

Typed Feature Structures for Semantic Composition: A Comparison between LRS and LTAG

In this paper we 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 1995), the second is Lexical Resource Semantics (LRS, Richter and Sailer 2004) and was first defined in HPSG. The two frameworks have several common characteristics that make them easy to compare: They both 1. use a Ty(2) language for semantic representations; 2. allow underspecification: LTAG uses scope constraints \geq while LRS provides component-of constraints \triangleleft ; 3. use feature structure descriptions for computing semantic representations; 4. are designed for computational applications.

Introduction Except for a few early and largely informal explorations of the relationship between semantic representations in unification-based frameworks using typed feature structures (TFSs) and the lambda calculus-based Montague Grammar of most formal semantic research in linguistics (c.f. Moore 1989; Nerbonne 1992), there has been little or no explicit connection between these two techniques for semantic representations. Sailer (2003) finally proved that a feature logic for HPSG such as RSRL is sufficiently expressive to encode a higher-order logic such as Intensional Logic or Ty(2), and the combinatorial system of the lambda calculus. This means that it is mathematically possible to embed a categorial semantics such as Flexible Montague Grammar completely within a grammar of TFSs. However, undecidability results make it unattractive to integrate RSRL or alternative feature logics that are expressive enough for a direct 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 Ty(2) and basic operations of the lambda calculus such as beta reduction.

Unification-based LTAG semantics (Kallmeyer and Romero 2005) and LRS draw different conclusions from this situation and use feature logics of different expressivity for similar purposes in semantic composition. Like Minimal Recursion Semantics (MRS, Copestake et al. 2003) 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 Ty(2). To overcome the tension between the expressivity needed in a feature logic for the specification of the combinatorics and representations of Ty(2) and the requirements of effective computation, LTAG semantics and LRS distribute various tasks differently among a feature logical specification, mathematical tools from beyond feature logics, and task-specific computational implementation. We sketch some important aspects of their respective architectures with the simple example in (1a). The (simplified and extensional) logical representation both frameworks assign to it is shown in (1b).

- (1) a. John sometimes laughs.
b. `sometimes(laugh(john))`

LRS In LRS the feature logic specifies the entire grammar, including well-formed Ty(2) terms as semantic representations, and their mode of composition. Fig. 2 sketches the analysis of (1a). Each word lexically 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). Feature percolation principles identify INCONT and EXCONT along head projections and collect the elements of the PARTS lists of the daughters at each phrase. The combination of the adjunct with a verbal projection introduces two component-of constraints: The EXCONT of *sometimes* must be within the EXCONT of *laughs*, and the INCONT of *laughs* must be in the scope of *sometimes*. The semantic argument of *laughs* is identified by subcategorization (not shown in Fig. 2). A closure condition requires that the semantic representation of an utterance use up all and only the PARTS contributions of all signs, which finally yields (1b).

LTAG In LTAG the typed feature structure descriptions only specify how to combine semantic representations, they do not encode the Ty(2) terms that receive a semantic interpretation. The feature identifications have the effect of performing functional applications. Fig. 1 sketches the analysis of (1a). The syntactic elements in LTAG are so-called elementary trees that are combined by substitutions and adjunctions (replacing a leaf/an internal node with a new tree). The derivation of (1a) is shown on the left. The derivation tree records the way the elementary trees were combined. In Fig. 1, the *John* and *sometimes* trees are both added to the *laughs* tree, at node addresses *np* and *vp*. Semantic computation is done on the derivation tree. Each elementary tree is linked to a semantic representation (a set of Ty(2) formulae and scope constraints). A semantic representation is equipped with a semantic feature structure description. Semantic computation consists of certain feature value identifications between mother and daughter nodes in the derivation tree. The feature value equations in Fig. 1 are indicated with dotted links. They give the identifications $\boxed{1} = \text{john}$ and $\boxed{4} = l_1$ which leads to σ from Fig. 1 when building the union of the semantic representations. Solving the scope constraint finally yields (1b).

As illustrated by this example, the LTAG feature structure descriptions do not encode the semantic expressions one is interested in. They only encode their contributions to functional applications, i.e., they state which elements are contributed as possible arguments for other semantic expressions and which arguments need to be filled. They thereby simulate lambda abstraction and functional application while assembling the semantic representations. Thus, a restricted first order logic is sufficient for the feature structure descriptions needed. The semantic representations generated in LTAG are underspecified Ty(2) terms similar to normal dominance constraints, which are known to be polynomially solvable.

