. (1995), illustrates such a case where wh-words end up in sentence-second position, preceded by a topic from the matrix clause. This type of interleaved combination of elementary trees, called *subsertion* in D-Tree Grammars, cannot be accounted for by adjunction of atomic, elementary lexicalised trees. – A case for (functional uncertainty in) f-structure.

(29)	Rameshan	kyaa$_i$	chu	baasaan	[ki	me	kor	t$_i$]
	Ramesh$_{\mathrm{ERG}}$	what	is	believe$_{\mathrm{Nperf}}$	that	I$_{\mathrm{ERG}}$	do	

What does Ramesh believe that I did?

6 Conclusion

In conclusion we argue that LTAG derivation trees are inappropriate for principle-based semantic composition. We developed an account for semantics construction from LTAG *derived trees* by establishing a tree labelling regime as an interface to Linear Logic-based meaning deduction. We showed that glue semantics from LTAG derived trees captures non-tree local dependencies in modification and control constructions, and extends to the adjunction-based analysis of long-distance dependencies, which is, however, formally and empirically more restricted than LFG's analysis in terms of functional uncertainty. Linear Logic-based semantics from *derived trees* successfully bridges LTAG's non-isomorphism between syntactic and semantic operations. It further allows us to derive scope ambiguities induced by modifiers and quantified NPs, and accounts for scope constraints in long-distance extraction contexts.

We established that labelled LTAG derived trees do not correspond to full-fledged dependency structures, as opposed to LFG f-structures, since LTAG tree labelling does not extend to the formal power of f-descriptions in LFG. However, dependencies which cannot be established by local variable substitutions can be appropriately bound at the meaning level, that is, through Linear Logic-based meaning assembly in the glue semantics interface.

References

Asudeh, A. (2000). Functional Identity and Resource-Sensitivity in Control. In Butt, M. and King, T., editors, *Proceedings of the LFG00 Conference*, pages 2–24, University of California, Berkley. CSLI Online Publications, `http://csli-publications.stanford.edu/`.

Asudeh, A. (2001). A Resource-Sensitive Semantics for Equi and Raising. In Beaver, D. and Kaufmann, S., editors, *Proceedings of Semantics Fest 2000*, pages 5–25. CSLI Publications, Stanford, CA.

Crouch, R. and van Genabith, J. (1999). Context Change, Underspecification, and the Structure of Glue Language Derivations. In Dalrymple, M., editor, *Semantics and Syntax in Lexical Functional Grammar*, pages 117–189. MIT Press.

Dalrymple, M., editor (1999). *Semantics and Syntax in Lexical Functional Grammar*. MIT Press.

Dalrymple, M. (2001). *Lexical-Functional Grammar*, volume 34 of *Syntax and Semantics*. Academic Press.

Hepple, M. (1999). A Functional Interpretation Scheme for D-Tree Grammars. In *Proceedings of the Third International Workshop on Computational Semantics*, pages 117–130, KUB, Tilburg.

Joshi, A. (1987). An Introduction to Tree Adjoining Grammar. In Manaster-Ramer, A., editor, *Mathematics of Language*. John Benjamins, Amsterdam.

Joshi, A. and Vijay-Shanker, K. (1999). Compositional Semantics with Lexicalized Tree-Adjoining Grammar (LTAG): How Much Underspecification is Necessary? In Bunt, H. and THijsse, E., editors, *Proceedings of the Third International Workshop on Computational Semantics (IWCS-3)*, pages 131–145, Tilburg.

Kallmeyer, L. and Joshi, A. (1999). Factoring Predicate Argument and Scope Semantics: Underspecified Semantics with LTAG. In Dekker, P., editor, *Proceedings of the 12th Amsterdam Colloquium*, pages 169–174.

Kaplan, R. and Bresnan, J. (1995). A Formal System for Grammatical Representation. In *Formal Issues in Lexical-Functional Grammar*. CSLI Lecture Notes, No.47.

Muskens, R. (2001). Talking about Trees and Truth-conditions. *Journal of Logic, Language, and Information*, 10(4).

Rambow, O., Vijay-Shanker, K., and Weir, D. (1995). D-Tree Grammars. In *Proceedings of ACL-95*, pages 151–158.

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

Schabes, Y. and Waters, R. C. (1995). Tree Insertion Grammar: A cubic-time, parsable formalism that lexicalizes context-free grammar withough changing the trees produced. In *Computational Linguistics 21(4)*, pages 479–513.

Shieber, S. and Schabes, Y. (1990). Synchronous Tree-Adjoining Grammars. In *Proceedings of COLING*, pages 1–6.