Feature Logic-based Semantic Composition: A Comparison between LRS and LTAG

Frank Richter[*]
University of Tübingen

Laura Kallmeyer[†]
University of Tübingen

In this paper we will explore the similarities and differences between two feature logic-based approaches to the composition of semantic representations. The first approach is formulated for Lexicalized Tree Adjoining Grammar (LTAG, Joshi and Schabes 1997), the second is Lexical Resource Semantics (LRS, Richter and Sailer 2004) and was first defined in Head-driven Phrase Structure Grammar. The two frameworks have several common characteristics that make them easy to compare: 1. They use languages of two-sorted type theory for semantic representations. 2. They allow underspecification: LTAG uses scope constraints $\geq$ while LRS provides component-of constraints $\lhd$. 3. They use feature logics for computing semantic representations. 4. They are designed for computational applications. By comparing the two frameworks we will also point out some characteristics and advantages of feature logic-based semantic computation in general.

1 Introduction

Except for a few early and largely informal explorations of the relationship between semantic representations in unification-based frameworks using typed feature logics (TFLs) and the lambda calculus-based Montague Grammar of mainstream research on semantics in linguistics (c.f. Moore 1989; Nerbonne 1992), for a long time there was little or no explicit connection between these two techniques for semantic representations. Sailer (2003) finally proved that a feature logic for Head-driven Phrase Structure Grammar (HPSG, Pollard and Sag 1994) such as Relational Speciate Re-entrant Language (RSRL, Richter 2004b) is sufficiently expressive to encode a higher-order logic such as Intensional Logic or two-sorted type theory (Ty2, Gallin 1975), and the combinatorial system of the lambda calculus. This means that it is mathematically possible to completely embed a categorial semantics such as Flexible Montague Grammar (Hendriks 1993) within a TFL grammar. However, undecidability results make it unattractive to integrate RSRL or alternative feature logics that are expressive enough for a di-

* University of Tübingen, Seminar für Sprachwissenschaft, Wilhelmstraße 19, D-72074 Tübingen, Germany. Email: fr@sfs.uni-tuebingen.de
† University of Tübingen, Collaborative Research Center 441, Nauklerstr. 35, D-72074 Tübingen, Germany. Email: lk@sfs.uni-tuebingen.de

rect logical specification of HPSG grammars wholesale in grammar development environments. On the other hand, reducing the expressivity of the feature logic prevents a logical specification in the feature logic of the syntax of Ty2 and basic operations of the lambda calculus such as beta reduction. In addition, it should be noted that it is at least an open question whether the assumption is justified that semantic composition in natural languages can be adequately described by the techniques provided by the lambda calculus. Other means of semantic composition might turn out to be better suited to analyze the relationship between the semantics of syntactically complex expressions and their constituents in natural languages.

To prevent misunderstandings it should be stressed very clearly that we do not mean to say that the lambda calculus is in any sense insufficient for specifying the composition of meanings along the syntactic structures of natural languages. Due to the universal nature of the lambda calculus as an abstract characterization of computation it is likely that this goal could be achieved. The question that we want to raise is whether the lambda calculus is a linguistically *adequate* tool for expressing as directly as possible the most important linguistic generalizations over the mechanisms of semantic composition in natural languages. One of the prime purposes of this paper is to present key concepts of an attractive alternative to the lambda calculus. These key concepts will appear in two mathematical implementations. The two views on the same concepts that our comparison of two frameworks offers are meant to highlight what belongs to the abstract ideas behind what we essentially view as one single alternative, and what is due to particular realizations of the basic concepts in terms of different mathematical structures in two grammar frameworks.

Unification-based LTAG semantics (Kallmeyer and Romero 2007) and LRS (Richter and Sailer 2004) draw different conclusions from the tension between the expressivity needed in a feature logic for the specification of the combinatorics and representations of Ty2, and the requirements of effective computation. As one of the consequences, they use feature logics of different expressivity for similar purposes in semantic composition. Like Minimal Recursion Semantics (MRS, Copestake et al. 2005) they do not use the lambda calculus but the feature logic for the semantic combinatorics. In contrast to MRS, however, which focuses almost entirely on the design of semantic representations for large-coverage grammars without saying much about the interpretation of the derived representations, LTAG semantics and LRS subscribe to model-theoretic semantics and truth conditions specified in terms of Ty2. To overcome the tension between the demands on the expressivity of the feature logic and on the computational properties of the system, LTAG semantics and LRS choose different options. LTAG semantics combines a restricted feature logic with other mathematical structures that provide semantic representations and take over computational tasks. LRS relies on a uniform logical specification and a re-implementation of the LRS module of grammars in a computationally tractable constraint language. Important features of these two options of implementing constraint-based semantic composition will be worked out in the course of our discussion below.

In this paper we will investigate the two solutions which our two frameworks provide for integrating a model-theoretic semantics with syntactic structures using typed feature logics. We will focus on the technical choice points, identifying those properties of the two syntax-semantics interfaces which the two approaches have in common despite the technical differences, and those which differ due to different means of combining the syntactic and semantic module of grammar. This will also help distinguish fundamental properties of a feature logic-based syntax-semantics interface from accidental properties of a single system which are due to particular grammar architectures.

The paper is structured as follows: Sections 2 and 3 will lay the mathematical foundations for the comparison of the two frameworks. Section 2 will briefly introduce the most important logical properties of the HPSG framework, indicate how LRS can be specified in the same TFL as an HPSG grammar, and present the most important principles of LRS together with the analysis of a simple sentence. Section 3 is the counterpart of Section 2 for LTAG: A short summary of the mathematical architecture of LTAG is followed by an overview of the framework for LTAG with semantic unification and the role of TFL in this framework. The analysis of our simple example illustrates how the components of the theory interact. The next two sections are concerned with a direct comparison of specific crucial aspects of LRS and LTAG semantics. Section 4 focuses on the treatment of scope ambiguities and the role which TFL plays in their description. Section 5 shows how the differences in the application of semantic underspecification techniques in LRS and LTAG lead to different analyses of negative concord, an interesting linguistic phenomenon at the syntax-semantics interface. Section 6 turns to the conceptually important question of whether semantic systems whose combinatorics is based on feature logic are compositional. Although received opinion has it that they are not, Section 6 sketches a construction for LTAG that indicates that this might not be true. Section 7 concludes our investigation by a summary of the differences between the two systems and of the common properties of TFL-based semantic computation.

2 Lexical Resource Semantics

In the LRS architecture the feature logic may be used to specify the entire grammar, including well-formed Ty2 terms as semantic representations, and their mode of composition. This idea is particularly straightforward to implement in HPSG, since HPSG assumes an expressive feature logic as the single means of stating the entire grammar. While HPSG is by no means the only grammar architecture which can be combined with an LRS component, combining them is particularly simple, because the HPSG constraint language itself can be employed to specify the LRS structures. In this section we will explain how this can be achieved and what it means for an HPSG grammar with LRS semantics.

2.1 HPSG: Grammars as Logical Theories

From a mathematical point of view, an HPSG grammar is a logical theory consisting of a signature and a set of axioms. The purpose of the logical theory is to characterize all and only the grammatical linguistic structures of a natural language. The signature declares the non-logical symbols which the grammar writer may use, and it imposes certain structural conditions on interpretations of the grammar. Non-logical symbols of this kind of feature logic are sorts, attributes and relation symbols. The set of sorts is organized in a partial order, which is called the *sort hierarchy*. Examples of sorts are *sign*, *word* and *phrase*, and the HPSG sort hierarchy puts *word* and *phrase* below *sign*. HPSG's *signs* must have a SYNSEM attribute with values of sort *synsem*, *category* objects (which are found as values of the attribute CATEGORY) must have a HEAD attribute with a small set of possible values while they do not have a SYNSEM attribute, and so on. Typical HPSG relations are the binary relation `member` (for stating that some entity is on a list or in a set) and the ternary relation `append`, which is often used to state that the list-value of an attribute is obtained by appending the list-value of a second attribute to the list-value of a third attribute. The structural restrictions on interpretations that we have described above come from appropriateness conditions which declare certain attributes (such as SYNSEM) appropriate to certain sorts (such as *sign*) and prescribe the possible sort-values of these pairs (e.g., *synsem*).[1]

The statements of the logical theory (the axioms) are known to linguists as the principles of grammar. Their syntax uses the standard boolean logical connectives (conjunction, disjunction, negation, etc.), existential and universal quantification, and the attributes, sorts and relation symbols of the signature. The syntactic component of an HPSG grammar uses these symbols to state principles such as the HEAD FEATURE PRINCIPLE (the head value of a phrase and its head daughter are identical), the SUBCATEGORIZATION PRINCIPLE (regulating the discharge of arguments of a syntactic functor) or the IMMEDIATE DOMINANCE PRINCIPLE (playing the role of phrase structure rules of generative frameworks). We will not repeat any of these principles but illustrate HPSG's typed feature logic with a brief sketch of a TFL specification of LRS.

2.2 HPSG with an LRS Semantics

In a TFL specification of LRS, two components can be distinguished. First, we need to specify the syntax of the language of semantic representations, i.e., the language which we want to use to specify the meaning of linguistic signs. In previous work on LRS, this has always been Ty2 for compatibility with the semantics literature in linguistics, but many other logical languages are conceivable candidates without major changes to the overall architecture of LRS. Second, the combinatoric system must be specified. The combinatorics determines how the restrictions on semantic representations provided by syntactic daughters and their

1 Readers more familiar with algebraic specifications might note that sorts here are like their types, and attributes are like unary term constructors. There are no counterparts of constants in algebraic signatures. Thanks are due to an anonymous reviewer for pointing this out.

mode of combination as well as the nature of their syntactic mother determine the restrictions on the meaning of the phrase. It is this combinatoric system and the kind of structural information in syntax and semantic representations that it refers to which form the core of the LRS theory. In principle, any logical language strong enough to express these principles, and any syntactic theory which comprises the relevant syntactic structures can be used to specify a grammar with an LRS module.

The TFL specification of the syntax of Ty2 is very technical, and we do not need all of its details in the present context. To provide a general impression of how it works, Fig. 1 shows a fragment of a signature for Ty2.

```
ty2
  me   TYPE   type
    variable   NUM-INDEX   integer
    constant   NUM-INDEX   integer
    application   FUNCTOR   me
                  ARG       me
    abstraction   VAR    variable
                  BODY   me
    equation   ARG1   me
               ARG2   me
    negation   ARG   me
    generalized-quantifier   VAR     var
                             RESTR   me
                             SCOPE   me
      every
      some
      three
    logical-constant   ARG1   me
                       ARG2   me
      disjunction
      conjunction
      implication
      bi-implication
  ...
```

Figure 1
Fragment of the signature for a grammar of Ty2 expressions

The sort *ty2* subsumes all other sorts in the hierarchy, as indicated by indentation. Its most important subsort is *me (meaningful expression)*, with maximally specific subsorts for the logical constructs needed in the language. All expressions are typed, with the types encoded as values of an attribute TYPE. Variables and constants bear a natural number as their index, since our semantic representation language provides a countably infinite set of variables and constants of each type. For example, the 127th constant of type e is designated $\mathsf{const}_{\langle 126,e\rangle}$. In linguistic grammars, these constants are usually given more intuitive names. The constant

$\mathsf{const}_{\langle 126,e\rangle}$ might thus be referred to by the symbol $\mathsf{john'}$, and $\mathsf{const}_{\langle 126,\langle e,t\rangle\rangle}$ by $\mathsf{laugh'}$. Our signature fragment also includes sorts for three generalized quantifiers, *every*, *some* and *three*, which we will need in our linguistic examples. These generalized quantifiers can of course already be expressed with other basic constructions in our syntax for semantic representations, *viz.* variables, application, lambda abstraction and equations. Including them explicitly in our syntactic constraint language for Ty2 expressions will, however, turn out more than just a simple convenience when we formulate restrictions on the occurrence of (subsets of) quantificational expressions within certain structural domains. Further details of the signature of Ty2 such as the sort symbols for the encoding of integers and the type system are omitted from Fig. 1.

Of course, the signature alone does not guarantee the well-formedness of the expressions in the denotation of the grammar. To obtain this, we need a theory of the set of well-formed expressions of Ty2. (1) shows two of the necessary principles:[2]

$$\text{(1)}\quad \text{a.}\quad \textit{application} \rightarrow \begin{bmatrix} \textsc{type} & \boxed{2} \\ \textsc{functor type} & \begin{bmatrix} \textsc{in} & \boxed{1} \\ \textsc{out} & \boxed{2} \end{bmatrix} \\ \textsc{arg type} & \boxed{1} \end{bmatrix}$$

$$\text{b.}\quad \textit{equation} \rightarrow \begin{bmatrix} \textsc{type} & \textit{truth} \\ \textsc{arg1 type} & \boxed{1} \\ \textsc{arg2 type} & \boxed{1} \end{bmatrix}$$

(1a) requires that in an application, the argument be of a type that the functor can combine with, and the resulting type is determined by the functor. For example, if the functor is of type $\langle e, t\rangle$, it takes an argument of type $\langle e\rangle$ and yields an expression of type $\langle t\rangle$. (1b) says that an equation is of a truth type (i.e., true or false), and the two arguments of an equation are of the same type. More restrictions of this kind are needed for all logical connectives, as well as restrictions which guarantee the finiteness of Ty2 structures and the existence of a bijection of Ty2 expressions and the Ty2 structures in the denotation of the TFL specification of Ty2.[3] The full set of axioms needed in a TFL encoding of Ty2 can be found in (Penn and Richter, 2004, pp. 426–429). In order to avoid cumbersome notation, TFL descriptions of Ty2 expressions are typically avoided in specifications of LRS grammars. Instead it is common practice to write (partial) Ty2 expressions in TFL descriptions. It is important to keep in mind that this notation actually abbreviates descriptions of Ty2 expressions, and one such meta-expression may in fact describe an infinite number of Ty2 expressions. This should become clearer in our examples below.

2 In RSRL, tags are treated as variables, and all variables in grammar principles must be bound by a quantifier. By convention, if no quantifier binds a tag in a given principle, this tag is understood to be bound by an existential quantifier taking wide scope over the entire expression.
3 This means that Ty2 expressions are encoded by the grammar in such a way that for each Ty2 expression there is one class of isomorphic structures in the denotation of the grammar such that these structures correspond to the expression.

For our comparison with semantics in LTAG the specification of the combinatorial system of LRS is even more important than the exact details of a TFL encoding of Ty2. The main idea here is as follows: Signs refer to various aspects of their meaning in various feature values. The values of their LRS features restrict the meaning contribution to the utterances in which the signs may occur. Although it is very tempting at the beginning, the Ty2 values of LRS attributes such as EXCONT, INCONT and PARTS should thus not be understood as describing separate Ty2 expressions. It is more useful to think of the descriptions of these feature values in a sign as separate but interacting constraints on the possible meanings of the utterance to which the sign belongs. Metaphorically speaking, these restrictions are collected as we go up the syntactic tree until we have collected them all as restrictions on the EXCONT value of the overall utterance. LRS grammars are written in such a way that the EXCONT value of an utterance is a Ty2 expression which specifies the meaning of the utterance. When looking at each utterance in the denotation of an LRS grammar one discovers that in fact all LRS attributes of all signs in the utterance have values which are components of this EXCONT value. In other words, the descriptions of these feature values in the grammar turn out to be restrictions on the EXCONT value of the utterances predicted by the grammar.

The schematic description of signs in (2) reveals the main distinctions made in the feature geometry:

$$(2)\quad \begin{bmatrix} \textit{sign} & \\ \textsc{phonology} & \textbf{phonological structure} \\ \textsc{synsem} & \begin{bmatrix} \textsc{local} & \begin{bmatrix} \textsc{category} & \textbf{(local) syntactic structure} \\ \textsc{content} & \begin{bmatrix} \textit{content} & \\ \textsc{index} & \begin{bmatrix} \textit{ext-index} & \\ \textsc{var} & \textit{me} \\ \textsc{phi} & \textit{index} \end{bmatrix} \\ \textsc{main} & \textit{me} \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \textsc{lf} & \begin{bmatrix} \textit{lrs} & \\ \textsc{excont} & \textit{me} \\ \textsc{incont} & \textit{me} \\ \textsc{parts} & \textit{list(me)} \end{bmatrix} \end{bmatrix}$$

LRS distinguishes between local (lexically oriented) and non-local (combinatorial) aspects of the semantics of signs. The local aspects can be selected by syntactic functors and are located under SYNSEM. More precisely, they are under the attributes VAR and MAIN, which are both located at the traditional place for semantic representations in HPSG, the CONTENT value.

The attributes which are responsible for building the semantic representations of phrases from the semantics of their daughters are under a new attribute LOGICAL-FORM (LF), which is not accessible for selectional restrictions since it is appropriate to the sort *sign*. Three combinatorial aspects of the semantic representation of a sign are identified: The external content (under EXCONT) is the semantic contribution which a sign makes at its highest syntactic projection to the overall utterance in which it occurs; the internal content (under INCONT) is that part of the semantic representation of a sign which is within the scope of any operator the sign combines with; the PARTS list marks those pieces of the semantic representations connected to words which count as being contributed to the utterance in which the word occurs.[4] We observe that every subterm of the meaning representation of an utterance must be introduced as an element on the PARTS list of (at least) one word in the utterance.[5] Conversely, the meaning representation of an utterance must contain all elements on all PARTS lists of all words in the utterance. Intuitively speaking, the meaning of an utterance consists precisely of those sub-expressions which come from the words in it. Nothing can be added from outside, and nothing gets lost.

The function of the new attributes is best understood by considering the semantic analysis of a few words in LRS:

(3) a. John:

$$\begin{bmatrix} \text{PHON} & \langle \textit{john} \rangle \\ \text{SS LOC} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \textit{noun} \\ \text{SUBCAT} & \langle\rangle \end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \text{INDEX VAR} & \boxed{1}\ \mathsf{john'} \\ \text{MAIN} & \boxed{1}\ \mathsf{john'} \end{bmatrix} \end{bmatrix} \\ \text{LF} & \begin{bmatrix} \text{EXCONT} & \textit{me} \\ \text{INCONT} & \boxed{1}\ \mathsf{john'} \\ \text{PARTS} & \langle \boxed{1}\ \mathsf{john'} \rangle \end{bmatrix} \end{bmatrix}$$

b. laughs:

$$\begin{bmatrix} \text{PHON} & \langle \textit{laughs} \rangle \\ \text{SS LOC} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \textit{verb} \\ \text{SUBCAT} & \langle \text{NP}_{\boxed{1}} \rangle \end{bmatrix} \\ \text{CONT MAIN} & \boxed{2a}\ \mathsf{laugh'} \end{bmatrix} \\ \text{LF} & \begin{bmatrix} \text{EXCONT} & \textit{me} \\ \text{INCONT} & \boxed{2}\ \mathsf{laugh'}(\boxed{1}) \\ \text{PARTS} & \langle \boxed{2}\ \mathsf{laugh'}(\boxed{1}),\ \boxed{2a}\ \mathsf{laugh'} \rangle \end{bmatrix} \end{bmatrix}$$

4 Readers familiar with the development of HPSG might recall that Kasper (1997) used attributes called ECONT and ICONT in an analysis of recursive modification which solved certain problems with Pollard and Sag's original proposal for analyzing the semantics of modifiers in HPSG. The attribute names are adapted from Kasper by LRS to acknowledge the inspiration Kasper's paper gave for distinguishing between constituent-internal and external content. However, the two approaches differ significantly in detail, and this is not the place for a comparison.

5 As we will see in Section 5, there are special cases in which at least one of the meaning contributions of two (or more) words may be identical. In particular, the single sentential negation in negative concord constructions may come from several n-words and the negative marker in a sentence.

c. always:

$$\begin{bmatrix} \textsc{phon} & \langle \textit{always} \rangle \\ \textsc{ss loc} & \begin{bmatrix} \textsc{cat} & \begin{bmatrix} \textsc{head} & \begin{bmatrix} \textit{adv} \\ \textsc{mod}\ \mathrm{V}[\textsc{loc cont main}\ \boxed{2a}\,] \end{bmatrix} \\ \textsc{subcat} & \textit{elist} \end{bmatrix} \\ \textsc{cont main} & \boxed{5a}\ \mathsf{always}' \end{bmatrix} \\ \textsc{lf} & \begin{bmatrix} \textsc{excont} & \textit{me} \\ \textsc{incont} & \boxed{5}\ \mathsf{always}'(\boxed{3}) \\ \textsc{parts} & \langle \boxed{5}\ \mathsf{always}'(\boxed{3}),\ \boxed{5a}\ \mathsf{always}' \rangle \end{bmatrix} \end{bmatrix}$$

$\&\ \boxed{2a} \triangleleft \boxed{3}$

The analysis of proper names such as *John* (3a) is particularly simple. Since proper names are analyzed as contributing only a non-logical constant (john') to the semantic representation, the INDEX VAR value, the MAIN value, the internal content and their semantic contribution (on PARTS) are all identical. Note that the four attributes refer to the very same symbol in the denotation of the TFL specification, as indicated by the tag $\boxed{1}$. The external content of the word *John* is not lexically determined. The EXCONT value *me* (meaningful expression) indicates that any Ty2 expression is permitted. Only when the word is combined with a functor will other principles fix the EXCONT value (which will also be john').

Verbs such as *laughs* are more interesting. Its local MAIN value is laugh', indicating the lexical meaning of the word. Its internal content is the application of the predicate laugh' to a lexically underspecified constant or variable of the appropriate type. Which constant or variable it is will be determined by the subject NP, whose VAR value, $\boxed{1}$, contributes the relevant logical argument.[6] From the perspective of semantic contributions to the utterance in which it occurs, *laughs* provides the application which we already saw as the INCONT value, and the non-logical constant laugh'.[7] The PARTS list of *laughs* does not contain $\boxed{1}$, since the verb does not contribute the relevant expression of type *e* to the meaning of the utterance in which it occurs. This expression is contributed by the subject.

The subject NP of *laughs* is syntactically selected by the verb as the first element on its SUBCAT list. The notation 'NP', employed here to describe the first element on the SUBCAT list of *laughs*, is a frequent abbreviation in the attribute-value matrix (AVM) notation of HPSG descriptions. It describes *synsem* objects with a saturated (empty) SUBCAT lists and HEAD value *noun*. Elsewhere we will use similar standard abbreviations for verbal *synsems* (V, VP) and adjunct *synsems* (A).

The description of the adverb *always*, (3c), introduces another important type of constraint on semantic representations, component-of constraints. The analy-

6 The identity requirement between the VAR value of the syntactically selected argument and the logical argument of laugh' must in fact be relaxed when we extend the analysis to arguments that are definite descriptions of type *e* such as *the student*, and semantically similar constructions. See Sailer 2004 for the relevant generalization in terms of a component-of constraint. For our present purposes, the simpler identity requirement will suffice.

7 For the purposes of the present paper, we ignore the INDEX VAR value of verbs, for which Richter and Sailer (2004) propose event variables.

sis of *always* resembles the analysis of *laughs*, except that *always* as an adverb selects its argument via the MOD attribute instead of a SUBCAT list. The selected argument, $\boxed{2a}$, of the operator always′ is, however, not analyzed as an immediate argument. Instead the lexical entry requires that $\boxed{2a}$ be a component of the argument, $\boxed{3}$, of *always*. If nothing else intervenes and the type of the MAIN value of the selected argument were appropriate, $\boxed{2a}$ could be identical to $\boxed{3}$ in a given sentence as far as the component-of constraint is concerned. As this example shows, component-of constraints are used for saying that (i) one expression belongs in an argument slot, or (ii) is in the scope of another expression. However, we usually do not know whether the first expression combines with something else before it fits into the relevant argument slot in (i), or whether we face a relationship of immediate scope in (ii). Moreover, type clashes might force the first expression to be combined with something else first before it fits into the alloted slot.

Two things deserve to be pointed out about the lexical entry in (3c): The typing of the always′ operator as a predicate taking a truth value is, of course, an oversimplification and only meant as an illustration of the guiding ideas.[8] Secondly, note that the type of laugh′ ($\langle e, t\rangle$) and the subterm requirement of (3c) suffice to guarantee that laugh′ has to apply to its argument first so as to fit into the argument slot of always′. Type restrictions of this kind play a very important role in the use of underspecification in LRS.

Before we can analyze sentences, we need to introduce the most important LRS principles, the INCONT PRINCIPLE, the EXCONT PRINCIPLE and the LRS PROJECTION PRINCIPLE. They are listed in (4). For simplicity, we assume binary branching structures throughout this paper.

(4) a. The INCONT PRINCIPLE:
In each *lrs*, the INCONT value is an element of the PARTS list and a component of the EXCONT value.

$$lrs \longrightarrow \left(\begin{bmatrix} \text{EXCONT} & \boxed{1} \\ \text{INCONT} & \boxed{2} \\ \text{PARTS} & \boxed{3} \end{bmatrix} \wedge \ \texttt{member}(\boxed{2}, \boxed{3}) \wedge \ \boxed{2} \triangleleft \boxed{1} \right)$$

b. The EXCONT PRINCIPLE:

Clause (a):
In every phrase, the EXCONT value of the non-head daughter is an element of the non-head daughter's PARTS list.

$$phrase \longrightarrow \left(\begin{bmatrix} \text{NH-DTR LF} & \begin{bmatrix} \text{EXCONT} & \boxed{1} \\ \text{PARTS} & \boxed{2} \end{bmatrix} \end{bmatrix} \wedge \ \texttt{member}(\boxed{1}, \boxed{2}) \right)$$

Clause (b):
In every utterance, every subexpression of the EXCONT value of the

8 For a real semantic analysis compatible with the present LRS framework, one could follow the LTAG proposal of Kallmeyer and Romero (2007).

utterance is an element of its PARTS list, and every element of the utterance's PARTS list is a subexpression of the EXCONT value.

u-sign $\longrightarrow$

$$\forall\boxed{1}\,\forall\boxed{2}\,\forall\boxed{3}\,\forall\boxed{4}\left(\begin{array}{l}\left(\left[\text{LF}\begin{bmatrix}\text{EXCONT} & \boxed{1}\\ \text{PARTS} & \boxed{2}\end{bmatrix}\right] \wedge \boxed{3} \triangleleft \boxed{1} \wedge \texttt{member}(\boxed{4},\boxed{2})\right) \rightarrow \\ (\texttt{member}(\boxed{3},\boxed{2}) \wedge \boxed{4} \triangleleft \boxed{1})\end{array}\right)$$

c. LRS PROJECTION PRINCIPLE:

In each *phrase*,
1. the EXCONT values of the head and the mother are identical,

$$\textit{phrase} \longrightarrow \begin{bmatrix}\text{LF EXCONT} & \boxed{1}\\ \text{H-DTR LF EXCONT} & \boxed{1}\end{bmatrix}$$

2. the INCONT values of the head and the mother are identical,

$$\textit{phrase} \longrightarrow \begin{bmatrix}\text{LF INCONT} & \boxed{1}\\ \text{H-DTR LF INCONT} & \boxed{1}\end{bmatrix}$$

3. the PARTS value contains all and only the elements of the PARTS values of the daughters.

$$\textit{phrase} \longrightarrow \left(\begin{bmatrix}\text{LF PARTS} & \boxed{1}\\ \text{H-DTR LF PARTS} & \boxed{2}\\ \text{NH-DTR LF PARTS} & \boxed{3}\end{bmatrix} \wedge \texttt{append}(\boxed{2},\boxed{3},\boxed{1})\right)$$

(4a) requires that the part of the meaning of a sign which is outscoped by everything else (the internal content) is actually contributed by the sign itself and is a subterm of its external content. The external content is governed by two principles which have to do with maximal projections. According to ((4b), Clause a) the maximal projection of a sign (identified as the non-head daughter of an embedding sign) must be a contributor of its external content. In other words, the external content must originate from within a maximal projection, it cannot come from outside. ((4b), Clause b) is a closure principle. Every sign in the language is a daughter of one unique unembedded sign or utterance. The closure principle says that the meaning of an utterance (its external content) consists of all and only those symbols and ways of combining symbols (by application and abstraction) which are contributed by the signs in the utterance. The projection principle (4c) makes sure that internal and external contents are identical along syntactic head projections and the contributions to the semantic representations of all daughters are collected in the PARTS lists of the mother nodes.

With the lexical entries and the core LRS principles we can already derive the semantic representation of a simple sentence. The analysis of (5) is shown in Fig. 2.

(5) John always laughs.

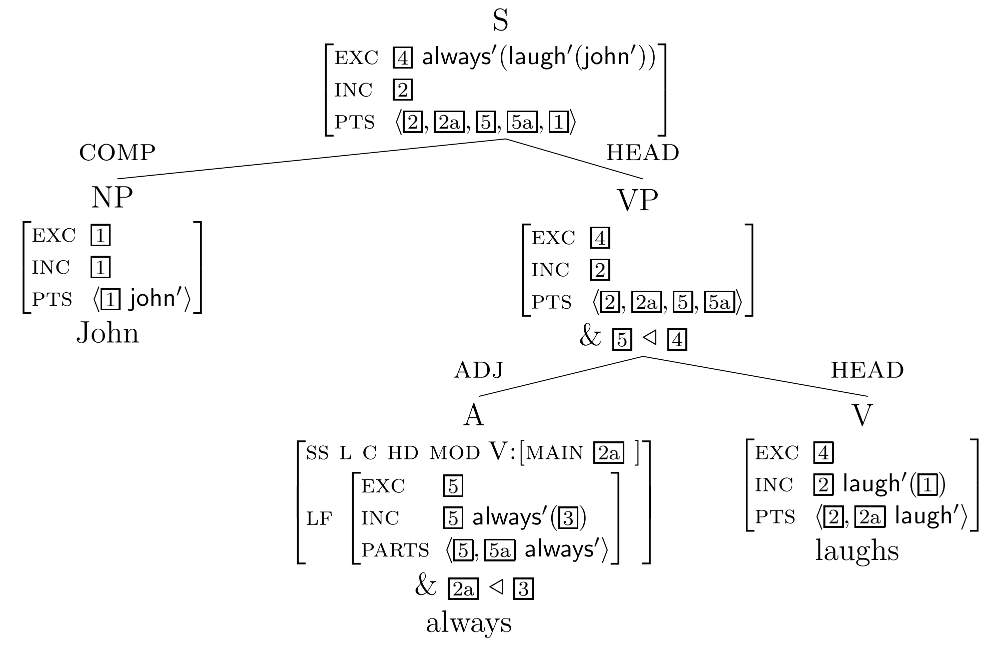


Figure 2
LRS analysis of *John always laughs*

Each word specifies its contribution to the overall meaning of the sentence (PARTS), the part of its semantics which is outscoped by all signs it combines with (INCONT), and the overall semantic contribution of its maximal projection (EXCONT). The feature percolation mechanism introduced by the LRS PROJECTION PRINCIPLE identifies INCONT and EXCONT along head projections and collects the elements of the PARTS lists of the daughters at each phrase. The combination of the adjunct with a verbal projection induces a number of restrictions: Since each non-head daughter's EXCONT must be on its PARTS list (EXCONT PRINCIPLE, Clause a) and the INCONT must be a component of the EXCONT (INCONT PRINCIPLE), the EXCONT of *always* must equal the INCONT of *always*. Moreover, the EXCONT of *always* must be within the EXCONT of *laughs* (EXCONT PRINCIPLE, Clause b). Next, the INCONT of *laughs* must be in the scope of *always* according to the component-of constraint in the lexical entry of *always*, (3c). The semantic argument of *laughs*, john′, is identified by subcategorization, as indicated in the lexical entry of *laughs* ((3b), tag $\boxed{1}$). The closure condition of the EXCONT PRINCIPLE ((4b), Clause b)) requires that the semantic representation of an utterance use up all and only the PARTS contributions of all signs, which finally yields the semantic representation spelled out as the EXCONT description of the S node in Fig. 2. As the reader may verify, this expression is the only solution of the combined constraints on the semantic representation of the overall sentence. These constraints come from the lexical entries as well as from syntactic properties of the sentence.

2.3 Summary: The Architecture

In this section, we showed that LRS can be integrated seamlessly with the logical architecture of HPSG. There is no distinction between the TFL specification of syntactic structures and the TFL specification of semantic representations which carry the truth conditional meaning of the entire structure. Syntactic and semantic constraints can interact freely. It is an important feature of this architecture that underspecification is a matter of the TFL level of grammatical descriptions. The structures in the denotation of the grammar are complete structures, including the semantic representations. This means that there is no semantic underspecification in the denotation of the grammar.

It is not necessary for a grammar with an LRS semantics that its constraints are expressed in the same language as the rest of the grammar. An example of a system which uses distinct languages is the computational implementation of LRS in the TRALE system described in Penn and Richter 2004. The *Constraint Language for Lexical Resource Semantics* (CLLRS) is a specialized constraint language designed to facilitate the notation of the constraints and to eliminate the technical overhead caused by the TFL encoding of the syntax of Ty2. In CLLRS the well-formedness of the semantic representations is guaranteed by an independently implemented set of well-formedness axioms that the user does not have to declare or even know (Penn and Richter 2005). Despite the different approaches in CLLRS and the LRS architecture presented in Subsection 2.2, CLLRS maintains the tight connection to syntactic structures. It supports constraints in which semantic inferences are grounded in syntactic structure, and semantic structure may trigger syntactic constraints. All properties of LRS relevant in our comparison between LTAG semantics and LRS are preserved in CLLRS. However, the comparison of LTAG semantics and LRS is more transparent when we can refer to a single typed feature logic with a uniform model theory which is responsible for semantic representations and for syntactic structure simultaneously.

3 Lexicalized Tree Adjoining Grammars and Semantic Unification

In contrast to the LRS integration with HPSG using one single TFL, LTAG is characterized by a modular architecture, where the feature logic is used solely for semantic computation and nothing else. The basis is a syntactic tree-generating formalism with a limited generative capacity. The trees from the grammar are linked to semantic representations that are sets of Ty2 formulas that need to be put together. The way these formulas are combined in order to obtain the meaning of natural language expressions involves two further characteristics of the grammar architecture of LTAG semantics: First, the semantic representations are linked to feature structure descriptions, which encode 1. the arguments needed to complete the formulas and 2. the values provided as possible arguments for other formulas. Depending on the tree combination operations performed in the syntax, feature value equations are computed between the different feature structure descriptions, at which point some of the open argument slots in the semantic formulas are filled. Second, the result of this process is still an underspecified representation,

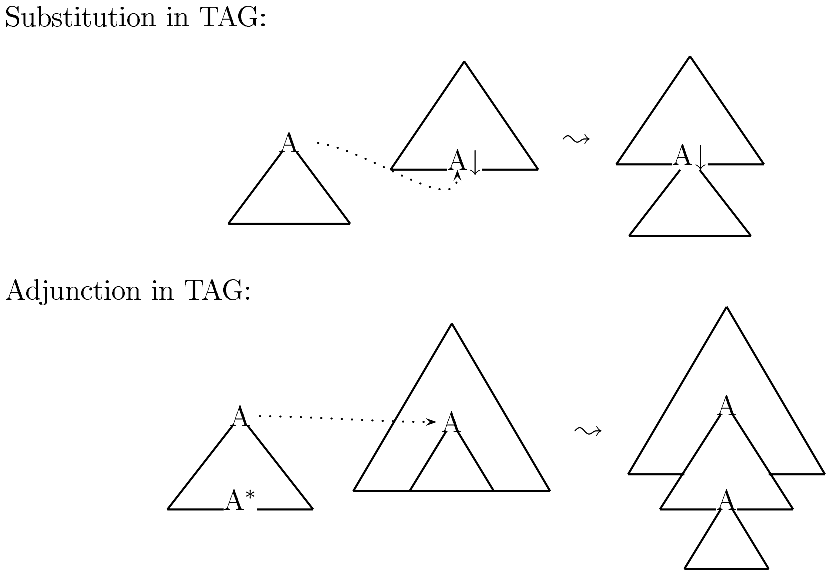


Figure 3
Substitution and adjunction in TAG

similar to the ones proposed in Hole Semantics (Bos 1995) and Minimal Recursion Semantics (MRS, Copestake et al. 2005). In order to obtain the final meanings of a sentence one has to compute the different disambiguations of this representation.

In the next sections we will describe the components of this system.

3.1 Lexicalized Tree Adjoining Grammars

LTAG (Joshi and Schabes 1997) is a tree-rewriting formalism. An LTAG consists of a finite set of *elementary* trees associated with lexical items. From these trees, larger trees are derived by substitution (replacing a leaf with a new tree) and adjunction (replacing an internal node with a new tree). The two operations are depicted in Fig. 3. In case of an adjunction, the new tree, called an *auxiliary* tree, has a special leaf node, the *foot node* (marked with an asterisk). LTAG requires the nodes involved in these operations to be labelled with the same non-terminal symbols (A in Fig. 3). When adjoining a tree to a node μ, in the resulting tree, the subtree with root μ from the old tree is put below the foot node of the auxiliary tree. Non-auxiliary elementary trees are called *initial* trees. Each derivation starts with an initial tree.

The elementary trees of an LTAG represent extended projections of lexical items and encapsulate all syntactic/semantic arguments of the lexical anchor. They are minimal in the sense that only the arguments of the anchor are encapsulated, all recursion is factored out. These linguistic properties are formulated in the *Condition on Elementary Tree Minimality (CETM)* in Frank (2002).

A crucial property of LTAG is its *extended domain of locality.* The recursive material that is factored out is put in separate auxiliary trees that can be adjoined.

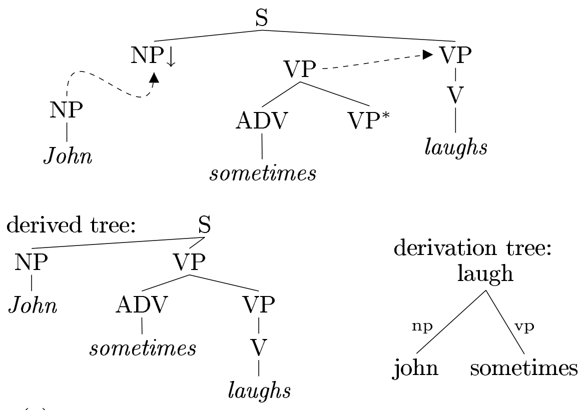


Figure 4
TAG derivation for (7)

As a consequence, in the final derived tree, the contribution of an elementary tree can be separated into different parts that might be far away from each other. For example, in a long-distance dependency such as (6), the slot for the wh-word and the verb *marry* are in the same elementary tree while the trees for *wants*, *said* and *think* all adjoin to the S node in the middle.

(6) Who does John think Bill said Mary wants to marry?

LTAG derivations are represented by derivation trees that record the way the elementary trees are put together. A derived tree is the result of carrying out the substitutions and adjunctions. Each edge in the derivation tree stands for an adjunction or a substitution. The edges are equipped with addresses of the nodes where the substitutions/adjunctions take place. The derivation of (7) in Fig. 4 illustrates this: Starting from the elementary tree of *laugh*, the tree for *John* is substituted for the node at position *np* and *sometimes* is adjoined at position *vp*.

(7) John sometimes laughs.

In contrast to the logical foundations underlying HPSG, LTAG has very limited generative capacity; it belongs to the class of mildly context-sensitive grammar formalisms (Joshi 1985) and only slightly extends the generative capacity of context-free grammars (CFG). This explains why LTAG has attractive formal properties; it is polynomially parsable (see among others Schabes and Joshi 1988; Vijay-Shanker and Weir 1993; Nederhof 1997), tree adjoining languages (TAL) have desirable closure properties (Vijay-Shanker and Joshi 1985; Vijay-Shanker 1987), there is a pumping lemma for TALs (Vijay-Shanker 1987), and there is an extension of pushdown-automata that accepts TALs (Vijay-Shanker 1987). In general, the use of LTAG for natural languages is motivated on the one hand by the fact that CFGs are not powerful enough to describe all natural language phenomena (Shieber 1985) and on the other hand by the desire to stay as close as possible to CFG in terms of complexity and generative capacity.

3.2 LTAG Semantics with Semantic Unification

In the LTAG semantics approach we consider here (see Kallmeyer and Romero 2007), each elementary tree in the Tree Adjoining Grammar (TAG) is linked to a pair consisting of a semantic representation and a semantic feature structure description. These feature structure descriptions are used to compute assignments for variables in the representations using conjunction and additional equations introduced depending on the derivation tree.

3.2.1 Semantic Representations and Semantic Feature Structure Descriptions As in Kallmeyer and Joshi 2003, we use flat semantic representations in the style of MRS (Copestake et al. 2005): Semantic representations consist of a set of labelled Ty2 formulas and a set of scope constraints. A scope constraint is an expression $x \geq y$ where x and y are propositional labels or propositional meta-variables (these correspond roughly to holes in Bos 1995).

The formulas in a semantic representation contain meta-variables – depicted as boxed Arabic numbers, e.g. $\boxed{1}$ – of type e (individuals), s (situations) and $\langle s, t\rangle$ (propositions).[9] Each semantic representation is linked to a semantic feature structure description which can include the meta-variables from the formulas. Between the descriptions, feature value equations are computed depending on the derivation tree. From the descriptions and these additional feature equations, which are interpreted conjunctively, assignments can be inferred for some of the meta-variables in the semantic representations.

As an example consider the semantic representation and the semantic feature structure of *laughs* in Fig. 5. The fact that the meta-variable of the argument of laugh' appears in the top (T) feature of the subject NP node position NP indicates that this argument will be obtained from the semantics of the tree substituted at the subject node. The label of the laugh' proposition, l_1, is linked to the bottom of the VP node. This signifies that the proposition l_1 is the minimal proposition corresponding to this node. If an adverb adjoins at the VP node, l_1 will be embedded under that adverb, and the larger proposition $\boxed{4}$ will be provided by the adverb. Note that the variables in the description, e.g., $\boxed{4}$ in this example, need not occur in the semantic representation.

$$\boxed{l_1 : \mathsf{laugh}'(\boxed{1})} \quad \begin{bmatrix} \textsc{np} & \begin{bmatrix} \textsc{t} & \begin{bmatrix} \textsc{i} & \boxed{1} \end{bmatrix} \end{bmatrix} \\ \textsc{vp} & \begin{bmatrix} \textsc{t} & \begin{bmatrix} \textsc{p} & \boxed{4} \end{bmatrix} \\ \textsc{b} & \begin{bmatrix} \textsc{p} & l_1 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figure 5
Semantic representation and semantic feature structure description of *laughs*

Since the focus of this paper is on the use of feature logics for computational semantics, let us say a little more about the semantic feature structure descriptions

9 We take the term "situation" to be more general than "worlds": worlds are considered to be special kinds of situations, namely maximally specified situations.

in LTAG. The feature logic descriptions serve the sole purpose of putting semantic representations together; they are a kind of glue. In principle, they could be defined either as partial feature structures with unification or as feature structure descriptions with conjunctions and equalities. We chose the latter option since this allows us to use a simple inference mechanism for calculating assignments for the meta-variables in our semantic representations.

Our semantic feature structures as well as the corresponding terms of the feature logic are typed. We will call the feature structure types *fs-types* to distinguish them from the types of the terms in the semantic representations. The whole feature structure that goes with an elementary tree is of fs-type *sem* and has attributes NP, VP, etc. for all node positions occurring in the elementary trees of the TAG (finite for each TAG) whose values are of fs-type *tb* (for "top-bottom"). These in turn have attributes T and B whose values are of fs-type *bindings* and have attributes I, P, S (for "individual", "proposition" and "situation") with values of fs-types var_e (these are the variables of type e from our Ty2 language), $var_{\langle s,t\rangle}$ (the labels of propositional type), and var_s (the variables of type s from our Ty2 language) respectively.

The fs-types do not have a hierarchical structure. In other words, there are no sub-types (no sort hierarchy) as in the feature logic of HPSG (including LRS).[10]

The intuition behind our typed feature structures is the following: A semantic feature structure description links individuals, situations and propositions to syntactic positions, i.e., to nodes in the (syntactic) elementary tree. Each node has a top and a bottom feature structure. If no substitution or adjunction occurs at a node, top and bottom get identified. Otherwise, they can have different values.

The feature structure descriptions linked to the semantic representations are simple first order formulas with attributes and with constants for values of atomic fs-type, similar to those introduced by Johnson (1988, 1990). The main difference is that our logic is typed, thus we do not need a symbol $\perp$ for undefined values. We avoid computing potentially undefined values by typing our feature terms and defining terms in such a way that attributes are applied only to terms of appropriate fs-types. The logic we are using is only a fragment of first order logic since we need neither negation nor disjunction or universal quantification.

We will use fs-variables $\boxed{0}, \boxed{1}, \ldots$. Feature structure descriptions will be given in the usual AVM notation. An example is provided in Fig. 6, where all conjuncts have a complex fs-term of the form $a(u)$ equated with a simple fs-variable or fs-constant.

3.2.2 Semantic Composition Semantic composition consists of conjoining feature structure descriptions while adding further feature value equations. It corresponds to the feature unifications in the syntax in Feature Structure-based TAG (FTAG, Vijay-Shanker and Joshi 1988) that are performed during substitutions,

10 Actually, one might do without fs-types. Attributes will then denote partial functions and, in the terms of our feature logic, they might occur in places where they do not make sense because their value is undefined. In our logic, the typing disallows building such terms.

Feature structure description:
$\text{I}(\text{T}(\text{NP}(\boxed{0}))) = \boxed{1}\wedge$
$\text{P}(\text{T}(\text{VP}(\boxed{0}))) = \boxed{4}\wedge$
$\text{P}(\text{B}(\text{VP}(\boxed{0}))) = l_1$

Corresponding avm:

$$\boxed{0}\begin{bmatrix} \text{NP} & \begin{bmatrix}\text{T} & \begin{bmatrix}\text{I} & \boxed{1}\end{bmatrix}\end{bmatrix} \\ \text{VP} & \begin{bmatrix} \text{T} & \begin{bmatrix}\text{P} & \boxed{4}\end{bmatrix} \\ \text{B} & \begin{bmatrix}\text{P} & l_1\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

Figure 6
Feature structure description in AVM notation

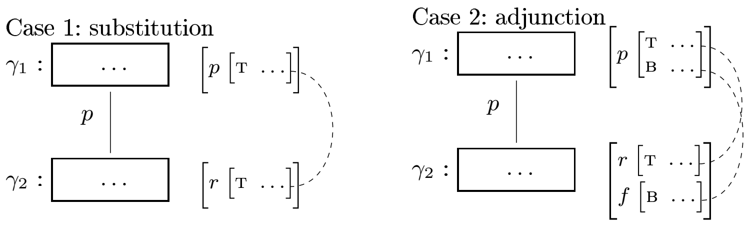


Figure 7
Feature identifications depending on substitutions and adjunctions

adjunctions and the final top-bottom unifications in the derived tree.

In the derivation tree, elementary trees are replaced by their semantic representations plus the corresponding semantic feature structure description. (See the derivation tree on the left and the structure for computing semantics on the right in Fig. 8.) We assume that each time a new elementary semantic entry is chosen from the grammar, it contains fresh instances of labels, individual and situation variables and meta-variables. This way, the sets of labels and variables occurring in different nodes of the derivation tree are pairwise disjoint.

The additional feature equations added at substitution or adjunction edges in the derivation tree are depicted schematically in Fig. 7. They are specified as follows: For each edge in the derivation tree from γ_1 to γ_2 with position p:

- The top feature of position p in γ_1 and the top feature of the root position in γ_2, i.e., the features $\gamma_1.p.\text{T}$ and $\gamma_2.r.\text{T}$ are equated (where r is the root node position),
- and if γ_2 is an auxiliary tree, then the bottom feature of the foot node of γ_2 and the bottom feature of position p in γ_1, i.e., the features $\gamma_1.p.\text{B}$ and $\gamma_2.f.\text{B}$ are equated (where f is the position of the foot node in γ_2).

Furthermore, for all γ in the derivation tree and for all positions p in γ such that there is no edge from γ to some other tree with position p, the T and B features of $\gamma.p$ are equated.

As an example consider the analysis of (7): Fig. 8 shows the derivation tree with the semantic representations and the semantic feature structure descriptions of the three elementary trees involved in the derivation. The formula $\mathsf{john}'(x)$ is in-

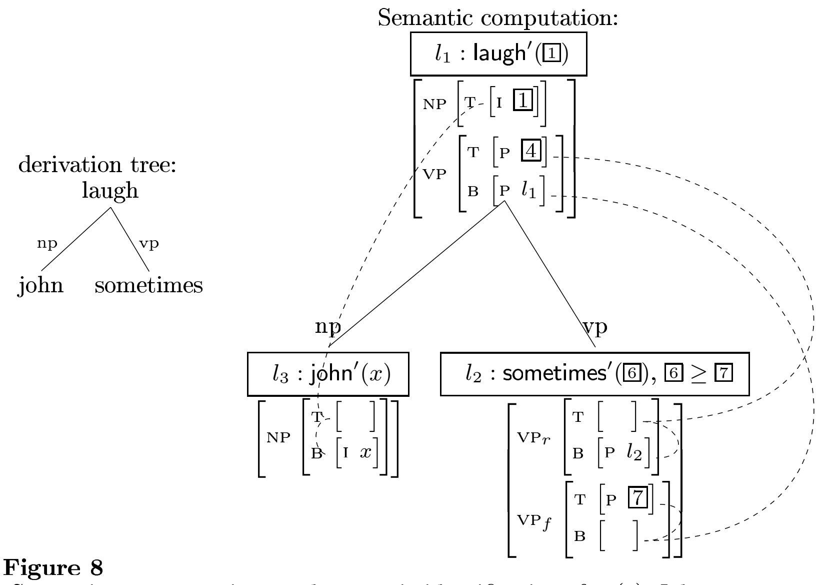


Figure 8
Semantic representations and semantic identifications for (7) *John sometimes laughs*

terpreted as meaning "there is a unique individual *John* and x is this individual".[11] *Sometimes* scopes over a proposition $\boxed{6}$ containing at least $\boxed{7}$ and contributes a new proposition l_2.

The feature value identifications lead to the identities marked in Fig. 8 with dotted lines. The top of the subject NP of *laughs* is identified with the top of the root NP of *John* (substitution) and with the bottom of the root of *John* (final top-bottom unification). Consequently $\boxed{1} = x$. The bottom of the VP in *laughs* is identified with the bottom and top of the foot VP$_f$ of *sometimes* (adjunction and final top-bottom unification), yielding $\boxed{7} = l_1$. Finally, the top of the VP in *laughs* is identified with the top and bottom of the root VP$_r$ of *sometimes* (again, adjunction and final top-bottom unification), with the result $\boxed{4} = l_2$.

Equality between fs-terms is reflexive, symmetric and transitive, and it extends to the different attributes allowed for the fs-type of the term. This permits us to derive further conjuncts using corresponding inference rules. Conjoining the different feature descriptions on the derivation tree, the new feature equations and the further conjuncts one can derive yields a large description δ.

If δ is satisfiable, we can continue computing an assignment function from δ. In order to check for satisfiability, we have to check whether for all fs-constants c_1, c_2 with $\delta \vdash c_1 = c_2$, c_1 is indeed equal to c_2. Then, from δ an assignment function g can be obtained for some of the meta-variables occurring in the semantic representations. We assume that the meta-variables are alphabetically ordered. Then g is defined as follows:

11 This is similar to the treatment of proper nouns in DRT of Kamp and Reyle (1993).

- for all fs-variables $\boxed{n}$ such that there is a fs-constant c with $\delta \vdash \boxed{n} = c$: $g(\boxed{n}) = c$,
- for all fs-variables $\boxed{n_1}$ such that there is no fs-constant c with $\delta \vdash \boxed{n_1} = c$: if $\boxed{n_2}$ is the alphabetically first fs-variable such that $\delta \vdash \boxed{n_1} = \boxed{n_2}$, then $g(\boxed{n_1}) = \boxed{n_2}$.

This assignment is then applied to the semantic representation and the union of the representations is built. In our example this leads to (8):

(8) $\boxed{l_1 : \mathsf{laugh}'(x), l_2 : \mathsf{sometimes}'(\boxed{6}), l_3 : \mathsf{john}'(x), \boxed{6} \geq l_1}$

3.2.3 Disambiguation The semantic representation obtained in the way described above is usually underspecified and cannot be interpreted yet. First, appropriate disambiguations must be found. These are assignments for the remaining meta-variables, i.e., functions that assign propositional labels to propositional meta-variables respecting the scope constraints. The definition of these disambiguations roughly corresponds to the possible pluggings in Bos (1995). The disambiguated representation is then interpreted conjunctively.

(8) has only one disambiguation, $\boxed{6} \to l_1$, since $\boxed{6}$ cannot possibly equal l_2 ($\boxed{6}$ must be in the scope of l_2) and $\boxed{6}$ cannot possibly equal l_3 (otherwise there would be no meta-variable left below $\boxed{6}$ to be equated with l_1 in order to satisfy the constraint $\boxed{6} \geq l_1$).

(9) $\mathsf{john}'(x) \wedge \mathsf{sometimes}'(\mathsf{laugh}'(x))$

3.3 Computational aspects

The computation of the underspecified semantic representation via feature identification on the derivation tree uses the same mechanisms as ordinary LTAG parsing in a feature structure-based TAG (FTAG, Vijay-Shanker and Joshi 1988). The only difference is that the set of possible feature values is not finite in general (e.g., possible values for features of propositional type are $l_1, l_2, l_3, \ldots$). However, in practical applications, the feature value set can always be limited. Then the complexity of syntactic FTAG parsing (which is $O(n^6)$) and FTAG parsing including semantics is the same except for some constant factors. There exists already an extension of an LTAG parser that takes into consideration semantics, assuming a syntax-semantics interface very similar to the one presented here. It outputs the TAG derivation trees and the underspecified semantic representations computed on these derivation trees (SemConst, Gardent and Parmentier 2005).

The second aspect to consider is the complexity of the disambiguation, i.e., the computation of the different readings that an underspecified representation yields. In general, the complexity of disambiguating expressions with scope constraints of the form $x \geq y$ is NP-complete (Koller et al., 2001). But the semantic representations we actually use are close to so-called *normal dominance constraints* (Koller et al., 2003; Fuchss et al., 2004). For this type of constraints an efficient polynomial solver has been developed.

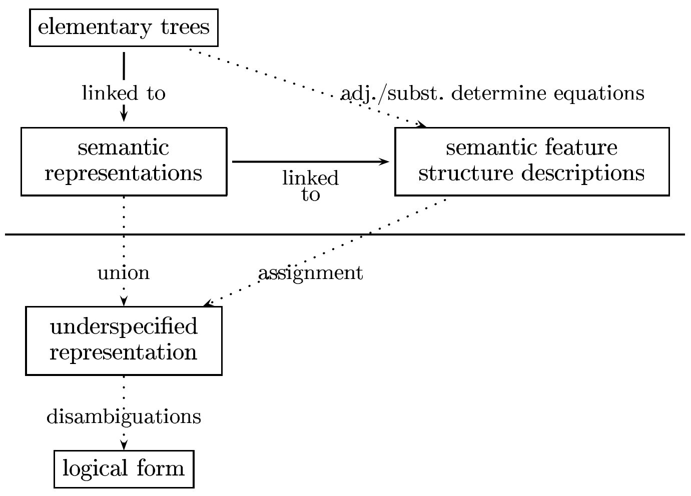


Figure 9
Components of the LTAG system

3.4 Summary: The Overall Architecture of LTAG

As already mentioned, in contrast to LRS, LTAG is a very modular system. The different components and their relations are summarized in Fig. 9. The parts above the horizontal line are components of the lexicon while the parts below this line are generated in the course of the derivation. The interface structure between syntax and semantics is the derivation tree; it determines locally both the compositions of elementary trees and the equations between feature structure descriptions.

Let us emphasize once again the difference in the use of feature logics: In LRS, a powerful feature logic is used to specify the whole grammar. In LTAG, a simple feature logic (a fragment of first order logic) is used to specify the argument requirements/contributions of semantic representations and, as we will see later, also to specify scope boundaries. In LRS the identities between feature values stemming from the feature value equations of different lexical entries arise from general principles which of course are also defined using the feature logic. In LTAG even these equations are not part of the TFL descriptions of the grammar writer. Rather, they arise from an extra feature identification mechanism defined on the derivation tree.

4 Expressing Scope Boundaries with Features

In this section we will compare the ways in which the two frameworks model the scopal behavior of quantificational NPs, adverbs and modal verbs. We focus on the similarities arising from the use of feature logics to encode scope boundaries, and we ignore a number of subtle differences in the linguistic theory between LTAG and LRS grammars. These differences primarily concern assumptions the nature and

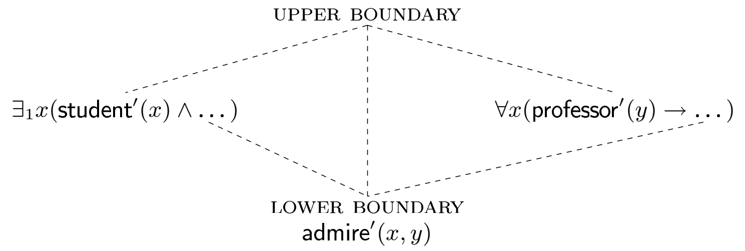

UPPER BOUNDARY
$\exists_1 x(\mathsf{student}'(x) \wedge \ldots)$
$\forall x(\mathsf{professor}'(y) \rightarrow \ldots)$
LOWER BOUNDARY
$\mathsf{admire}'(x, y)$

Figure 10
Depiction of the scope window accounting for the ambiguity in (10)

grammatical characterization of scope boundaries in complex NP constructions or with propositional attitude verbs. We will be interested in constructions like those in (10)–(13):

(10) Exactly one student admires every professor.
$\exists > \forall, \forall > \exists$

(11) Two policemen spy on someone from every city.
$\forall > \exists > two$ (among others)

(12) John seems to have visited everybody.
$seem > \forall, \forall > seem$

(13) Three girls are likely to come.
$three > likely, likely > three$

As illustrated in the examples (10)–(13), in principle quantificational NPs in English can scope freely. An analysis of quantifier scope must minimally guarantee two things: 1) The proposition to which a quantifier attaches must be in the nuclear scope of the quantifier, and 2) A quantifier cannot scope over the next higher finite clause.[12] One way to model this is to define a scope window delimited by a maximal scope boundary and a minimal scope boundary for a quantifier. This idea is illustrated in Fig. 10. A dotted edge from x (higher node) to y (lower node) signifies that y is in the scope of (i.e., is a component of) x. Both LTAG and LRS specify such scope windows for quantifiers. We will now outline the two analyses.

4.1 Specifying a Scope Window for Quantifiers: LTAG

Kallmeyer and Romero (2007) assume that each verb specifies a scope window for attaching quantifiers. The lower boundary is the label of the proposition introduced by the verb. The upper boundary is a meta-variable. The boundaries are encoded by features MAXS and MINS (corresponding to UPPER BOUNDARY and

12 The linguistic theory of scope in LRS is in fact constructed to accommodate the assumption that in certain syntactic environments a limited class of quantifiers may scope out of the finite clause where they syntactically originate. We will henceforth ignore this genuinely linguistic difference from the theory formulated in LTAG.

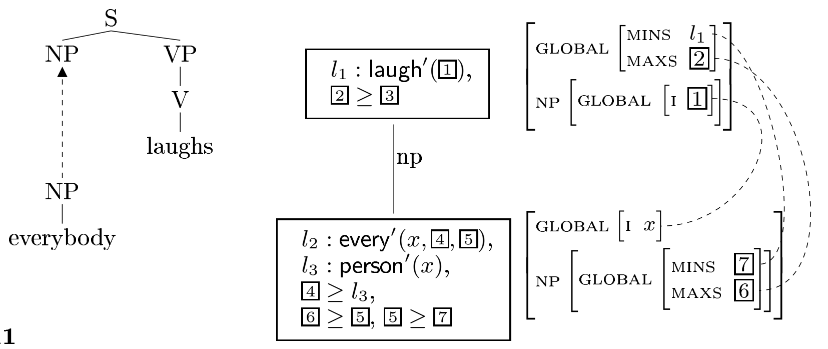

S
NP
VP
V
laughs
NP
everybody
l_1 : laugh$'(\boxed{1})$, $\boxed{2} \geq \boxed{3}$
np
l_2 : every$'(x, \boxed{4}, \boxed{5})$, l_3 : person$'(x)$, $\boxed{4} \geq l_3$, $\boxed{6} \geq \boxed{5}$, $\boxed{5} \geq \boxed{7}$
GLOBAL MINS l_1 MAXS $\boxed{2}$
NP GLOBAL I $\boxed{1}$
GLOBAL I x
NP GLOBAL MINS $\boxed{7}$ MAXS $\boxed{6}$

Figure 11
LTAG analysis of (14) *Everybody laughs*

LOWER BOUNDARY in Fig. 10). Since these two features represent a property of the verb that is not linked to a specific node position, they are defined as global features.[13]

(14) Everybody laughs.

Let us go through the analysis of (14), shown in Fig. 11. The lexical entry of *laugh* (the root node of the derivation tree in Fig. 11) provides an upper boundary MAXS $= \boxed{2}$ and a lower boundary MINS $= l_1$ delimiting the scope window for any quantifier attaching to *laughs*. These boundaries mean that a quantifier that attaches to *laughs* has to minimally scope over the laugh$'$ proposition (label l_1), and it cannot scope higher than $\boxed{2}$.[14]

The semantics of *everybody* has two parts: The generalized quantifier every$'$[15] with its restriction $\boxed{4}$ and its nuclear scope $\boxed{5}$, and a proposition person$'(x)$ that is part of the restriction (constraint $\boxed{4} \geq l_3$). The quantifier looks for the global MAXS and MINS features of the verb it attaches to in order to find the upper and lower boundaries for its nuclear scope. In the semantic representation, the nuclear

13 Global features are grouped under a feature GLOBAL that is linked to the elementary tree as a whole and not to single node positions. Each semantic representation can look into the global features of the mother node in the derivation tree (by putting a request on its root node position or its foot node position) or into the global features of a daughter (by putting a request on the node position to which the daughter attaches).

14 In this simple case, there are no further constraints on the MAXS value $\boxed{2}$, i.e., this upper limit does not have any effect. However, an attitude verb embedding *laugh* as in (15) would embed the *laugh* MAXS feature in its propositional argument and thereby prevent the quantifiers occurring in the embedded clause from taking scope outside of the embedded clause.

(15) Mary fears that everybody laughs.

Note that existential quantifiers such as *someone* always allow for a referential wide scope reading, even when being embedded under an attitude verb. In LTAG we assume that this reading is not a proper scope reading but it is obtained by a different mechanism.

15 Note that in LTAG, generalized quantifiers are treated as constants of a specific type in the underlying Ty2 logic. For a generalized quantifier q, $q(x, p_1, p_2)$ can be read as $q(\lambda x.p_1, \lambda x.p_2)$. In this sense, q is a constant 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 particular, generalized quantifiers are not treated as syncategorematic in LTAG and cannot be identified as a special class of symbols. This is different in LRS where a sort hierarchy of subsorts of *generalized-quantifier* is employed in formulating principles of (immediate) scope constraints and quantifier islands (see Section 4.2 for simple principles referring to the relevant sorts).

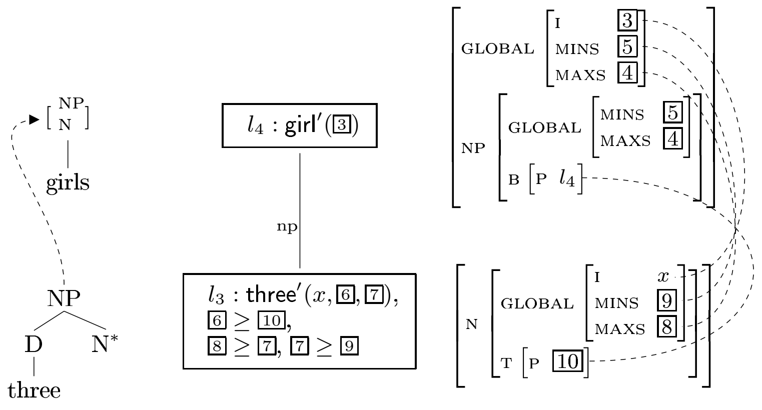


Figure 12
Analysis of NP *three girls* in LTAG

scope $\boxed{5}$ is situated between the variables $\boxed{6}$ and $\boxed{7}$ (constraints $\boxed{6} \geq \boxed{5}$, $\boxed{5} \geq \boxed{7}$). In order to equate these with the MAXS and MINS values of the verb, a request for these features is put on the root node (position NP) of *everybody.* The mechanism of adding feature equations for global features guarantees that this request gets identified with the global feature of *laughs.*

The feature identifications (indicated by dotted lines) lead to the constraints $\boxed{2} \geq \boxed{5}, \boxed{5} \geq l_1$. With the assignments following from the feature identifications ($\boxed{1} \rightarrow x, \boxed{6} \rightarrow \boxed{2}, \boxed{7} \rightarrow l_1$), we obtain the semantic representation (16):

(16)
$$\boxed{\begin{array}{l} l_1 : \mathsf{laugh}'(x), \\ l_2 : \mathsf{every}'(x, \boxed{4}, \boxed{5}),\ l_3 : \mathsf{person}'(x) \\ \boxed{2} \geq l_1, \\ \boxed{4} \geq l_3, \boxed{2} \geq \boxed{5}, \boxed{5} \geq l_1 \end{array}}$$

There is one possible disambiguation consistent with the scope constraints, namely $\boxed{2} \rightarrow l_2$, $\boxed{4} \rightarrow l_3, \boxed{5} \rightarrow l_1$. This leads to the semantics (17):

(17) $\mathsf{every}'(x, \mathsf{person}'(x), \mathsf{laugh}'(x))$

In order to illustrate the way quantifiers interact with other scope taking operators, let us consider the analysis of (13) *Three girls are likely to come.* The combination of the determiner and the noun into the NP *three girls* is shown in Fig. 12. The generalized quantifier *three* carries a request for the global features MAXS (variable $\boxed{8}$) and MINS (variable $\boxed{9}$) of the tree it attaches to, which is the *girls* tree. This tree, in turn, identifies its own global features MAXS (variable $\boxed{4}$) and MINS (variable $\boxed{5}$) with the global features obtained by requesting the global features of the verb. This means that the request for the global MAXS and MINS of the verb at the root of the *girls* tree ultimately concerns the upper and lower scope boundaries of *three.* The P feature provided by *girls* (label l_4) provides the restriction of *three.*

The syntactic analysis of (13) in LTAG is shown in Fig. 13. The raising predicate *be likely* is analyzed like adverbs, using a VP auxiliary tree that adjoins to the verbal spine of the infinitive.

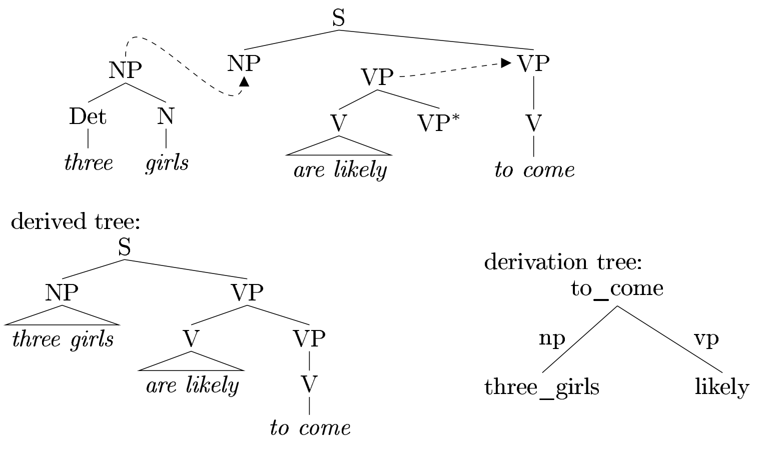


Figure 13
Syntactic analysis of (13) *Three girls are likely to come* in LTAG

The semantic analysis on the derivation tree is shown in Fig. 14. (The node positions VP$_1$ and VP$_2$ stand for the root and foot node positions of the *likely* tree.) The scope properties of the raising verb do not depend on the MAXS–MINS scope window. Instead, its scope depends on its attachment site. It is in a sense inserted between the top and the bottom part of the VP node to which it adjoins, embedding the proposition it finds at the bottom (here l_1) and providing a new proposition (label l_2) at the top. This new proposition is then passed up the verbal spine. This analysis is motivated by the empirical observation that in cases where more than one operator attaches to the verbal spine, the order on the spine determines the scope order:

(18) John tries to seem to be a nice boy.
tries > *seem* > *be a nice boy*, **seem* > *tries* > *be a nice boy*

The scope constraints for the raising verb interact with the quantifier scope window: The larger proposition on the verbal spine, $\boxed{2}$, is below the MAXS boundary (constraint $\boxed{3} \geq \boxed{2}$), and the proposition l_1 contained in the scope of the raising verb is the MINS boundary. As a consequence, *likely* also scopes within the quantifier scope window.

From the additional feature equations in Fig. 14, we obtain the assignments $\boxed{1} \rightarrow x, \boxed{2} \rightarrow l_2, \boxed{6} \rightarrow \boxed{3}, \boxed{7} \rightarrow l_1, \boxed{9} \rightarrow l_1$, which lead to the semantic representation in (19):

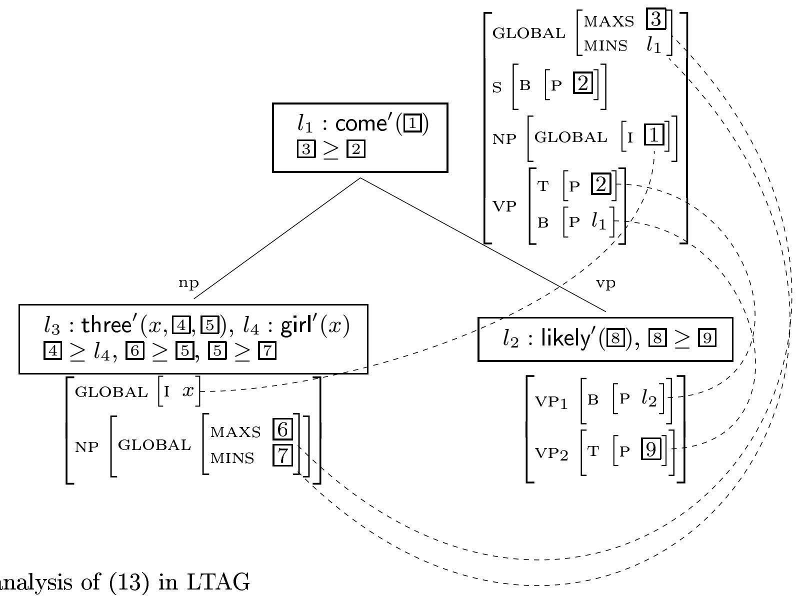

l_1 : come′($\boxed{1}$)
$\boxed{3} \geq \boxed{2}$
GLOBAL MAXS $\boxed{3}$ MINS l_1
S B P $\boxed{2}$
NP GLOBAL I $\boxed{1}$
VP T P $\boxed{2}$ B P l_1
np
vp
l_3 : three′(x,$\boxed{4}$,$\boxed{5}$), l_4 : girl′(x)
$\boxed{4} \geq l_4$, $\boxed{6} \geq \boxed{5}$, $\boxed{5} \geq \boxed{7}$
GLOBAL I x
NP GLOBAL MAXS $\boxed{6}$ MINS $\boxed{7}$
l_2 : likely′($\boxed{8}$), $\boxed{8} \geq \boxed{9}$
VP1 B P l_2
VP2 T P $\boxed{9}$

Figure 14
Semantic analysis of (13) in LTAG

(19)
l_1 : come′(x)
l_3 : three′$(x, \boxed{4}, \boxed{5})$, l_4 : girl′(x)
l_2 : likely′$(\boxed{8})$
$\boxed{3} \geq l_2$,
$\boxed{4} \geq l_4$, $\boxed{3} \geq \boxed{5}$, $\boxed{5} \geq l_1$
$\boxed{8} \geq l_1$

This underspecified representation has the two disambiguations and scope orders in (20):

(20) a. $\boxed{3} \to l_3, \boxed{5} \to l_2, \boxed{8} \to l_1, \boxed{4} \to l_4$.
l_3 : three′$(x, l_4$: girl′$(x), l_2$: likely′$(l_1$: come′$(x)))$

b. $\boxed{3} \to l_2, \boxed{8} \to l_3, \boxed{5} \to l_1, \boxed{4} \to l_4$.
l_2 : likely′$(l_3$: three′$(x, l_4$: girl′$(x), l_1$: come′$(x)))$

4.2 Specifying a Scope Window for Quantifiers: LRS

To analyze the sentences (13) and (14) in LRS, we need to introduce the lexical entries for the quantificational expressions and a new principle, the Semantics Principle. The Semantics Principle of LRS consists of construction-specific clauses which put restrictions on the possible ways in which the semantic representations of syntactic daughters can be combined. Let us start with the lexical entry of *everybody*:

(21) everybody:

$$\begin{bmatrix} \text{PHON} & \langle \textit{everybody} \rangle \\ \text{SS LOC} & \begin{bmatrix} \text{CAT} & \begin{bmatrix}\text{HEAD} & \textit{noun}\\ \text{SUBCAT} & \langle\rangle\end{bmatrix} \\ \text{CONT} & \begin{bmatrix}\text{INDEX VAR} & x \\ \text{MAIN} & \mathsf{every}(x,\alpha,\beta)\end{bmatrix}\end{bmatrix} \\ \text{LF} & \begin{bmatrix} \text{EXCONT} & \textit{me} \\ \text{INCONT} & \boxed{2}\ \mathsf{person}'(x) \\ \text{PARTS} & \langle x, \boxed{2}, \boxed{2a}\mathsf{person}', \boxed{4}\ \mathsf{every}(x,\alpha,\beta)\rangle \end{bmatrix}\end{bmatrix}$$

$$\& \boxed{2} \lhd \alpha \ \& \ x \lhd \alpha \ \& \ x \lhd \beta$$

From a syntactic point of view, *everybody* is a quantified noun phrase like *every girl* and *many boys in my class*. The combination of quantified noun phrases with a verbal projection is subject to a restriction by the SEMANTICS PRINCIPLE, given in (22). We introduce two clauses of this fundamental principle.[16] The first one, (22a), applies when a quantificational determiner combines with a nominal projection; the second, (22b), governs the combination of a quantified noun phrase with a verbal projection.

(22) SEMANTICS PRINCIPLE:

In each *headed-phrase*, the following conditions hold:

a. if the non-head is a quantificational determiner then its INCONT value is of the form *gen-quantifier*(x, ρ, ν), the INCONT value of the head is a component of ρ, and the INCONT value of the non-head daughter is identical with the EXCONT value of the head daughter,

$$\begin{bmatrix}\text{NH-DTR SS LOC} & \begin{bmatrix}\text{CAT HEAD} & \textit{det}\\ \text{CONT MAIN} & \textit{gen-quantifier}\end{bmatrix}\end{bmatrix} \longrightarrow$$

$$\left(\begin{bmatrix}\text{H-DTR LF} & \begin{bmatrix}\text{EXCONT} & \boxed{1}\\ \text{INCONT} & \boxed{2}\end{bmatrix}\\ \text{NH-DTR LF} & \begin{bmatrix}\text{INCONT} & \boxed{1}\begin{bmatrix}\textit{gen-quantifier}\\ \text{RESTR} \ \boxed{3}\end{bmatrix}\end{bmatrix}\end{bmatrix} \wedge \ \boxed{2} \lhd \boxed{3}\right)$$

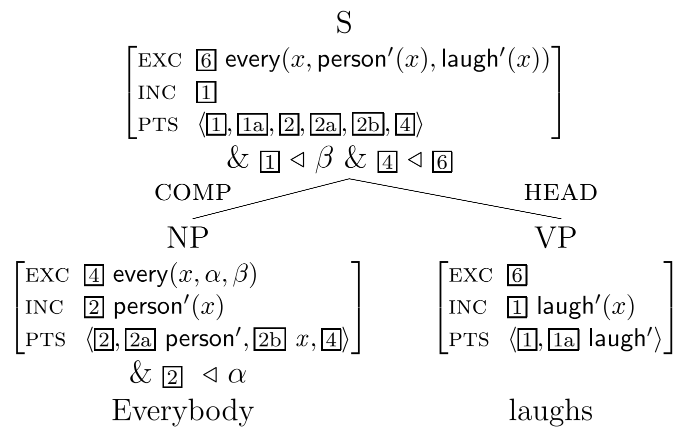
b. if the non-head is a quantified NP with an EXCONT value of the form *gen-quantifier*(x, ρ, ν), then the INCONT value of the head is a component of ν,

16 The SEMANTICS PRINCIPLE for LRS was first formulated with more sub-clauses in (Richter and Sailer, 2001, p. 283), where it still included the LRS principles in (4a)–(4c). Additional clauses of an extended SEMANTICS PRINCIPLE introduce combinatoric restrictions on head-marker phrases, different kinds of head-adjunct phrases, and head-filler phrases. In general, these clauses contribute semantic restrictions on phrases obtained by certain modes of syntactic combination and often take the semantic types or syntactic and semantic class of the immediate daughters of the phrase into account.

$$\forall\boxed{1}\left(\left[\text{NH-DTR}\ \begin{bmatrix}\text{SS LOC CAT} & \begin{bmatrix}\text{HEAD} & \textit{noun}\\ \text{SUBCAT} & \langle\rangle\end{bmatrix}\\ \text{LF EXCONT} & \begin{bmatrix}\textit{gen-quantifier}\\ \text{SCOPE}\ \boxed{1}\end{bmatrix}\end{bmatrix}\right] \rightarrow \exists\boxed{2}\left(\begin{array}{l}[\text{H-DTR LF INCONT}\ \boxed{2}]\\ \wedge\ \boxed{2} \triangleleft \boxed{1}\end{array}\right)\right)$$

According to (22a), when a quantificational determiner is combined with a nominal head, the internal content of the nominal goes into the restrictor of the generalized quantifier. Note that this is a true ‘component-of’ constraint. There might be more in the restrictor of the generalized quantifier than just the internal content of the nominal projection. This is the case if the nominal is modified by a restrictive relative clause or by an intersective adjectival modifier. Their representations will also be part of the restrictor of the generalized quantifier (with the possible exception of certain quantificational operators).

In LRS, the EXCONT value of the utterance is the upper scope boundary while the INCONT value of the syntactic head which selects a quantifier is the lower boundary for scope. This can be seen in the analysis of (14), which is depicted in Fig. 15. The upper boundary is obtained through the interaction of 1) the LRS PROJECTION PRINCIPLE, (4c), stating that the PARTS list of a phrase contains all elements on the PARTS lists of its daughters, and 2) the EXCONT PRINCIPLE, (4b), which states that a) the PARTS list of each non-head contains its own EXCONT, and b) in an utterance, everything on the PARTS list is a component of the EXCONT. This leads to the constraint $\boxed{4} \triangleleft \boxed{6}$ in Fig. 15, among others. The lower boundary is obtained from the SEMANTICS PRINCIPLE, (22b), which states that if the non-head of a headed phrase is a quantifier, then the INCONT of the head is a component of its nuclear scope. This yields $\boxed{1} \triangleleft \beta$ in Fig. 15.



Summary of relevant subterm constraints: $\boxed{2} \triangleleft \alpha$, $\boxed{1} \triangleleft \beta$, $\boxed{4} \triangleleft \boxed{6}$

Figure 15
LRS analysis of (14) *Everybody laughs*

The situation becomes more complex when several scope taking elements interact, as in the ambiguous sentence (13), repeated in (23).

(23) Three girls are likely to come.

In order to analyze this sentence, we first need to explain how LRS handles the combination of a quantificational determiner with a count noun. (24a) and (24b) introduce the relevant parts of the lexical entries for *three* and *girls*:

(24) a. three:

$$\begin{bmatrix} \textsc{phon} & \langle \textit{three} \rangle \\ \textsc{ss loc} & \begin{bmatrix} \textsc{cat} & \begin{bmatrix} \textsc{head} & \textit{det} \\ \textsc{subcat} & \langle\rangle \end{bmatrix} \\ \textsc{cont} & \begin{bmatrix} \textsc{index var} & \boxed{4a}\, x \\ \textsc{main} & \boxed{4}\ \mathsf{three}(x,\alpha,\beta) \end{bmatrix} \end{bmatrix} \\ \textsc{lf} & \begin{bmatrix} \textsc{excont} & \textit{me} \\ \textsc{incont} & \boxed{4}\ \mathsf{three}(x,\alpha,\beta) \\ \textsc{parts} & \langle \boxed{4}, \boxed{4a} \rangle \end{bmatrix} \end{bmatrix}$$

$\&\ x \lhd \alpha\ \&\ x \lhd \beta$

b. girls:

$$\begin{bmatrix} \textsc{phon} & \langle \textit{girls} \rangle \\ \textsc{ss loc} & \begin{bmatrix} \textsc{cat} & \begin{bmatrix} \textsc{head} & \textit{noun} \\ \textsc{subcat} & \left\langle \textsc{DetP}_{\boxed{4a}} \right\rangle \end{bmatrix} \\ \textsc{cont} & \begin{bmatrix} \textsc{index var} & \boxed{4a} \\ \textsc{main} & \mathsf{girl}' \end{bmatrix} \end{bmatrix} \\ \textsc{lf} & \begin{bmatrix} \textsc{excont} & \begin{bmatrix} \textit{gen-quantifier} \\ \textsc{var}\ \boxed{4a} \end{bmatrix} \\ \textsc{incont} & \boxed{2}\ \mathsf{girl}'(\boxed{4a}) \\ \textsc{parts} & \langle \boxed{2}\ \mathsf{girl}'(\boxed{4a}),\ \boxed{2a}\ \mathsf{girl}' \rangle \end{bmatrix} \end{bmatrix}$$

Using the first clause of the SEMANTICS PRINCIPLE, we obtain the structure shown in Fig. 16. The nominal head of the noun phrase gains access to the variable x introduced by the determiner by selecting the determiner as the first element on the SUBCAT list and identifying the argument of the predicate girl' with the variable found under INDEX VAR ($\boxed{4a}$). A comparison of the semantic representation of the quantified noun phrase *three girls* at the NP node with the semantic representation of the quantifier *everybody* (21) immediately reveals their parallel structure.

Before we can analyze (23) we need to provide the semantics of the predicative adjective *likely*. According to (25), it embeds its own INCONT value in the argument α of the operator likely'. The INCONT value is assumed to be raised from its verbal complement. Since the INCONT of the complement is not part of the selected *synsem* structure of the complement, this INCONT raising cannot be lexically specified in the lexical entry of *likely*. Following Sailer (2006) we assume that a general INC RAISING PRINCIPLE takes care of INCONT raising and identifies the INCONT values of the appropriate predicates and their complements.

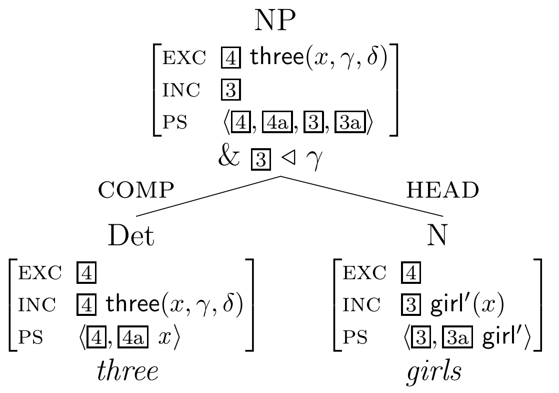


Figure 16
LRS analysis of *three girls*

(25) likely:

$$\begin{bmatrix} \text{PHON} & \langle \textit{likely} \rangle \\ \text{SS LOC} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \textit{adj} \\ \text{SUBCAT} & \left\langle \begin{bmatrix} \text{LOC} & \begin{bmatrix} \text{CAT HEAD} & \textit{verb} \\ \text{CONT MAIN} & \boxed{1b} \end{bmatrix} \end{bmatrix} \right\rangle \end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \text{MAIN} & \boxed{2a}\ \mathsf{likely}' \end{bmatrix} \end{bmatrix} \\ \text{LF} & \begin{bmatrix} \text{EXCONT} & \textit{me} \\ \text{INCONT} & \boxed{1} \\ \text{PARTS} & \langle \boxed{1},\ \boxed{2}\ \mathsf{likely}'(\alpha),\ \boxed{2a} \rangle \end{bmatrix} \end{bmatrix}$$

$$\&\ \boxed{1} \lhd \alpha\ \&\ \boxed{1b} \lhd \boxed{1}$$

The semantic constraints of the lexical entry of *come* are parallel to the lexical entry of *laugh* (3b); the auxiliaries *to* and *are* are analyzed as INC raising predicates similar to *likely* but without contributing any constant to the semantic representations.

Fig. 17 shows the structure of (23) which follows from the lexical entries and the LRS principles above. The figure repeats only those subterm constraints which are essential to see the treatment of scope windows and the way the two readings of the sentence are derived.

The restrictions in Fig. 17 (in combination with the grammar of well-formed expressions of Ty2 presented in Section 2.2) leaves two possibilities for the EXCONT value $\boxed{5}$:

(26) a. $\boxed{5} = \mathsf{three}(x,\ \mathsf{girl}'(x),\ \mathsf{likely}'(\mathsf{come}'(x)))$

b. $\boxed{5} = \mathsf{likely}'(\mathsf{three}(x,\ \mathsf{girl}'(x),\ \mathsf{come}'(x)))$

The scope window for the operators in (23) is delineated by the internal content of *come*, $\boxed{1}$, and the external content of the complete utterance, $\boxed{5}$. By definition, the internal content of each sign s is outscoped by all scope-taking elements

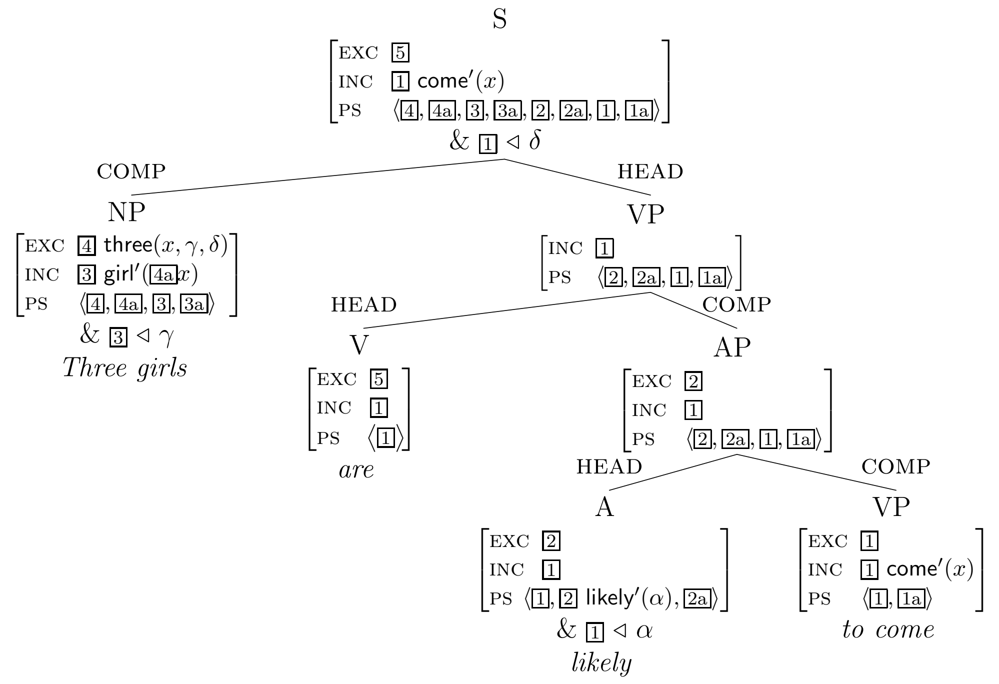

S
EXC $\boxed{5}$
INC $\boxed{1}$ $\mathsf{come}'(x)$
PS $\langle\boxed{4},\boxed{4a},\boxed{3},\boxed{3a},\boxed{2},\boxed{2a},\boxed{1},\boxed{1a}\rangle$
& $\boxed{1} \lhd \delta$
COMP
HEAD
NP
EXC $\boxed{4}$ $\mathsf{three}(x,\gamma,\delta)$
INC $\boxed{3}$ $\mathsf{girl}'(\boxed{4a}x)$
PS $\langle\boxed{4},\boxed{4a},\boxed{3},\boxed{3a}\rangle$
& $\boxed{3} \lhd \gamma$
Three girls
VP
INC $\boxed{1}$
PS $\langle\boxed{2},\boxed{2a},\boxed{1},\boxed{1a}\rangle$
HEAD
COMP
V
EXC $\boxed{5}$
INC $\boxed{1}$
PS $\langle\boxed{1}\rangle$
are
AP
EXC $\boxed{2}$
INC $\boxed{1}$
PS $\langle\boxed{2},\boxed{2a},\boxed{1},\boxed{1a}\rangle$
HEAD
COMP
A
EXC $\boxed{2}$
INC $\boxed{1}$
PS $\langle\boxed{1},\boxed{2}\ \mathsf{likely}'(\alpha),\boxed{2a}\rangle$
& $\boxed{1} \lhd \alpha$
likely
VP
EXC $\boxed{1}$
INC $\boxed{1}$ $\mathsf{come}'(x)$
PS $\langle\boxed{1},\boxed{1a}\rangle$
to come

Figure 17
LRS analysis of *Three girls are likely to come*

of the signs with which s combines. The equality of the INCONT of the sentence with the INCONT of *come* is mediated by the fact that *to*, *likely* and *are* are analyzed as INCONT raisers which identify their own INCONT with the INCONT of their VP (or AP) argument. Two subterm constraints interact with the internal content of *come*. The predicative adjective *likely* outscopes $\boxed{1}$ according to its lexical restrictions ($\boxed{1} \lhd \alpha$). Clause b of the SEMANTICS PRINCIPLE, (22), adds the restriction that $\boxed{1}$ be in the nuclear scope of the quantifier *three girls* ($\boxed{1} \lhd \delta$). In essence, these component constraints demand that both the quantifier and likely' outscope $\mathsf{come}'(x)$, their relative scope is not determined. Adding the fact that $\mathsf{girl}'(x)$ must be in the restrictor of *three girls* ($\boxed{3} \lhd \gamma$), the two expressions in (26) are the only Ty2 terms satisfying all restrictions of the grammar.[17]

For a complete understanding of underspecification in the HPSG-LRS architecture, it is important to consider the (exhaustive) models in the denotation of the grammar. These models contain two structures for the sentence (23), one for each reading. The two structures are syntactically identical, but one has the EXCONT value in (26a) whereas the other has the EXCONT value in (26b). This means that there is no underspecification at the level of semantic representations. Underspecification is only a matter of the TFL specification of the grammar.[18]

17 Note that the reading in (26b) implies $\boxed{5} = \boxed{2}$, which is not the case in reading (26a).
18 Underspecification is also a crucial ingredient of the CLLRS implementation. Computation with CLLRS can thus be viewed as computation using underspecification techniques.

4.3 Summary: Features and Scope Constraints

The two quantifier scope analyses in LTAG and LRS illustrate two points. The first concerns the comparison of LTAG and LRS. As we have shown, both approaches use a feature architecture for a quantifier scope window to capture the freedom of quantifier scope within certain syntactically defined domains. LTAG semantics and LRS use a level of underspecification involving 'component-of'-constraints, although the status of the underspecification layer of grammar is different in the two frameworks. In finite sentences, there is a clear correspondence between LTAG's attributes MAXS and MINS and the attributes EXCONT and INCONT in LRS. The striking similarity between the two analyses shows that, despite the mathematical differences between the frameworks, central insights can be modelled in parallel. Interesting differences are expected to emerge in detailed analyses of subtle linguistic facts and possibly in the computational behavior of implementations of LTAG semantics and CLLRS.[19] An area in which differences between the two architectures matter will be discussed in the next section. As for computational differences, it is still too early to draw conclusions.

The second point concerns a contrast between feature logic-based computational semantics and the tradition of logical form semantics as an extension of generative syntax in the style of Heim and Kratzer (1998). As we have demonstrated, the use of feature logics with feature value identifications in combination with underspecification techniques allows us to avoid syntactic movement operations such as quantifier raising for the representation of scope. In other words, features are not only used to establish predicate-argument relations but they also serve to determine scope boundaries. This is possible because of the mechanisms for percolating feature values on the derived tree provided in LRS and LTAG and, in addition, because of LTAG's extended domain of locality.

5 Consequences of Encoding Semantic Formulas in a Feature Logic: The Case of Negative Concord

Negative concord can be characterized as a type of construction in which the occurrence of several negation-bearing elements such as negative quantifiers (*no one, nothing*) and negative particles (*not*) lead to an interpretation with only one negation. The analysis of negative concord in Polish in LTAG and LRS described in this section highlights differences in the theories' implementation of underspecification techniques. They are of particular interest to our discussion because they go along with the different functions of the feature logics in the two frameworks. Both LTAG and LRS use component-of constraints, but they are used in different ways in the two grammar architectures. In LTAG, these constraints link underspecified Ty2 terms that are augmented with holes and labels of Hole Semantics, while in LRS, they belong to the descriptions of fully specified Ty2 terms. The possibility in LRS of referring to the same Ty2 expressions multiple times and at different points in the constituent structure of a sentence permits an interesting

19 See the remarks in Section 2.3 on CLLRS.

treatment of negative concord which cannot be mirrored directly in LTAG.

5.1 Negative Concord in Polish

Polish is a classical negative concord language. The basic facts of sentential negation and negative concord in Polish are illustrated in (27)–(29):

(27) Janek nie pomaga ojcu.
Janek NM helps father
'Janek doesn't help his father.'

(28) a. Janek nie pomaga nikomu.
Janek NM helps nobody
'Janek doesn't help anybody.'

b. *Janek pomaga nikomu.

(29) Nikt nie przyszedł.
nobody NM came
'Nobody came.'

The verbal prefix *nie* is obligatory for expressing sentential negation, and it can co-occur with any number of n-words (such as *nikt*, 'nobody/anybody') without ever leading to a double negation reading. As a consequence, (29) expresses only one logical sentential negation, although both the negation prefix *nie* on the verb and the n-word *nikt* can carry logical negation alone in other contexts. We will now present analyses in LTAG and LRS of negative concord in Polish, taking sentence (29) as our example.

5.2 Concord Phenomena in LRS

LRS takes advantage of the fact that its specifications of semantic representations are descriptions of logical expressions which can, in principle, mention the same parts of the expressions several times. (30) shows the relevant part of the lexical entries of *nikt (nobody)* and *nie przyszedł* which we need in the analysis of (29). Following Kupść (2000) we assume that *nie* is a verbal prefix and forms a morphological unit with the verb. The lexical entry of *nikt (nobody)* in (30a) is similar to the relevant parts of English *everybody* in (21). However, as a negative quantifier it also introduces a negation into the semantic representation. This negation receives a special treatment. According to (30a), the negation component, $\boxed{4}$, of *nikt* is not a component of its external content, which looks like an existential quantifier. Instead, the condition $\boxed{5} \triangleleft \beta$ only demands that the existential quantifier in the external content be in the scope of the negation.

(30) a. nikt (*nobody*):

$$\begin{bmatrix} \textit{word} & \\ \text{PHON} & \langle \textit{nikt} \rangle \\ \text{SS LOC CONT} & \begin{bmatrix} \text{INDEX VAR} & \boxed{3b}\ x \\ \text{MAIN} & \boxed{5}\ \mathsf{some}(x,\gamma,\delta) \end{bmatrix} \\ \text{LF} & \begin{bmatrix} \textit{lrs} & \\ \text{EXCONT} & \boxed{5}\ \mathsf{some}(x,\gamma,\delta) \\ \text{INCONT} & \boxed{3}\ \mathsf{person'}(\boxed{3b}x) \\ \text{PARTS} & \langle \boxed{3}, \boxed{3a}\mathsf{person'}, \boxed{3b}, \boxed{4}\neg\beta, \boxed{5} \rangle \end{bmatrix} \end{bmatrix}$$

$$\&\ \boxed{5} \triangleleft \beta \ \&\ \boxed{3} \triangleleft \gamma \ \&\ x \triangleleft \gamma \ \&\ x \triangleleft \delta$$

b. nie przyszedł (*NM came*):

$$\begin{bmatrix} \textit{word} & \\ \text{PHON} & \langle \textit{nie przyszedł} \rangle \\ \text{SS LOC} & \begin{bmatrix} \text{CAT SUBCAT} & \left\langle \left[\text{LOC CONT INDEX VAR } \boxed{3b} \right] \right\rangle \\ \text{CONT MAIN} & \mathsf{come'} \end{bmatrix} \\ \text{LF} & \begin{bmatrix} \textit{lrs} & \\ \text{EXCONT} & \boxed{0} \\ \text{INCONT} & \boxed{1}\ \mathsf{come'}(\boxed{3b}) \\ \text{PARTS} & \langle \boxed{1}, \boxed{1a}\mathsf{come'}, \boxed{2}\neg\alpha, \rangle \end{bmatrix} \end{bmatrix}$$

$$\&\ \boxed{1} \triangleleft \alpha \ \&\ \boxed{2} \triangleleft \boxed{0}$$

In contrast to the specifications for *nikt*, the verb *nie przyszedł* realizes the negation within its external content. The lexical entry (30b) does this by stating that the negation, $\boxed{2}$, must be a subterm of the external content, $\boxed{0}$.

Without additional principles these lexical specifications are not sufficient to guarantee the only available reading of the sentence (29), i.e., the reading with a single sentential negation. First of all, nothing enforces the obligatory presence of the negation prefix with the finite verb in the presence of *nikt*. Second, a double negation reading may result from not identifying the negation contributed by *nikt* and by the verb. To overcome these shortcomings, Richter and Sailer (2004) introduce two language-specific principles which determine the behavior of negative concord in Polish. We restate these two principles informally in (31a) and (31b).

(31) a. The NEG CRITERION:
For every finite verb, if there is a negation in the external content of the verb that has scope over the verb's MAIN value, then the negation must be an element of the verb's PARTS list.

b. The NEGATION COMPLEXITY CONSTRAINT:
For each sign, there may be at most one negation which is a component of the external content and has the MAIN value as its component.

According to the NEG CRITERION for Polish a (finite) verb in the scope of negation must contribute negation itself. The NEGATION COMPLEXITY CONSTRAINT limits to one the number of negations taking scope over a MAIN value within the projection domain of an external content. With these additional restrictions in place, we can now derive the meaning of (29) in LRS. Fig. 18 shows that both *nikt* and the verb *nie przyszedł* introduce descriptions of negations ($\boxed{4}$ and $\boxed{2}$, respectively). The constraints on negative concord in Polish conspire to force the negations contributed by the two words to be the same in the overall logical representation $\boxed{0}$ of the sentence ($\boxed{2} = \boxed{4}$ $(= \boxed{0})$). Moreover, the negation must outscope the existential quantifier introduced by *nikt* due to the lexical scope constraint $\boxed{5} \triangleleft \beta$ of *nikt*. The restriction $\boxed{1} \triangleleft \delta$ comes from the second clause of the SEMANTICS PRINCIPLE, (22).

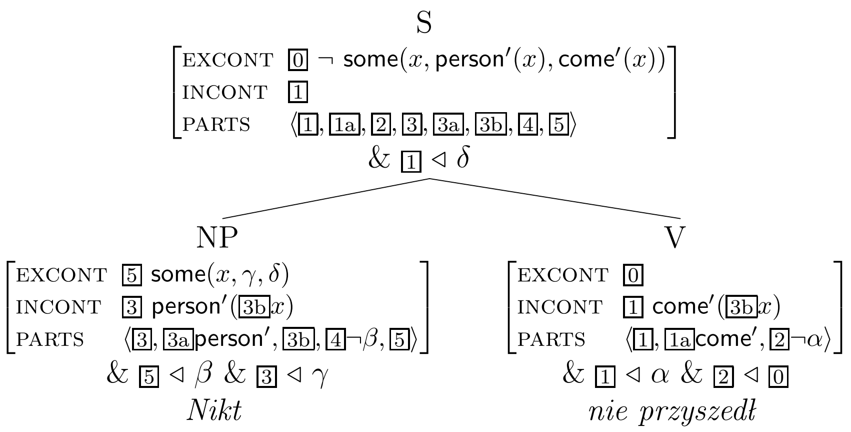

S
EXCONT $\boxed{0}$ $\neg$ some$(x, \text{person}'(x), \text{come}'(x))$
INCONT $\boxed{1}$
PARTS $\langle \boxed{1}, \boxed{1a}, \boxed{2}, \boxed{3}, \boxed{3a}, \boxed{3b}, \boxed{4}, \boxed{5} \rangle$
$\& \boxed{1} \triangleleft \delta$
NP
EXCONT $\boxed{5}$ some(x, γ, δ)
INCONT $\boxed{3}$ person$'(\boxed{3b}x)$
PARTS $\langle \boxed{3}, \boxed{3a}\text{person}', \boxed{3b}, \boxed{4}\neg\beta, \boxed{5} \rangle$
$\& \boxed{5} \triangleleft \beta \ \& \boxed{3} \triangleleft \gamma$
Nikt
V
EXCONT $\boxed{0}$
INCONT $\boxed{1}$ come$'(\boxed{3b}x)$
PARTS $\langle \boxed{1}, \boxed{1a}\text{come}', \boxed{2}\neg\alpha \rangle$
$\& \boxed{1} \triangleleft \alpha \ \& \boxed{2} \triangleleft \boxed{0}$
nie przyszedł

Figure 18
LRS analysis of (29) *Nikt nie przyszedł* (Nobody came)

5.3 Concord Phenomena in LTAG

An LRS-style analysis of negative concord is not possible in LTAG. Recall that the feature logic is not used to encode the Ty2 formulas as is the case in LRS. In LTAG, the formulas in the semantic representations are considered different objects, i.e., different subformulas of the final semantic representations we obtain after disambiguating the underspecified representation. Therefore, each negation in the interpretation corresponds to exactly one negated term introduced in the semantic representations from the lexicon.

Since the interpretation of (29) contains only one negation, there can be only one negation in the lexical entries involved in the derivation. As there can be several n-words in a sentence without resulting in multiple negation and the presence of the negative marker *nie* is obligatory, *nie* necessarily introduces the negation. The n-word *nikt* can then be analyzed as an existential quantifier that requires 1. the presence of a negation and 2. being in the scope of this negation.

An LTAG analysis along these lines is sketched in Fig. 19. It resembles in many respects the NPI analysis proposed in Lichte and Kallmeyer (2006). Let us explain its different aspects.

Verbs have two more global features, besides MAXS and MINS that were already introduced in Section 4: a feature NEG indicating the presence of a negation and a feature N-SCOPE containing the scope of this negation. If the verb is negated (as it is the case in Fig. 19), the global NEG should be set to *yes*, otherwise it should be set to *no*. (In the latter case, the feature N-SCOPE is irrelevant.) To achieve this, we introduce an additional local feature NEG on the V node. At the bottom, this feature has the value *no*. If no negative marker adjoins, this will be identified with the top (variable $\boxed{3}$) and passed from there to the global NEG. The negative marker adjoins to the V node and switches this local NEG feature to *yes* by specifying NEG= *yes* at the top of its root node. This *yes* gets identified with $\boxed{3}$ because of the adjunction. In addition, the negation *nie* identifies its scope (variable $\boxed{9}$) with the global N-SCOPE of the verb it attaches to (here $\boxed{1} = \boxed{9}$) and scopes over the proposition of the verb (the MINS feature, constraint $\boxed{9} \geq \boxed{10}$).

The n-word is an existential quantifier. It requires the global feature NEG of the verb it attaches to to be *yes* thereby checking the presence of a negation. Furthermore, its maximal scope boundary (constraint $\boxed{7} \geq \boxed{6}$) is not the MAXS value of the verb but the N-SCOPE value (identification $\boxed{1} = \boxed{7}$). This ensures that the existential quantifier is in the scope of the negation.

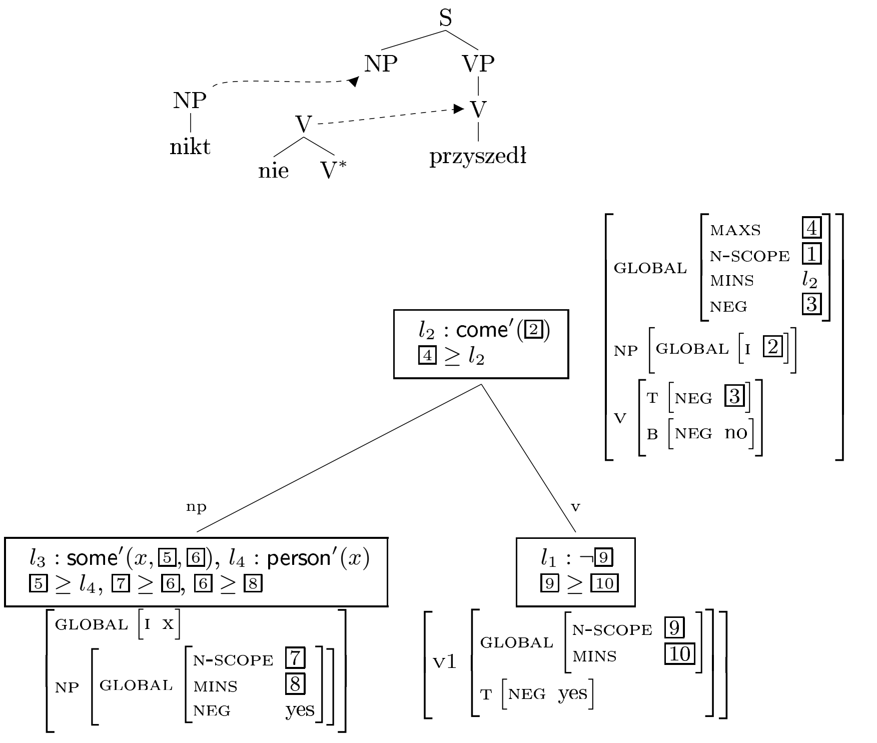

S
NP
VP
NP
nikt
V
nie
V*
V
przyszedł
l_2 : come′($\boxed{2}$)
$\boxed{4} \geq l_2$
GLOBAL MAXS $\boxed{4}$ N-SCOPE $\boxed{1}$ MINS l_2 NEG $\boxed{3}$
NP GLOBAL I $\boxed{2}$
V T NEG $\boxed{3}$ B NEG no
np
v
l_3 : some′(x,$\boxed{5}$,$\boxed{6}$), l_4 : person′(x)
$\boxed{5} \geq l_4$, $\boxed{7} \geq \boxed{6}$, $\boxed{6} \geq \boxed{8}$
GLOBAL I x
NP GLOBAL N-SCOPE $\boxed{7}$ MINS $\boxed{8}$ NEG yes
l_1 : $\neg\boxed{9}$
$\boxed{9} \geq \boxed{10}$
v1 GLOBAL N-SCOPE $\boxed{9}$ MINS $\boxed{10}$
T NEG yes

Figure 19
LTAG analysis of (29) *Nikt nie przyszedł*

As a result, building the union of the semantic representations and applying the assignment obtained from the feature identification, we obtain the semantic representation in (32) for Fig. 19:

(32) $$\boxed{\begin{array}{l} l_2 : \mathsf{come}'(x),\ l_3 : \mathsf{some}'(x, \boxed{5}, \boxed{6}),\ l_4 : \mathsf{person}'(x),\ l_1 : \neg\boxed{1} \\ \boxed{4} \geq l_2,\ \boxed{5} \geq l_4,\ \boxed{1} \geq \boxed{6},\ \boxed{6} \geq l_2,\ \boxed{1} \geq l_2 \end{array}}$$

5.4 Summary: Underspecification in LRS and in LTAG

The analysis of negative concord demonstrates that the two frameworks differ substantially in their treatment of underspecification: 1. LRS employs partial descriptions of fully specified models, whereas LTAG generates underspecified representations in the style of Bos (1995) that require the definition of a disambiguation (a "plugging" in the terminology of Bos). 2. LRS constraints contain descriptions of Ty2 terms rather than Ty2 terms. Therefore, unlike in LTAG, two descriptions can denote the same formula. Because of this, the analysis of negative concord in LRS described above can introduce several negations at the TFL description level that get identified in the models of the constraint system. This is not possible in LTAG, where the feature logic only mediates between pieces of underspecified Ty2 expressions. As a result, LTAG is more limited than LRS. On the other hand, the way semantic representations are defined in LTAG guarantees that they almost correspond to *normal dominance constraints*, which are known to be polynomially parsable (see Althaus et al. 2003).

The difference in the use of underspecification techniques reflects the more general difference between the two types of mathematical systems: In a generative linear rewriting system such as LTAG the elements of the grammar are objects (here: elementary trees paired with sets of Ty2 terms), and copying or erasing is disallowed during derivations. By contrast, in a purely description-based formalism such as HPSG token identities between different components of linguistic structures are natural and frequently employed.

6 Feature Logic-Based Semantic Computation and Compositionality

At first sight, feature logic-based computational semantics systems such as LTAG and LRS do not seem compatible with a notion of compositionality. Clearly, in these frameworks the derived trees do not determine the meaning of a phrase in such a way that it is the result of applying the meaning of one daughter to the meaning of another (in binary branching structures). In order to show that these systems are still compositional, we have to identify a different structure that determines syntax and semantics. In this section we will sketch some ideas for LTAG concerning this question.

The key to answering the question about the compositionality of LTAG semantics is the fact that LTAG is a mildly context-sensitive grammar: The derivation process (i.e., the process of syntactic combination) can be described by a context-free structure, the derivation tree. In this section we will demonstrate that this context-free structure also specifies the process of semantic combination in a way that corresponds to Hodges' (2001) definition of a compositional semantics.

6.1 TAG as a Linear Context-free Rewriting System

TAGs are mildly context-sensitive; they belong to the class of linear context-free rewriting systems (LCFRS, Weir (1988)). Consequently, they have an underlying context-free backbone – the derivation trees – that denotes the trees that can be derived. In this section we will outline how to define a context-free grammar describing the derivation trees. We can then define syntactic and semantic denotations for this grammar: The syntactic denotations are the derived trees while the semantic denotations are the resulting semantic representations plus the feature structure descriptions one obtains from the conjunction of the different descriptions involved and the equations arising from the substitutions and adjunctions.

An LCFRS consists of

- a generalized context-free grammar (GCFG) generating terms in a term algebra that correspond, in our case, to the derivation trees,
- the (syntactic) denotations of these terms (the derived trees), and
- functions specifying how to compute the strings they yield.

In the following, we will ignore the strings produced by the terms of the GCFG. Instead, we will focus on defining the semantic denotations of the terms.

6.1.1 The Generalized Context-free Grammar A Generalized Context-free Grammar (GCFG) is a context-free grammar that generates terms. It consists of

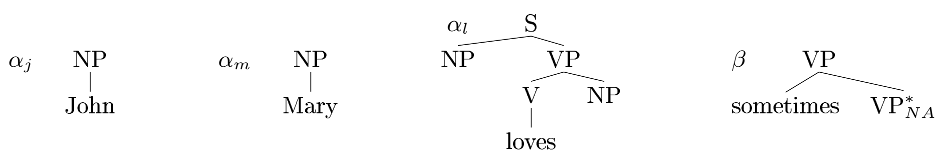
- disjoint alphabets N and F, the nonterminals and the function symbols,
- a start symbol $S \in N$, and
- a finite set of productions P of the form $A \to f(A_1, \dots, A_n)$ where $n \geq 0, f \in F$ and $A, A_1, \dots, A_n \in N$.

A GCFG derives a set of terms in the following way:

- $A \Rightarrow f()$ if $A \to f()$ is a production.
- $A \overset{*}{\Rightarrow} f(t_1, \dots, t_n)$ if there is a production $A \to f(A_1, \dots, A_n)$ and $A_i \overset{*}{\Rightarrow} t_i$ for $1 \leq i \leq n$.

The idea of the GCFG specifying the derivations of a TAG is as follows: The GCFG productions specify possible adjunctions and substitutions for each elementary tree. The elementary trees are the nonterminal symbols of the GCFG. The set of terms one can derive from some elementary γ specifies all the derivation trees with root symbol γ. In particular, the γ-productions specify the different possibilities for the daughters of γ in the derivation tree, i.e., the different combinations of adjunctions and substitutions possible for γ. More concretely, the productions have the form $\gamma \to f_{\gamma:p_1,\dots,p_n}(\gamma_1, \dots, \gamma_n)$ with $\gamma, \gamma_1, \dots, \gamma_n$ being elementary trees and $p_1, \dots, p_n$ being node addresses in γ. This production indicates that $\gamma_1, \dots, \gamma_n$ can be attached (by substitution or adjunction) to γ at node addresses $p_1, \dots, p_n$.

Consider for example the TAG in Fig. 20 with the corresponding GCFG that characterizes the derivation trees.[20] With this grammar, for (33) we obtain the derivation tree and corresponding term tree in (34).[21] (The term itself is $f_{\alpha_l:1,2,22}(f_{\alpha_j}(), f_{\beta:\epsilon}(f_\beta()), f_{\alpha_m}())$. Interpreted as a bracketed tree, this gives the second tree in (34).)



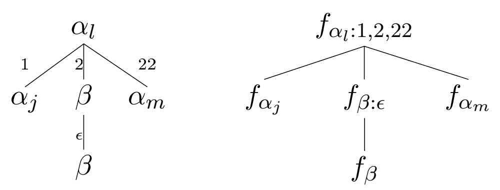
α_j NP John
α_m NP Mary
α_l S NP VP V NP loves
β VP sometimes VP^*_{NA}

Corresponding GCFG:

$\alpha_j \rightarrow f_{\alpha_j}()$	no subst./adj. to α_j
$\alpha_m \rightarrow f_{\alpha_m}()$	no subst./adj. to α_m
$\beta \rightarrow f_\beta()$	no subst./adj. to β
$\beta \rightarrow f_{\beta:\epsilon}(\beta)$	adjunction of β to the root of β
$\alpha_l \rightarrow f_{\alpha_l:1,22}(\alpha_j, \alpha_m)$	substitutions of α_j and α_m at addr. 1 and 22 resp. in α_l
$\alpha_l \rightarrow f_{\alpha_l:1,22}(\alpha_j, \alpha_j)$	...
$\alpha_l \rightarrow f_{\alpha_l:1,22}(\alpha_m, \alpha_j)$	...
$\alpha_l \rightarrow f_{\alpha_l:1,22}(\alpha_m, \alpha_m)$	...
$\alpha_l \rightarrow f_{\alpha_l:1,2,22}(\alpha_j, \beta, \alpha_m)$	subst./adj. of α_j, β and α_m at addr. 1, 2 and 22 resp. in α_l
$\alpha_l \rightarrow f_{\alpha_l:1,2,22}(\alpha_j, \beta, \alpha_j)$	...
$\alpha_l \rightarrow f_{\alpha_l:1,2,22}(\alpha_m, \beta, \alpha_j)$	...
$\alpha_l \rightarrow f_{\alpha_l:1,2,22}(\alpha_m, \beta, \alpha_m)$	...

Figure 20
Sample LTAG and corresponding GCFG

(33) John sometimes sometimes loves Mary

(34)


The general construction of the GCFG G for a given TAG is as follows:

- The nonterminal symbols of G are the elementary trees.
- For each elementary γ without OA constraints[22] and without substitution nodes: there is a zero arity function f_γ and a production $\gamma \rightarrow f_\gamma()$.

20 Here, we use Gorn addresses for the positions of the nodes: The root has the address ϵ and the jth child of a node with address p has address $p \cdot j$.
21 We are aware that (33) is not really an English sentence. But for the examples in this section, we try to keep our grammar as small as possible and thus provide only one VP modifier. This is why we chose this odd example.
22 OA stands for "obligatory adjunction". TAG allows one to specify for each node whether adjunction at that node is obligatory or not. If it is obligatory, the node is said to have an OA constraint.

- For each γ and positions $p_1, \ldots, p_n$ in γ comprising all OA nodes and all substitution nodes and $\gamma_1, \ldots, \gamma_n$ that can be adjoined/substituted at positions $p_1, \ldots, p_n$ respectively: There is a n-ary function $f_{\gamma:p_1,\ldots p_n}$ and a production $\gamma \rightarrow f_{\gamma:p_1,\ldots p_n}(\gamma_1, \ldots, \gamma_n)$

6.1.2 The syntactic denotation The term trees denote derived trees in the same way as derivation trees determine derived trees:

- For all productions of the form $\gamma \rightarrow f_\gamma()$: $[\![f_\gamma()]\!]_{syn} := \gamma$.
- For all productions of the form $\gamma \rightarrow f_{\gamma:p_1,\ldots p_n}(\gamma_1, \ldots, \gamma_n)$: $[\![f_{\gamma:p_1,\ldots p_n}(t_1, \ldots, t_n)]\!]_{syn} := \gamma[p_1, [\![t_1]\!]_{syn}] \ldots [p_n, [\![t_n]\!]_{syn}]$.[23]

For our sample LTAG we obtain:

(35)
$$\begin{array}{lll} [\![f_{\alpha_j}()]\!]_{syn} = \alpha_j & [\![f_{\alpha_m}()]\!]_{syn} = \alpha_m & [\![f_\beta()]\!]_{syn} = \beta \\ [\![f_{\beta:\epsilon}(X)]\!]_{syn} = \beta[\epsilon, [\![X]\!]_{syn}] & & \\ [\![f_{\alpha_l:1,22}(X,Y)]\!]_{syn} = \alpha_l[1, [\![X]\!]_{syn}][22, [\![Y]\!]_{syn}] & & \\ [\![f_{\alpha_l:1,2,22}(X,Y,Z)]\!]_{syn} = \alpha_l[1, [\![X]\!]_{syn}][2, [\![Y]\!]_{syn}][22, [\![Z]\!]_{syn}] & & \end{array}$$

The syntactic denotation of the term tree (34) for (33) is then (36):

(36) $[\![f_{\alpha_l:1,2,22}(f_{\alpha_j}(), f_{\beta:\epsilon}(f_\beta()), f_{\alpha_m}())]\!]_{syn} = \alpha_l[1, \alpha_j][2, \beta[\epsilon, \beta]][22, \alpha_m]$

The expression $\alpha_l[1, \alpha_j][2, \beta[\epsilon, \beta]][22, \alpha_m]$ in (36) denotes the derived tree one obtains by starting with α_l (the *likes* tree), substituting the node at position 1 for a tree t_1, adjoining a tree t_2 at position 2 and substituting the node at position 22 for a tree t_3 where t_1, t_2, t_3 are as follows: t_1 is the tree α_j of *John* without any further adjunctions or substitutions, t_3 is the tree α_m of *Mary* without any further adjunctions or substitutions, and t_2 ($\beta[\epsilon, \beta]$) can be obtained by taking the *sometimes* tree β and adjoining to its root (position ϵ) again the *sometimes* tree β.

6.2 Semantic Denotation of the GCFG Terms

The crucial question now is whether the GCFG productions also specify semantic composition. In the following we will show that this is the case.

Let σ be a function assigning to each elementary tree a pair consisting of a semantic representation and a feature structure description. The function σ for our sample grammar is shown in Fig. 21.

We assume that each time a syntactic category (an elementary tree γ) occurs in a term, σ assigns a fresh instance of $\sigma(\gamma)$ (i.e., an instance with fresh labels, variables and meta-variables).

23 $\gamma[p, \gamma']$ is defined as follows: if γ' is (derived from) an initial tree and the node at position p in γ is a substitution node, then $\gamma[p, \gamma']$ is the tree one obtains by substituting γ' into γ at node position p. If γ' is (derived from) an auxiliary tree and the node at position p in γ is an internal node, then $\gamma[p, \gamma']$ is the tree one obtains by adjoining γ' to γ at node position p. Otherwise $\gamma[p, \gamma']$ is undefined. (Note that the undefined case cannot happen here due to the construction of the GCFG.)

$$\sigma(\alpha_j) = \langle \sigma_{\alpha_j}, \delta_{\alpha_j} \rangle := \langle \quad \boxed{\mathsf{john}'(x)} \quad , \quad \boxed{0}\begin{bmatrix} \epsilon & \begin{bmatrix} \textsc{b} & \begin{bmatrix} \textsc{i} & x \end{bmatrix} \end{bmatrix} \end{bmatrix} \quad \rangle$$

$$\sigma(\alpha_m) = \langle \sigma_{\alpha_m}, \delta_{\alpha_m} \rangle := \langle \quad \boxed{\mathsf{mary}'(y)} \quad , \quad \boxed{1}\begin{bmatrix} \epsilon & \begin{bmatrix} \textsc{b} & \begin{bmatrix} \textsc{i} & y \end{bmatrix} \end{bmatrix} \end{bmatrix} \quad \rangle$$

$$\sigma(\alpha_l) = \langle \sigma_{\alpha_l}, \delta_{\alpha_l} \rangle := \langle \quad \boxed{l_1 : \mathsf{love}'(\boxed{2}, \boxed{3})} \quad , \quad \boxed{4}\begin{bmatrix} 1 & \begin{bmatrix} \textsc{t} & \begin{bmatrix} \textsc{i} & \boxed{2} \end{bmatrix} \end{bmatrix} \\ 22 & \begin{bmatrix} \textsc{t} & \begin{bmatrix} \textsc{i} & \boxed{3} \end{bmatrix} \end{bmatrix} \\ 2 & \begin{bmatrix} \textsc{t} & \begin{bmatrix} \textsc{p} & \boxed{5} \end{bmatrix} \\ \textsc{b} & \begin{bmatrix} \textsc{p} & l_1 \end{bmatrix} \end{bmatrix} \end{bmatrix} \quad \rangle$$

$$\sigma(\beta) = \langle \sigma_{\beta}, \delta_{\beta} \rangle := \langle \quad \boxed{\begin{array}{l} l_2 : \mathsf{sometimes}'(\boxed{6}) \\ \boxed{6} \geq \boxed{7} \end{array}} \quad , \quad \boxed{8}\begin{bmatrix} \epsilon & \begin{bmatrix} \textsc{b} & \begin{bmatrix} \textsc{p} & l_2 \end{bmatrix} \end{bmatrix} \\ 2 & \begin{bmatrix} \textsc{t} & \begin{bmatrix} \textsc{p} & \boxed{7} \end{bmatrix} \end{bmatrix} \end{bmatrix} \quad \rangle$$

Figure 21
Function σ for LTAG from Fig. 20

Now we have to define the semantic functions corresponding to the $f_{\gamma\ldots}$, i.e., the semantic denotations of the terms in our term algebra.[24] In order to compute the semantics of a node in the term tree, we need to know a) the unions of the semantic representations from the subtrees, b) the feature structure descriptions computed from the different subtrees, and c) the top fs-variables of the feature structure descriptions of the daughters. For a feature structure description δ linked to the semantic representation of an elementary tree, let $top(\delta)$ be the unique top variable. (E.g., in Fig. 21, $top(\delta_{\alpha_j}) = \boxed{0}, top(\delta_{\alpha_m}) = \boxed{1}\ldots$) We then define our semantic denotations as triples $\langle \overline{\sigma_\gamma}, \delta'_\gamma, top(\delta_\gamma) \rangle$ where the three components correspond to a)–c) above.

For a pair $\langle \sigma, \delta \rangle$ let $\overline{\sigma}$ be the result of applying to σ the meta-assignments following from δ.

Let us first illustrate the idea of the semantic denotations looking at some terms from our sample grammar.

(37) $[\![f_{\alpha_j}()]\!]_{sem} := \langle \overline{\sigma_{\alpha_j}}, \delta'_{\alpha_j}, top(\delta_{\alpha_j}) \rangle$ with

$$\delta'_{\alpha_j} \quad = \quad \delta_{\alpha_j} \bigwedge \{ \textsc{t}(p(top(\delta_{\alpha_j}))) = \textsc{b}(p(top(\delta_{\alpha_j}))) \,|\, p \text{ position in } \alpha_j \}$$

This means that the semantic denotation of the tree α_j with no other trees attaching to it consists of the top variable of the feature structure description δ_{α_j} (third component), the description δ_{α_j} conjoined with top-bottom equations for all nodes (second component), and the semantic representation obtained from applying the assignment arising from δ_{α_j} and the new equations to σ_{α_j}. The denotations for $f_{\alpha_m}()$ and $f_\beta()$ look similar.

24 For simplicity we do not consider global features (an extension to global features is straightforward).

(38) $[\![f_{\beta:\epsilon}(X)]\!]_{sem} := \langle\overline{\sigma_\beta \cup \sigma_X}, \delta', top(\delta_\beta)\rangle$ where $[\![X]\!]_{sem} = \langle\sigma_X, \delta_X, top_X\rangle$ with

$$\begin{array}{rcl} \delta' & = & \delta_\beta \wedge \delta_X \\ & & \wedge\textsc{t}(\epsilon(top(\delta_\beta))) = \textsc{t}(\epsilon(top_X)) \wedge \textsc{b}(\epsilon(top(\delta_\beta))) = \textsc{b}(f_X(top_X)) \\ & & \bigwedge\{\textsc{t}(p(top(\delta_\beta))) = \textsc{b}(p(top(\delta_\beta)))\,|\,p \neq \epsilon, p \text{ position in } \beta\} \end{array}$$

f_X gives the foot node position of the tree γ such that the term X has the form $f_{\gamma\ldots}(\ldots)$.

For terms denoting adjunctions or substitutions, things are more complex: The new description is the conjunction of the daughter descriptions plus additional equations corresponding to the adjunctions/substitutions and the final top-bottom identifications. In (38) these new equations identify the top of the adjunction site (address ϵ) in β with the top of the root of the adjoined tree, and the bottom of the adjunction site (address ϵ) in β with the bottom of the foot node of the adjoined tree. Furthermore, for all positions in β other than the adjunction site, top and bottom are identified. The new semantic representation is of course the union of the representations from the daughter denotations after application of the assignment computed from the new description.

The general definition of the semantic denotation is as follows:

- For all productions of the form $\gamma \to f_\gamma()$: Take a fresh instance $\langle\sigma_\gamma, \delta_\gamma\rangle$ of $\sigma(\gamma)$.

  $[\![f_\gamma()]\!]_{sem} := \langle\overline{\sigma_\gamma}, \delta'_\gamma, top(\delta_\gamma)\rangle$ with

  $$\delta'_\gamma = \delta_\gamma \bigwedge\{\textsc{t}(p(top(\delta_\gamma))) = \textsc{b}(p(top(\delta_\gamma)))\,|\,p \text{ position in } \gamma\}$$

- For all productions of the form $\gamma \to f_{\gamma:p_1,\ldots p_n}(\gamma_1, \ldots, \gamma_n)$: Without loss of generality let $p_1, \ldots, p_k$ $(0 \leq k \leq n)$ be the substitution node positions among the $p_1, \ldots p_n$.

  Take a fresh instance $\langle\sigma_\gamma, \delta_\gamma\rangle$ of $\sigma(\gamma)$.

  $[\![f_{\gamma:p_1,\ldots p_n}(X_1, \ldots, X_n)]\!]_{sem} := \langle\overline{\sigma_\gamma \cup \sigma_{X_1} \cup \cdots \cup \sigma_{X_n}}, \delta', top(\delta_\gamma)\rangle$ with

  $$\begin{array}{rcl} \delta' & = & \delta_\gamma \wedge \delta_{X_1} \wedge \ldots \delta_{X_n} \\ & & \bigwedge_{i=1}^{n} \textsc{t}(p_i(top(\delta_\gamma))) = \textsc{t}(\epsilon(top_{X_i})) \\ & & \bigwedge_{i=k+1}^{n} \textsc{b}(p_i(top(\delta_\gamma))) = \textsc{b}(f_{X_i}(top_{X_i})) \\ & & \bigwedge\{\textsc{t}(p(top(\delta_\gamma))) = \textsc{b}(p(top(\delta_\gamma)))\,|\,p \notin \{p_1, \ldots, p_n\}, \\ & & \qquad\qquad p \text{ position in } \gamma\} \end{array}$$

Let us go back to the example (33) *John sometimes sometimes loves Mary.* The term describing its derivation was $f_{\alpha_l:1,2,22}(f_{\alpha_j}(), f_{\beta:\epsilon}(f_\beta()), f_{\alpha_m}())$. The computation of the semantic denotation of this term is shown in Fig. 22.

We have shown that for each LTAG G, a GCFG can be defined that generates a term algebra describing the structural analyses of the strings generated by G in the sense of having syntactic denotations that are the derived trees yielding these strings. For this term algebra, there are

$$[\![f_{\beta}()]\!]_{sem} = \langle \sigma^1_{\beta}, {\delta^1_{\beta}}', \boxed{11} \rangle \text{ with}$$

$$\begin{aligned}
\sigma^1_{\beta} \quad &= \quad \boxed{l_3 : \mathsf{sometimes}'(\boxed{9}),\ \boxed{9} \geq \boxed{10}} \\
{\delta^1_{\beta}}' \quad &= \quad \textsc{p}(\textsc{b}(\epsilon(\boxed{11}))) = l_3 \wedge \textsc{p}(\textsc{t}(2(\boxed{11}))) = \boxed{10} \\
&\qquad \wedge \textsc{t}(\epsilon(\boxed{11})) = \textsc{b}(\epsilon(\boxed{11})) \wedge \textsc{t}(2(\boxed{11})) = \textsc{b}(2(\boxed{11}))
\end{aligned}$$

$$[\![f_{\beta:\epsilon}(f_{\beta}())]\!]_{sem} = \langle \sigma^2_{\beta}, {\delta^2_{\beta}}', \boxed{8} \rangle \text{ with}$$

$$\begin{aligned}
\sigma^2_{\beta} \quad &= \quad \boxed{l_2 : \mathsf{sometimes}'(\boxed{6}),\ l_3 : \mathsf{sometimes}'(\boxed{9}),\ \boxed{6} \geq \boxed{7},\ \boxed{9} \geq l_2} \\
{\delta^2_{\beta}}' \quad &= \quad \textsc{p}(\textsc{b}(\epsilon(\boxed{8}))) = l_2 \wedge \textsc{p}(\textsc{t}(2(\boxed{8}))) = \boxed{7} \\
&\qquad \wedge {\delta^1_{\beta}}' \\
&\qquad \wedge \textsc{t}(\epsilon(\boxed{8})) = \textsc{t}(\epsilon(\boxed{11})) \wedge \textsc{b}(\epsilon(\boxed{8})) = \textsc{b}(2(\boxed{11})) \\
&\qquad \wedge \textsc{t}(2(\boxed{8})) = \textsc{b}(2(\boxed{8}))
\end{aligned}$$

$$[\![f_{\alpha_j}()]\!]_{sem} = \langle \sigma_{\alpha_j}, \delta'_{\alpha_j}, \boxed{0} \rangle \text{ with}$$

$$\begin{aligned}
\sigma_{\alpha_j} \quad &= \quad \boxed{\mathsf{john}'(x)} \\
\delta'_{\alpha_j} \quad &= \quad \textsc{i}(\textsc{b}(\epsilon(\boxed{0}))) = x \\
&\qquad \wedge \textsc{t}(\epsilon(\boxed{0})) = \textsc{b}(\epsilon(\boxed{0}))
\end{aligned}$$

$$[\![f_{\alpha_m}()]\!]_{sem} = \langle \sigma_{\alpha_m}, \delta'_{\alpha_m}, \boxed{1} \rangle \text{ with}$$

$$\begin{aligned}
\sigma_{\alpha_m} \quad &= \quad \boxed{\mathsf{mary}'(y)} \\
\delta'_{\alpha_m} \quad &= \quad \textsc{i}(\textsc{b}(\epsilon(\boxed{1}))) = y \\
&\qquad \wedge \textsc{t}(\epsilon(\boxed{1})) = \textsc{b}(\epsilon(\boxed{1}))
\end{aligned}$$

$$[\![f_{\alpha_l:1,2,22}(f_{\alpha_j}(), f_{\beta:\epsilon}(f_{\beta}()), f_{\alpha_m}())]\!]_{sem} = \langle \sigma_{\alpha_l}, {\delta_{\alpha_l}}', \boxed{4} \rangle \text{ with}$$

$$\begin{aligned}
\sigma_{\alpha_l} \quad &= \quad \boxed{\begin{array}{l} \mathsf{john}'(x),\ \mathsf{mary}'(y),\ l_1 : \mathsf{love}'(x,y), \\ l_2 : \mathsf{sometimes}'(\boxed{6}),\ l_3 : \mathsf{sometimes}'(\boxed{9}), \\ \boxed{6} \geq l_1,\ \boxed{9} \geq l_2 \end{array}} \\
\delta'_{\alpha_l} \quad &= \quad \textsc{i}(\textsc{t}(1(\boxed{4}))) = \boxed{2} \wedge \textsc{i}(\textsc{t}(22(\boxed{4}))) = \boxed{3} \\
&\qquad \wedge \textsc{p}(\textsc{t}(2(\boxed{4}))) = \boxed{5} \wedge \textsc{p}(\textsc{b}(2(\boxed{4}))) = l_1 \\
&\qquad \wedge \delta'_{\alpha_j} \wedge {\delta^2_{\beta}}' \wedge \delta'_{\alpha_m} \\
&\qquad \wedge \textsc{t}(\epsilon(\boxed{0})) = \textsc{t}(1(\boxed{4})) \\
&\qquad \wedge \textsc{t}(\epsilon(\boxed{1})) = \textsc{t}(22(\boxed{4})) \\
&\qquad \wedge \textsc{t}(2(\boxed{4})) = \textsc{t}(\epsilon(\boxed{8})) \wedge \textsc{b}(2(\boxed{4})) = \textsc{b}(2(\boxed{8})) \\
&\qquad \wedge \textsc{t}(\epsilon(\boxed{4})) = \textsc{b}(\epsilon(\boxed{4})) \wedge \textsc{t}(21(\boxed{4})) = \textsc{b}(21(\boxed{4}))
\end{aligned}$$

Figure 22
Computation of the semantic denotation for (33) *John sometimes sometimes loves Mary*

(i) a function b giving the semantic denotations of atomic expressions $f()$ in the term algebra ($b(f()) := [\![f()]\!]_{sem}$),

(ii) and rules r_f specifying for each function (syntactic rule) f in the term algebra how to compute $[\![f(\gamma_1,\ldots,\gamma_n)]\!]_{sem}$ from $[\![\gamma_1]\!]_{sem}, \ldots [\![\gamma_n]\!]_{sem}$.

The relationship between the term algebra generated by the GCFG and the semantic denotation given by (i) and (ii) directly corresponds to a property of meaningful terms in a term algebra which Hodges establishes as one of four equivalent ways to characterize a compositional semantics (Hodges, 2001, p. 12, Theorem 4 (b)). Hodges defines a grammar as a set of expressions that can be obtained from atomic expressions by combining them according to a set of syntactic rules. The admissible syntactic combinations are captured by means of a grammatical term algebra, where a term t is grammatical if its value is defined and t is assigned a structural analysis. In our case, the GCFG generates all grammatical terms of the LTAG G. (i) and (ii) above determine the semantic denotations for each term. As a result, each term is *μ-meaningful* in the sense of Hodges' Theorem 4, and LTAG semantics fulfills the requirements of a compositional semantics.[25]

6.3 Summary: LTAG and Compositionality

As we have shown, since LTAG belongs to the class of linear context-free rewriting systems, it is possible to define a term algebra describing the syntactic composition (the substitutions and adjunctions) such that the semantics of a term $f(t_1,\ldots,t_n)$ depends only on f and the semantics of the subterms $t_1,\ldots,t_n$. In this sense, LTAG semantics is compositional.

For LRS, it is less obvious whether compositionality can be shown. We still need to identify the relevant structures that determine syntax and semantics. A starting point might be to look for some kind of functor-argument structure, similar to LTAG derivation trees. We leave this issue for further research.

7 Conclusion

We presented and compared two approaches to computational semantics, LRS and LTAG semantics. They are formulated in considerably different grammar frameworks but agree on the use of feature logics as a central mechanism in the specification of dependencies between the meaning of syntactic constituents and their components. This idea sets them apart from most of the current semantic theories of natural languages, which use the lambda calculus for specifying semantic composition.

Beyond their use of feature logic in semantic composition, we can identify a number of additional common characteristics of LTAG semantics and LRS: They both 1. use a Ty2 language for semantics; 2. allow underspecification (scope constraints, $\geq$, in LTAG semantics; component-of constraints, $\triangleleft$, in LRS); 3. use

25 Hodges (2001) proposes two alternative characterizations of a compositional semantics. The one we use here is the stronger, more restricted version, which we consider the more interesting notion of compositionality for linguistics.

logical descriptions for semantic computation, including the identification of the arguments of logical functors; 4. use the feature logics for specifying the upper and lower scope boundaries of quantificational operators; 5. are designed for computational applications. Due to these similarities, the analyses of several empirical phenomena and certain generalizations about the nature of semantic composition in natural languages can be formulated in almost identical ways in the two theories. Among these we focussed on the treatment of quantifier scope, and the mechanisms for identifying semantic arguments using attribute values rather than functional application with the lambda calculus.

7.1 Differences between LRS and LTAG

LRS structures are specified by means of a typed feature logic that supports the specification of all aspects of semantic composition. In fact, beyond semantic composition the feature logic can even take over the task of specifying the syntax of the semantic representation language, Ty2. This 'all-in-one' strategy is particularly attractive in combination with a grammar framework such as HPSG, because it makes it possible to investigate the syntax-semantics interface with a uniform model theory that applies to the syntactic structures of natural languages as well as to the semantic representations that are associated with them. From an abstract point of view, the intuitive function of LRS constraints on the meaning of utterances is 1. to specify the meaning contributions of words to the utterances in which they occur, and 2. to govern the way in which a particular mode of syntactic combination restricts the possibilities of putting the meaning contributions of lexical elements together.

In contrast to LRS in HPSG, LTAG is a modularly organized system with mathematically clearly separated subsystems. The syntactic framework is a tree-generating grammar. The elementary trees of the TAG system are linked to underspecified semantic representations that are augmented with feature logical expressions. The underspecification techniques that are applied to the semantic representations come from Hole Semantics (Bos 1995). A feature logic extension of the underspecified representations is responsible for the treatment of predicate-argument relationships and the scope of quantificational operators. Semantic computation adds feature value equations to the lexical specifications; these feature value equations are triggered by syntactic operations in the derivation of trees. The semantic representation which results from the syntactic derivation is an underspecified representation that awaits further disambiguation. The disambiguation procedure then leads to the interpretation(s) of a sentence in terms of sets of fully specified logical formulas. The disambiguation step can be viewed as another modular extension of the overall architecture and is performed according to the techniques of Hole Semantics.

Both LRS and LTAG semantics use feature logics to express predicate-argument relations and also to treat scope boundaries. However, from a technical perspective they do this in completely different ways. LRS as a whole is formulated in an expressive feature logic. Every element in an HPSG grammar with LRS is a logical description.[26] Linguists call these logical statements the 'grammar principles'. The linguistic expressions that are the subject of linguistic theorizing are perceived as configurations of entities licensed by the totality of logical statements in the grammar. In particular, the Ty2 terms that indicate the real world-meaning of linguistic expressions are also among the structures licensed by the grammar. We can say that they are in the denotation of the set of feature logical statements that constitute the grammar. The feature logic of LTAG semantics, on the other hand, is simply a restricted first order logic that serves solely to compute underspecified semantic representations. In this architecture the feature logic has nothing to do with the semantic representations linguists are interested in when they want to know the meaning of an utterance. Therefore the models of the feature structure descriptions never play a role in the LTAG architecture.

This difference is related to a much more general difference between Head-driven Phrase Structure Grammar and Tree Adjoining Grammar. HPSG takes a model theoretic view on natural languages. It sees the task of linguists in the logical specification of the well-formed expressions of a natural language with a uniform typed feature logic for all modules of grammar. To make this a feasible enterprise, the feature logic must be very flexible and expressive, because the kinds of principles linguists might want to express and the kinds of structures they might want to characterize cannot be anticipated. The starting point of LTAG is quite different. LTAG belongs to the class of mildly context-sensitive grammar formalisms, which is a class of formal systems with attractive computational properties for parsing. The entire architecture of the semantic framework in LTAG is guided by the desire to uphold its mildly context-sensitive nature. As a consequence, the feature logic extension of the core formalism is kept as weak as possible.

A key aspect of mildly context-sensitive grammars is that they are defined in a way as to guarantee the existence of an underlying context-free structure that uniquely determines both syntactic and semantic composition. The existence of such a context-free structure that links syntax and semantics justifies the claim that LTAG semantics is compositional. In HPSG a corresponding context-free structure might exist for some grammars or even for a particular class of HPSG grammars with LRS, but its existence is not guaranteed by the linguistic framework or the TFL formalism itself. It is the responsibility of the grammar writer to make sure that structures in the denotation of his grammar meet these or similar conditions.

The difference in the use of underspecification in LTAG semantics and in LRS that we discussed in this paper goes in the same direction. LRS employs partial descriptions of fully specified models. It follows immediately that two Ty2 term descriptions in grammatical constraints can denote the same formula in the model of an utterance in the denotation of the grammar. LTAG generates underspecified representations consisting of (pairwise distinct) sub-formulas linked by

26 Apart from the signature, which the linguist has to declare before writing the principles of grammar. See Section 2.1 for a short explanation.

scope constraints. If a type-identical formula is mentioned twice in grammatical descriptions, the two occurrences of the formula are necessarily distinct tokens. As distinct tokens they will stay distinct throughout all semantic computations during the derivation of a sentence. As first-class citizens of the grammar architecture, the underspecified representations of LTAG semantics also require the explicit definition of a disambiguation procedure.

The heavily parsing-oriented aspect of its overall architecture makes LTAG ultimately less flexible than LRS. Semantic concord cannot be the consequence of (partially) identifying semantic representations of sub-constituents in larger syntactic units. What we get in return are, once more, desirable computational properties: LTAG's semantic representations and the structures that the scope constraints impose on them guarantee that the constraints on underspecified representations resulting from semantic computation are just a slight extension of *normal dominance constraints*, which are known to be polynomially parsable. One of the goals of research in LTAG semantics is to show that its extension of normal dominance constraints still stays within the realm of polynomially parsable systems.

7.2 General Properties of Feature Logic-based Semantic Computation

Despite the architectural differences, the two frameworks for computational semantics share several important characteristics. We believe that these common features are general properties of frameworks with feature logic-based semantic computation. These frameworks can be distinguished in two respects from the influential frameworks in the immediate tradition of generative syntax such as Heim and Kratzer 1998:

First, they avoid functional application as the main method of semantic composition, which also means that they do not obligatorily pair up syntactic rules and semantic translation rules. One immediate consequence of not using functional application and similar operations as the mode of semantic composition is that the feature logic-based frameworks do not have to appeal to type shifting the semantics of lexical (or even phrasal) elements either, which is necessary in other theories in order to allow for all necessary functional applications in the course of semantic composition.[27] Similarly, type raising to the worst case is an artefact of the lambda calculus-based systems which is needed to be able to treat all elements of a given syntactic category in the process of semantic composition the same way, even though the basic semantic types of some lexical classes within a given syntactic category could in principle be much simpler. A famous example is the type raising of proper names to quantifiers to obtain a uniform type for all nominal phrases. The feature logic-based systems will always analyze every lexical element with the simplest available typing that is compatible with the empirically

27 The families of relations called *argument raising* and *value raising* in the Flexible Montague Grammar of Hendriks (1993) are one possible implementation of the technique of type shifting. In a lexicalized version Flexible Montague Grammar is also compatible with HPSG (Sailer 2003), which makes it possible to compare LRS and a semantics using type shifting operations within one grammar framework; see Richter (2004a) for details.

observed semantic behavior of the element.

Second, we saw that feature logic constraints permit a straightforward and flexible specification of scope boundaries. In particular, the use of feature logics in combination with underspecification avoids the introduction of otherwise unmotivated syntactic movement operations such as a tree-configurational mechanism of quantifier raising ('QR'). Consequently, our syntax is very surface-oriented; syntactic structure is assumed only for those units which can be argued for on syntactic grounds. Postulating a level of logical form (often called LF in the literature) to provide an additional layer of syntactic structure for computing the semantics which potentially introduces many empty categories is superfluous. Computational implementations may thus focus on syntactic representations that are known to be tractable more efficiently instead of having to deal with an inconvenient structural overhead with opaque properties, which, moreover, might change dramatically as the semantic theory develops.

It is interesting to note that the use of a standard semantic representation language such as Ty2 in our two frameworks is closely related to the use of feature logics in the combinatorics. Expressions of Ty2 become available as concrete structures due to the existence of independent mechanisms for percolating feature values on the derived syntax tree. In LTAG, the extended domain of locality of the elementary trees provides additional support for a direct specification of semantic representations in a standard higher-order logic.

The large and increasingly important research area of syntactic and semantic licensing requirements is an empirical domain in which a mathematically precise theory of features and feature values provides a firm basis for expressing elegant generalizations. Such contextual factors keep gaining ground in computational linguistics, where they appear in the form of collocation conditions, and they are also of interest in computational pragmatics. A typical syntactic example of a contextual licensing condition is the LTAG analysis of negative concord of Section 5.3; a related LTAG analysis of NPI (negative polarity item) licensing was proposed by Lichte and Kallmeyer (2006). Similarly, Richter and Soehn (2006) propose a theory of NPI licensing in HPSG that combines an LRS semantics with elements of a theory of idiomatic expressions. This theory of idioms was originally presented by Soehn (2006), who built on previous HPSG work on idioms that had produced a general architecture of a grammar module for the description of various kinds of collocations. Feature logic-based theories of context conditions in semantic representations of the type that we see in the semantic composition mechanisms of LRS and LTAG semantics may thus be viewed as a natural variant and new branch of feature logic-based theories of contextual licensing. In the course of these developments, semantic composition may finally become much more similar to other linguistic mechanisms than was assumed when it was set apart from other modules of grammar by exceptional composition mechanisms.

LTAG semantics as well as LRS have emphasized computational considerations of grammar design from the start. HPSG has always figured prominently in grammar implementation efforts; LRS has been implemented as the CLLRS module of the TRALE grammar development system (Penn and Richter 2004, 2005). However, unlike in our discussion of LTAG semantics, in which we have paid attention to many computational aspects, we largely ignored computational issues in LRS. The reason was that appreciating the relationship between CLLRS and LRS presupposes a much more detailed study of the model theory of LRS than is appropriate in the present context. The starting point of an efficient LRS implementation is the investigation of the intended models of an LRS grammar, and a thorough understanding of the interaction of the LRS constraints with syntactic constraints. On the basis of the models of the logical theory of LRS, CLLRS provides a separate constraint language specifically designed to reason over the intended class of models. Giving up the generality of a feature logic such as RSRL, CLLRS offers new constraint primitives such as component-of constraints and contribution constraints to support precisely those kinds of statements that are prominent in LRS principles. Since CLLRS can be defined as an extension of a standard feature logic, with which it may share meta-variables, the tight integration of syntax and semantics remains possible. The computational efficiency of the resulting system depends on the constraint handling system of CLLRS and its resolution procedures for underspecified descriptions of expressions of the higher-order logic. In current research, optimizing the computational behavior of CLLRS in response to practical experiences with implemented CLLRS grammars is an important issue.[28]

There exist several LTAG parsers and a large coverage grammar for English (XTAG Research Group 2001). However, this grammar does not include a semantics yet. More recently the Metagrammar (MG) tool developed in Nancy (Crabbé and Duchier 2004) was augmented by functions for the compact specification of a semantic module in a TAG. Gardent and Parmentier (2005) presented a parser for processing syntax and semantics which builds on the grammar format supported by the MG tool.

7.3 Open Questions

An obvious open question is whether HPSG with LRS is a compositional semantics. In Section 6 we sketched a non-trivial notion of compositionality for LTAG semantics which crucially relied on an underlying context-free structure linking the derived trees of the TAG component to underspecified semantic representations. For LRS, we still need to identify the relevant structures that determine a compositional relationship between the syntactic structures and the semantics. However, the fact that such structures exist in a closely related framework such as LTAG semantics indicates that finding them might not be as difficult as a first look at the LRS architecture might suggest.

Another interesting topic for further research is the problem of a more exact specification of the relationship between the feature logic attributes for semantic representations in LTAG and LRS. In the present paper, we focused on how the use of semantic feature values in the two systems leads to systems with very similar overall functions. Pursuing the technical details of the two systems further, it

28 For some more remarks on the relationship between LRS and CLLRS, see Section 2.3 above.

might even be possible to identify translation procedures from one framework to the other. The possibility of translating semantic analyses between two grammars implemented in different frameworks could be a very interesting application for computational grammar development.

As we emphasized throughout this paper, the limited generative capacity of the LTAG formalism is desirable because it guarantees a satisfactory computational behavior of LTAG grammars in the general case. It is not very surprising that this convenience comes at a price. The heavy restrictions on the expressivity of the framework occasionally cause problems: Some natural language phenomena cannot be described within traditional TAG. Most TAG extensions that have been proposed to remedy these problems involve the factoring of elementary trees into multicomponent sets (Weir, 1988; Rambow, 1994; Kallmeyer, 2005). This means that the lexical entries are no longer just single trees; they become sets of trees. If one of these lexical tree sets is used in a derivation step, then all of its elements must be added in this derivation step by substitution or adjunction. If the TAG framework is to reach a better empirical coverage in syntax and in semantics, extending the present syntax-semantics interface to these new TAG variants is an important issue for future research. For example, recent extensions of LTAG to multicomponent sets lead to greater success in the description of word order variability, and this extension in syntactic coverage leads to new questions at the syntax-semantics interface. One of them is the problem of capturing relations between word order and meaning in languages such as German that require a multicomponent extension of LTAG. First ideas on these topics are presented in Kallmeyer and Romero (2006). The connection of LTAG semantics to LRS might also be useful here, since the treatment of free word order languages has received much attention in HPSG.

We hope that our comparison of LRS and LTAG semantics will contribute to an increasing and fruitful interaction between the research communities from which the two theories originate. The comparison highlighted some important successful features that the two theories have in common. Due to the significant architectural differences between the two systems, comparing them also improved our understanding of those properties which distinguish both of them as feature logic-based approaches to semantic composition from the current alternative theories of the syntax-semantics interface. At the same time, the differences between LRS and LTAG semantics, in particular the differences in the motivation behind their major design decisions, are substantial enough to make one of the systems occasionally more successful in some tasks than the other. It is again the common core of the two systems that can be useful in transfering successful solutions in one system to the other for their mutual benefit.

Acknowledgments

For many long and fruitful discussions of many aspects of LTAG semantics and LRS, we would like to thank Timm Lichte, Wolfgang Maier, Maribel Romero, Manfred Sailer and Jan-Philipp Söhn. Furthermore, we are grateful to the participants of the workshop on *Typed Feature Structure Grammars* in Aalborg, Denmark, to the audience of our tutorial *Constraint-based Computational Semantics: LTAG and LRS* at the '13th International Conference on Head-Driven Phrase Structure Grammar 2006' in Varna, Bulgaria, and, in particular, to Anders Søgaard and Petter Haugereid and two reviewers for very helpful comments. A much shorter presentation of some aspects of the research reported in this paper appeared in Kallmeyer and Richter 2006. Without Janina Radó's help this paper would be much harder to read.

References

Althaus, Ernst, Duchier, Denys, Koller, Alexander, Mehlhorn, Kurt, Niehren, Joachim, and Thiel, Sven 2003. An efficient graph algorithm for dominance constraints. *Journal of Algorithms*, 48.1:194–219.

Bos, Johan 1995. Predicate logic unplugged. In Paul Dekker and Martin Stokhof (eds), *Proceedings of the 10th Amsterdam Colloquium*, 133–142.

Copestake, Ann, Flickinger, Dan, Pollard, Carl, and Sag, Ivan A. 2005. Minimal Recursion Semantics: An introduction. *Research on Language and Computation*, 3:281–332.

Crabbé, Benoit and Duchier, Denys 2004. Metagrammar Redux. In *International Workshop on Constraint Solving and Language Processing*, 32–47, Copenhagen.

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

Fuchss, Ruth, Koller, Alexander, Niehren, Joachim, and Thater, Stefan 2004. Minimal Recursion Semantics as Dominance Constraints: Translation, Evaluation, and Analysis. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL'04)*, 247–254, Barcelona, Spain.

Gallin, Daniel 1975. *Intensional and Higher-Order Modal Logic*. North-Holland, Amsterdam.

Gardent, Claire and Parmentier, Yannick 2005. Large scale semantic construction for Tree Adjoining Grammars. In *Logical Aspects of Computational Linguistics, LACL'2005*, 131–146, Bordeaux.

Heim, Irene and Kratzer, Angelika 1998. *Semantics in Generative Grammar*. Blackwell.

Hendriks, Herman 1993. *Studied Flexibility*, (= *ILLC Dissertation Series 1995-5*). Institute for Logic, Language and Computation, Amsterdam.

Hodges, Wilfrid 2001. Formal Features of Compositionality. *Journal of Logic, Language, and Information*, 10:7–28.

Johnson, Mark 1988. *Attribute-value logic and the theory of grammar*, (= *CSLI Lecture Notes Series*). Chicago: University of Chicago Press.

Johnson, Mark 1990. Expressing disjunctive and negative feature constraints with classical first-order logic. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, 173–179.

Joshi, Aravind K. 1985. Tree adjoining grammars: How much contextsensitivity is required to provide reasonable structural descriptions? In D. Dowty, L. Karttunen, and A. Zwicky (eds), *Narural Language Parsing*, 206–250. Cambridge University Press.

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

Kallmeyer, Laura 2005. Tree-local multicomponent tree adjoining grammars with shared nodes. *Computational Linguistics*, 31.2:187–225.

Kallmeyer, Laura and Joshi, Aravind K. 2003. Factoring Predicate Argument and Scope Semantics: Underspecified Semantics with LTAG. *Research on Language and Computation*, 1.1–2:3–58.

Kallmeyer, Laura and Richter, Frank 2006. Constraint-based computational semantics: A comparison between LTAG and LRS. In *Proceedings of the 8th International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+8)*, 109–114, Sydney, Australia: The Association for Computational Linguistics.

Kallmeyer, Laura and Romero, Maribel 2006. Quantifier Scope in German: An MCTAG Analysis. In *Proceedings of the 8th International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+8)*, 73–80, Sydney, Australia: The Association for Computational Linguistics.

Kallmeyer, Laura and Romero, Maribel 2007. Scope and Situation Binding in LTAG using Semantic Unification. *Note:* To appear in *Research on Language and Computation.*

Kamp, Hans and Reyle, Uwe 1993. *From Discourse to Logic*, (= *Studies in Linguistics and Philosophy*). Dordrecht, Boston, London: Kluwer.

Kasper, Robert 1997. Semantics of recursive modification. *Note:* Unpublished Manuscript. The Ohio State University.

Koller, Alexander, Niehren, Joachim, and Thater, Stefan 2003. Bridging the gap between underspecification formalisms: Hole semantics as dominance constraints. In *Meeting of the European Chapter of the Association of Computational Linguistics*, 195–202.

Koller, Alexander, Niehren, Joachim, and Treinen, Ralf 2001. Dominance Constraints: Algorithms and Complexity. In M. Moortgat (ed), *Proceedings of the Third International Conference on Logical Aspects of Computational Linguistics (LACL'98), Grenoble, France*, (= *Lecture Notes in Computer Science*, 2014/2001), 106–125, Berlin / Heidelberg: Springer.

Kupść, Anna 2000. *An HPSG Grammar of Polish Clitics*. Ph.d. thesis, Warszawa, Poland: Institute of Computer Science, Polish Academy of Sciences and Université Paris 7.

Lichte, Timm and Kallmeyer, Laura 2006. Licensing German Negative Polarity Items in LTAG. In *Proceedings of the 8th International Workshop on Tree Adjoining Grammar and Related Formalisms*, 81–90, Sydney, Australia: The Association for Computational Linguistics.

Moore, Robert C. 1989. Unification-based semantic interpretation. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, 33–41.

Nederhof, Mark-Jan 1997. Solving the correct-prefix property for TAGs. In T. Becker and H.-U. Krieger (eds), *Proceedings of the Fifth Meeting on Mathematics of Language*, 124–130, Schloss Dagstuhl, Saarbrücken.

Nerbonne, John 1992. Constraint-based semantics. In Paul Dekker and Martin Stokhof (eds), *Proceedings of the Eighth Amsterdam Colloquium*, 425–444. Institute for Logic, Language and Information.

Penn, Gerald and Richter, Frank 2004. Lexical Resource Semantics: From theory to implementation. In Stefan Müller (ed), *Proceedings of the 11th International Conference on Head-Driven Phrase Structure Grammar*, 423–443. CSLI Publications.

Penn, Gerald and Richter, Frank 2005. The other syntax: Approaching natural language semantics through logical form composition. In Henning Christiansen, Peter Rossen Skadhauge, and Jørgen Villadsen (eds), *Constraint Solving and Language Processing. First International Workshop, CSLP 2004, Roskilde, Denmark, September 1-3, 2004, Revised Selected and Invited Papers*, (= *Lecture Notes in Computer Science*, 3438), 48–73. Springer.

Pollard, Carl and Sag, Ivan A. 1994. *Head-Driven Phrase Structure Grammar*, (= *Studies in Contemporary Linguistics*). Chicago, London: The University of Chicago Press.

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

Richter, Frank 2004a. Foundations of Lexical Resource Semantics. Habilitation Thesis. Eberhard Karls Universität Tübingen.

Richter, Frank 2004b. *A Mathematical Formalism for Linguistic Theories with an Application in Head-Driven Phrase Structure Grammar*. Phil. dissertation (2000), Eberhard Karls Universität Tübingen.

Richter, Frank and Sailer, Manfred 2001. On the left periphery of German finite sentences. In W. Detmar Meurers and Tibor Kiss (eds), *Constraint-Based Approaches to Germanic Syntax*, 257–300. Stanford: CSLI Publications.

Richter, Frank and Sailer, Manfred 2004. Basic concepts of Lexical Resource Semantics. In Arnold Beckmann and Norbert Preining (eds), *ESSLLI 2003 – Course Material I*, (= *Collegium Logicum*, 5), 87–143. Kurt Gödel Society Wien.

Richter, Frank and Soehn, Jan-Philipp 2006. 'Braucht niemanden zu scheren': A survey of NPI licensing in German. In Stefan Müller (ed), *Proceedings of the 13th International Conference on Head-Driven Phrase Structure Grammar*, 421–440. CSLI Publications.

Sailer, Manfred 2003. Combinatorial semantics and idiomatic expressions in Head-Driven Phrase Structure Grammar. Phil. Dissertation (2000). Arbeitspapiere des SFB 340. 161, Eberhard Karls Universität Tübingen.

Sailer, Manfred 2004. Propositional relative clauses in German. In Stefan Müller (ed), *Proceedings of the 11th International Conference on Head-Driven Phrase Structure Grammar*, 223–243. CSLI Publications.

Sailer, Manfred 2006. "Don't Believe" in Underspecified Semantics. Neg Raising in Lexical Resource Semantics. In Olivier Bonami and Patricia Cabredo Hofherr (eds), *Empirical Issues in Formal Syntax and Semantics*, 375–403. Presses de l'Université de Paris-Sorbonne.

Savitch, Walter J., Bach, Emmon, Marxh, William, and Safran-Naveh, Gila (eds) 1987. *The Formal Complexity of Natural Language*, (= *Studies in Linguistics and Philosophy*). Dordrecht, Holland: Reidel.

Schabes, Yves and Joshi, Aravind K. 1988. An Earley-type parsing algorithm for Tree Adjoining Grammars. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, 258–269.

Shieber, Stuart M. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343. *Note:* Reprinted in Savitch et al. (1987).

Soehn, Jan-Philipp 2006. *Über Bärendienste und erstaunte Bauklötze - Idiome ohne freie Lesart in der HPSG*. Peter Lang (Frankfurt/M). *Note:* Phil. dissertation. Friedrich-Schiller-Universität Jena.

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

Vijay-Shanker, K. and Joshi, Aravind K. 1985. Some computational properties of Tree Adjoining Grammars. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, 82–93.

Vijay-Shanker, K. and Joshi, Aravind K. 1988. Feature structures based tree adjoining grammar. In *Proceedings of COLING*, 714–719, Budapest.

Vijay-Shanker, K. and Weir, David J. 1993. Parsing some constrained grammar formalisms. *Computational Linguistics*, 19.4:591–636.

Weir, David J. 1988. *Characterizing mildly context-sensitive grammar formalisms*. PhD thesis, University of Pennsylvania.

XTAG Research Group 2001. A Lexicalized Tree Adjoining Grammar for English. Technical report, Institute for Research in Cognitive Science, Philadelphia. *Note:* Available from `ftp://ftp.cis.upenn.edu/pub/xtag/release-2.24.2001/tech-report.pdf`.