Computational Semantics Tools for Glue Semantics

Mark-Matthias Zymla
University of Konstanz
mark-matthias.zymla
@uni-konstanz.de

Mary Dalrymple
University of Oxford
mary.dalrymple
@ling-phil.ox.ac.uk

Agnieszka Patejuk
Institute of Computer Science
Polish Academy of Sciences
aep@ipipan.waw.pl

Abstract

This paper introduces a suite of computational semantic tools for Glue Semantics, an approach to compositionality developed in the context of Lexical Functional Grammar (LFG), but applicable to a variety of syntactic representations, including Universal Dependencies (UD). The three tools are: 1) a Glue Semantics prover, 2) an interface between this prover and a platform for implementing LFG grammars, and 3) a system to rewrite and add semantic annotations to LFG and UD syntactic analyses, with a native support for the prover. The main use of these tools is computational verification of theoretical linguistic analyses, but they have also been used for teaching formal semantic concepts.

1 Introduction

This paper introduces a suite of tools related to Glue Semantics (Dalrymple 1999, Asudeh 2022, 2023), an approach to compositionality based on the idea of resource sensitivity, for a wider computational semantic audience.[1] On this approach, the compositional process is not necessarily determined directly by phrasal constituency (as in, for example, Heim and Kratzer 1998), but is rather guided by pairing (partial) semantic representations with linear logic formulas referring to parts of syntactic representations. While Glue Semantics has been most extensively applied in the context of Lexical Functional Grammar (LFG; Kaplan and Bresnan 1982, Bresnan et al. 2015, Dalrymple et al. 2019, Dalrymple 2023), it has also been successfully combined with other syntactic formalisms, including Universal Dependencies (e.g., Gotham and Haug 2018), Lexicalized Tree Adjoining Grammar (Frank and van Genabith 2001), Head-driven Phrase Structure Grammar (Asudeh and Crouch 2002), and Minimalism (Gotham 2018). It is compatible with various formal meaning representations, including predicate logic with lambdas and DRT (Kamp and Reyle 1993).

Within LFG, computational research evolves around the Xerox Linguistics environment (XLE; Crouch et al. 2017), a platform that has been primarily tailored towards the modeling of syntax. Although XLE grammars are being developed all across the world, the investigation of semantic issues in LFG from a computational perspective received impetus with the introduction of an early version of the Glue Semantics Workbench (GSWB; Meßmer and Zymla 2018).[2] This paper presents new contributions to GSWB and two recently developed resources that make use of it.[3]

The central resource presented in this paper is the Glue Semantics Workbench (GSWB), a modular system for calculating Glue Semantics (henceforth, Glue) proofs. It provides three different Glue provers and is designed to permit the implementation of additional provers based on varying linear logic fragments and meaning languages (e.g., predicate logic with lambdas, DRT, etc.).

The second tool, XLE+Glue, implements an interface between GSWB and XLE.[4] This tool allows users to specify semantic contributions of lexical items and syntactic rules in XLE grammars, which can then be fed into GSWB for semantic calculation. The system has been mainly developed to explore what is called a "co-descriptive approach" to Glue (explained in §2.2). XLE+Glue also illustrates the possibility of GSWB to work with different meaning languages.

[1] Early versions of two of these tools have been presented LFG-internally, the third is presented here for the first time.

[2] Earlier works in computational semantics related to LFG include Asher and Wada 1988, Crouch 2005, Crouch and King 2006, Bobrow et al. 2007, Lev 2007.

[3] See §3 for links to Github repositories of these resources.

[4] The original idea is presented in Dalrymple et al. 2020. This paper presents further developments.

Proceedings of the 16th International Conference on Computational Semantics, pages 199–217

The third tool presented in this paper is a system for linguistic graph expansion and rewriting (LiGER). It is inspired by the original XLE transfer system, which was initially used for machine translation (Frank 1999) and later mainly for semantic parsing (Crouch 2005, Crouch and King 2006), but also as a full-fledged reasoning engine (Bobrow et al. 2007), indicating its versatility. LiGER has been developed because the original transfer component of XLE is no longer supported by XLE. Like the original transfer system, LiGER can be used to enrich XLE analyses with information from other linguistic resources. With respect to semantic analysis, it provides the possibility of exploring the second major approach to deriving Glue representations, "description-by-analysis" (see §2.2), and thus complements XLE+Glue.

Overall, the tools presented here allow researchers to experiment with different settings within the Glue framework, including the choice of a suitable linear logic fragment, the choice of meaning language, and the choice of co-description vs. description-by-analysis approaches to deriving meaning representations. The goal of this paper is to illustrate the capabilities of these tools and how they can be used for verifying theoretical analyses and for exploring formal semantic concepts. Section 2 explains the LFG architecture, focusing on two aspects: the projection structure and Glue. Section 3 describes the three tools in more detail, while §4 mentions some use cases. Section 5 concludes.

2 Background

Within the LFG community, the development of XLE grammars, as well as associated resources such as treebanks, is carried out mainly in the scope of the Parallel Grammar (ParGram) project (Butt et al. 2002, Sulger et al. 2013). Such grammars have been developed for a wide variety of typologically diverse languages, demonstrating the cross-linguistic and formal validity of LFG's (morpho)syntactic component.[5] The work presented in this paper aims to facilitate extending such syntac-

[5]Some of the grammars that are publicly available for testing via INESS (`https://clarino.uib.no/iness/xle-web`; Rosén et al. 2012), and some that are not yet publicly available (in parentheses), are:

(i) Larger grammars for English, German, French, Norwegian, and Polish (as well as Chinese and Japanese)

(ii) Smaller grammars for Georgian, Indonesian, Malagasy, Turkish, Welsh, Wolof, and Urdu (as well as Greek and Hungarian)

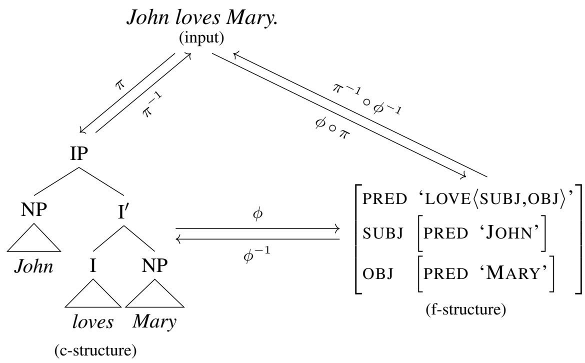

Figure 1: LFG correspondence structure as implemented in XLE

tic work to semantics. This section first describes the underlying concepts of the LFG formalism, and then the LFG approach to semantics.

2.1 LFG projection architecture

LFG is developed around the idea of mutually constraining parallel representations. The two syntactic representations, implemented in XLE, are c(onstituent)-structure and f(unctional)-structure (cf. Figure 1). While c-structure encodes the surface structure in terms of a constituent parse that preserves linear word order, f-structure encodes functional information, primarily grammatical functions and morphosyntactic features, in an attribute-value matrix. Grammars encode both structures simultaneously. C-structures are constrained by phrase structure rules (as in the first row in (1)), with categories specified in lexical entries (see "N" in (2)). F-structures are constrained using functional annotations (usually equations) in phrase-structure rules and lexical entries.

(1) $\text{IP} \rightarrow \underset{(\uparrow \text{SUBJ}) = \downarrow}{\text{NP}} \quad \underset{\uparrow = \downarrow}{\text{I}'}$

(2) John N $(\uparrow \text{PRED}) = \text{'JOHN'}$

This simultaneous specification of two levels is called local co-description (Bresnan et al. 2015). In this architecture, the different structures are related via projection functions. This ensures structural correspondence between different levels of analysis and entails mutual accessibility of projections.

Consider Figure 1. The c-structure is generated from the input via the π-projection – a constituent parse. The f-structure is specified based on constraints that are annotated on c-structure nodes and specified in the lexicon. The corresponding mapping function from c- to f-structure is encoded in the ϕ-projection. The mapping from the input to f-structure is a combination of the two projections: the $\phi \circ \pi$ mapping.[6] LFG also assumes an inverse of each mapping function; while such inverse mappings are less often discussed, they play a role in the possibility of generation, as explored in early work within XLE.[7]

The next section discusses two ways of integrating semantics into the LFG projection architecture.

2.2 Semantics in LFG

Adding any projection that preserves the kind of bi-directionality described in the previous section to this framework is a challenge, and this also holds for the semantic projection. It is beyond the scope of this paper to delve into all the fine details of semantics in LFG, but we briefly address some of the main challenges that the tools presented here may help address. For this purpose, we first provide a quick introduction to Glue, which is a semantic formalism that has been developed for LFG but is generally applicable to different linguistic frameworks, making the present tools interesting for projects that go beyond LFG as well.

The formalism of Glue is modeled around the idea of resource sensitivity (Dalrymple 1999). Resource management is ensured by the use of a fragment of the resource-sensitive *linear logic* (Girard 1987) that is paired with a meaning representation, forming a *meaning constructor*. Example (3) shows meaning constructors for all words in *John loves Mary*. In this example, each word in the sentence introduces a single meaning constructor.[8] In (3), the meaning representation j of the subject *John* is associated with the resource g, the meaning m of the object *Mary* is associated with the resource h, and the more complex meaning of the verb *loves* is associated with the linear logic formula $g \multimap (h \multimap f)$. This formula uses the linear implication $\multimap$ to indicate that it requires the resource in its antecedent to produce the resource in its consequent. Thus, by consuming the subject resource g, we can produce the resource $(h \multimap f)$, which in turn consumes the object resource h to produce the final result f corresponding to the meaning of the full sentence; see the full Glue proof in (4). In line with the Curry-Howard isomorphism (CHI), modus ponens on the linear logic side corresponds to function application on the meaning side.

(3)
John	$j : g$
Mary	$m : h$
loves	$\lambda x.\lambda y.love(x,y) : g \multimap (h \multimap f)$

(4)
$$\dfrac{\dfrac{\lambda x.\lambda y.love(x,y) : g \multimap (h \multimap f) \quad j : g}{\lambda y.love(j,y) : h \multimap f} \quad m : h}{love(j,m) : f}$$

This relation between resource consumption and semantic composition is the foundation of Glue. As long as the CHI is preserved, different fragments of linear logic can be paired with different meaning representations, resulting in two dimensions of variation.

Additionally, as mentioned above, Glue has been combined with different syntactic theories, assuming different approaches to the syntax/semantics interface. In this paper, we briefly discuss the two main such approaches explored in LFG: co-description (Kaplan and Wedekind 1993) and description by analysis (Halvorsen and Kaplan 1988).

In the co-descriptive approach, particular to LFG, meaning constructors are introduced in lexical entries (and, possibly, grammatical rules), in parallel with categorical and functional information. This is illustrated on the left-hand side of Figure 2. The lexical entries use the $\uparrow$-variable to refer to specific elements in the f-structure. The nominal entries specify the semantics for the substructures they contribute (corresponding to g and h at the bottom of the figure). The inflected verb uses the functional descriptions $(\uparrow \text{SUBJ})$ and $(\uparrow \text{OBJ})$ to retrieve these substructures via their indices to form the meaning constructor of the verb.

On the other hand, description-by-analysis uses a fully assembled f-structure as input to derive meaning constructors. This is usually done by rules that match partial f-structure descriptions and introduce corresponding meaning constructors; see the right-hand side of Figure 2. There, `#f`, `#g`, and `#h` are variables referring to f-structures (see the corresponding f, g, and h at the bottom of Figure 2),

[6]The projection structure is usually depicted in linear order on a form-to-meaning mapping (Kaplan 1995, Asudeh 2006); however, to avoid directionality, we present the projection structure as a (complete) graph, with no order between nodes since the order might well change depending on specific processing tasks (Jackendoff 2010).

[7]Both parsing and generation are in principle undecidable in LFG and require additional constraints on the formalism to be made workable (Kaplan and Bresnan 1982). See Wedekind 1988 for early LFG work on generation from a separate semantic structure, i.e., involving an inverse of the mapping from semantics to the surface string, and Wedekind and Kaplan 2020 and Kaplan and Wedekind 2019 for more recent work. Such work motivates the existence of inverse projection mappings, and such mappings are assumed in this paper.

[8]This is not a rule; a word can introduce any number of meaning constructors, and meaning constructors may also be introduced by syntactic rules.

co-description:

John	N	$(\uparrow \text{PRED}) = \text{`JOHN'}$ $j : \uparrow$
Mary	N	$(\uparrow \text{PRED}) = \text{`MARY'}$ $m : \uparrow$
loves	I	$(\uparrow \text{PRED}) = \text{`LOVE}\langle\text{SUBJ,OBJ}\rangle\text{'}$ $\lambda x.\lambda y.love(x,y) :$ $(\uparrow \text{SUBJ}) \multimap ((\uparrow \text{OBJ}) \multimap \uparrow)$

description-by-analysis:

```
#f SUBJ #g PRED %g  ==>  #g GLUE %g : #g.

#f OBJ #h PRED %h  ==>  #h GLUE %h : #h.

#f SUBJ #g & #f OBJ #h & #f PRED %f
    ==>  #f GLUE %f : #g -o (#h -o #f).
```

result (for both approaches):

$$f\begin{bmatrix}\text{PRED} & \text{`LOVE}\langle\text{SUBJ,OBJ}\rangle\text{'}\\ \text{SUBJ} & g\begin{bmatrix}\text{PRED} & \text{`JOHN'}\end{bmatrix}\\ \text{OBJ} & h\begin{bmatrix}\text{PRED} & \text{`MARY'}\end{bmatrix}\end{bmatrix} \qquad \begin{array}{l} j : g \\ m : h \\ \lambda x.\lambda y.love(x,y) : g \multimap (h \multimap f)\end{array}$$

Figure 2: Co-descriptive lexicon vs. description-by-analysis rules

used as resources in the linear logic side of the introduced meaning constructors, while %f, %g, and %h refer to the corresponding PRED values and are used in the meaning sides. The first two rules introduce resources for the subject and the object, while the rule for the verb specifies the meaning constructor in a way similar to the co-descriptive approach. This means that both approaches generally map the same kind of nodes onto meaning constructors as indicated by the f-structure and the corresponding instantiated meaning constructors to its right (see the indices g, h, and f there).

Both co-description and description-by-analysis are currently in use in theoretical LFG work; it might well be the case that it is best to combine the two approaches to deal with different kinds of semantic phenomena.[9] The present tool suite is designed to allow for this.

2.3 Semantic autonomy

The flexibility in modeling the syntax/semantics interface is due to one of the key advantages of Glue Semantics: a high level of semantic autonomy (Asudeh 2004). As Figure 2 suggests, semantic composition does not rely on word order – it relies instead on more general concepts such as grammatical functions. Furthermore, semantic autonomy provides a purely semantic treatment of quantification, one that is independent of syntactic considerations such as, for instance, quantifier raising (Heim and Kratzer 1998). This is illustrated in Figure 3 on the basis of quantifier scope ambiguity. For a more in-depth discussion on quantifier scope, see, e.g., Gotham (2019, 2021), Dalrymple et al. (1999). Semantic autonomy provides a unique view on formal semantics that can be explored using the tools presented in this paper.

2.4 Related work

The tools presented here are inspired by work in grammar engineering (e.g., Flickinger et al. 2017) and semantic annotation (e.g., Basile et al. 2012). There is also some overlap with toolkits such as the NLTK (Bird et al. 2009). The main difference is a focus on Glue Semantics and its compositional properties, as well as its relation to various syntactic approaches, especially LFG and Universal Dependencies (UD). The present tools have not yet been employed in large-scale grammar engineering efforts, but rather at the interface between formal and computational linguistics to verify analyses (but see Zymla et al. 2025, Findlay et al. 2023).

3 The tools

The ParGram project provided a cross-linguistically informed approach to syntactic and semantic parsing, though the latter was mostly worked out for English, while concrete implementations for other languages were of limited scope. This is largely due to the fact that the semantics relied heavily on various external resources that were not available cross-linguistically. Semantic parsing relied on ordered rewriting rules implemented as part of a transfer system in XLE (Crouch and King 2006, Bobrow et al. 2007). Another important issue addressed with the present tools is that the existing transfer system is neither publicly available nor compatible with the currently available XLE releases provided by

[9] It seems that description-by-analysis may be more suitable for the semantic interpretation of functional features, whereas phenomena involving information structure are more suitably encoded in a co-descriptive fashion (Andrews 2008).

Every monkey likes a banana.

a. $\lambda x.\lambda y.\text{like}(x,y)$:
$m_\sigma \multimap (b_\sigma \multimap f_\sigma)$

b. $\lambda P.\forall x[\text{monkey}(x) \rightarrow P(x)]$:
$(m_\sigma \multimap f_\sigma) \multimap f_\sigma$

c. $\lambda Q.\exists y[\text{banana}(y) \wedge Q(y)]$:
$(b_\sigma \multimap f_\sigma) \multimap f_\sigma$

$$\dfrac{\lambda P.\forall x[\text{monkey}(x) \rightarrow P(x)] : (m_e \multimap f_t) \multimap f_t \qquad \dfrac{\dfrac{\dfrac{\dfrac{[X : m_e]^1 \qquad \lambda x.\lambda y.\text{like}(x,y) : m_e \multimap (b_e \multimap f_t)}{\lambda y.like(X,y) : b_e \multimap f_t}\multimap_E \qquad \lambda Q.\exists y[\text{banana}(y) \wedge Q(y)] : (b_e \multimap f_t) \multimap f_t}{\exists y[\text{banana}(y) \wedge \text{like}(X,y)] : f_t}\multimap_E}{\lambda x.\exists y[\text{banana}(y) \wedge \text{like}(x,y)] : m_e \multimap f_t}\multimap_{I,1}}{}}{\forall x[\text{monkey}(x) \rightarrow \exists y[\text{banana}(y) \wedge \text{like}(x,y)]] : f_t}\multimap_E$$

Figure 3: **Quantification in Glue:** Quantifier scope falls out naturally from the properties of linear logic, giving appropriate typings. Implication introduction (lambda abstraction) allows to capture flexible scope configurations (the alternative reading for this example is shown in Figure 7 in appendix A).

the University of Konstanz.[10] The tools described below are open source and compatible with various systems, including XLE, and they are designed to be useful in theoretical linguistic work as well as in investigation of general issues of integrating semantics into the LFG projection architecture.

3.1 The Glue Semantics Workbench

The Glue Semantics Workbench (GSWB)[11] is a modular system for deriving Glue proofs. To this end, it provides the possibility of using different provers as well as different input formats for meaning languages, with a built-in parser for formulas based on typed lambda-calculus, and support for meaning representations written in Prolog (in particular, those developed on the basis of Blackburn and Bos 2005, i.e., untyped lambda calculus and λ-DRT). Furthermore, functionality was recently added that allows users to interface GSWB with NLTK's (Bird et al. 2009) semantic capabilities (Klein 2006).

GSWB uses a string format for linear logic and semantic representations that is close to actual Glue semantic representations, as illustrated in (5).

(5)
```
john : g
mary : h
[/x_e.[/y_e.love(x,y)]] :
    (g -o (h -o f))
```

There, the meaning side is on the left of :, and the linear logic side is on the right. The entry for the verb shows the encoding of complex linear logic formulas and lambda expressions which can be computed using the basic tools for function application (Blackburn and Bos 2005).

To ensure flexibility, the meaning side of a meaning constructor can be replaced with any semantic representation that can be encoded as a string. In this case, users can specify procedures that preserve CHI, by implementing function application directly in GSWB or by feeding the output to a separate system.[12] The latter option is used to integrate GSWB with a modified version of the DRT part of Boxer tools (Bos 2008; based on Blackburn and Bos 2005) and with NLTK (Findlay et al. 2023).

GSWB contains three different provers for the implicational fragment of linear logic: one with linear quantification (prover 1) and two variants of a prover without linear quantification (prover 2). Both variants of prover 2 are based on Hepple 1996 and Lev 2007, but one is extended with a notation for conducting multistage proving (Findlay and Haug 2022), a process that essentially allows for the grouping of meaning constructors to constrain the order of application. This is one way of accounting for restrictions on scope-taking expressions like quantifiers, embedding verbs, etc.

These provers provide separate additional functionalities for exploring the resulting Glue derivations, including reasons why a derivation might fail. Specifically, prover 1 has two functionalities. First, it allows for a depth-first search of intermediate results in a failed proof, extracting those partial solutions that would need to be combined to find a successful proof. Second, it allows the proofs to be given in natural deduction form. This is illustrated in Figure 4 based on (5).

The two variants of prover 2 also allow users to visualize a derivation. More specifically, they

[10] `ling.sprachwiss.uni-konstanz.de/pages/xle/`

[11] `https://github.com/Mmaz1988/GlueSemWorkbench_v2`

[12] For a string *a* corresponding to a function and an argument string *b*, the default procedure produces the string *a(b)*.

```
[/x e.[/y e.love(x,y)]] : (g -o (h -o f))     john : g
-----------------------------------------------------E
      [/x e.[/y e.love(x,y)]](john) : (h -o f)          mary : h
      ------------------------------------------------------------E
              [/x e.[/y e.love(x,y)]](john)(mary) : f
```

Figure 4: Natural deduction proof by GSWB, based on meaning constructors in (5)

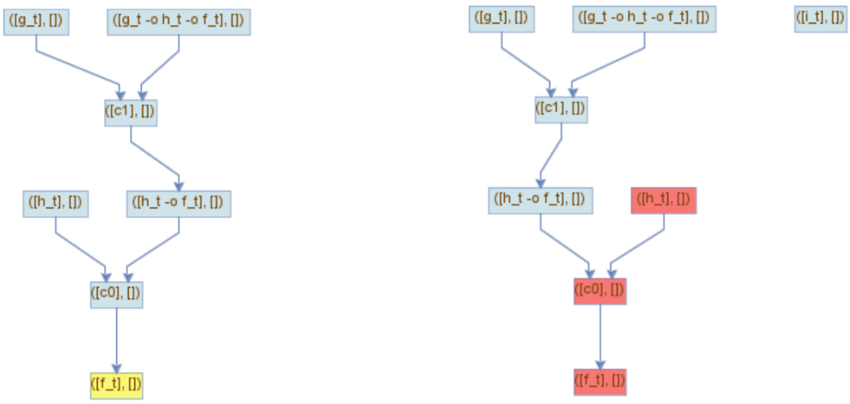

Figure 5: **Successful derivation graph for the proof in (4) and an alternative failed derivation graph**: The graph on the left presents input meaning constructors and combination steps as blue nodes and highlights the goal category in yellow. The graph on the right is based on an erroneous input that is superficially similar to (4) . Missing resources (leaves of the graph) and failed derivation steps are marked in red so as to make it easier to debug the proof. The proof fails since h is required by the verb as a resource corresponding to the object. However, in this unsuccessful proof the object was assigned the resource i, which is a dangling node since it has no consumer.

produce a derivation graph. This graph roughly corresponds to a proof tree but highlights cyclic elements in the derivation (indicating compositional ambiguities), if present (cf. Lev 2007: ch. 6). Figure 5 illustrates the visualization. (The derivation there does not have any cyclic elements.)[13]

Current and future developments of GSWB are mainly geared toward the interpretability of the output of GSWB, as illustrated in Figures 4–5, as well as the integration in broader processing pipelines. This is illustrated by reference to the next two tools, which use the capabilities presented above.

3.2 XLE+Glue

XLE+Glue has been developed as an interface between XLE and GSWB corresponding to LFG+Glue in the theoretical literature. It is integrated into the XLE user interface and can be used out of the box.

The original version[14] consists of a specification for Glue meaning constructors in terms of attribute-value matrices that can be represented as part of f-structures (Dalrymple et al. 2020).

Example (7) illustrates the encoding of the Glue meaning constructor in (6) as an AVM in an f-structure. As shown there, linear logic resources are added via the GLUE attribute, whose value is a set of semantic representations. These are described in terms of AVMs encoding their MEANING side (simply a string corresponding to the meaning) and their linear logic side. The latter uses nested expressions to reflect linear implication: ARG1 and ARG2 refer to linear logic resources (not semantic arguments) that need to be consumed to produce the resource f with type t.

(6) $\textit{love} : g_e \multimap (h_e \multimap f_t)$

(7)
$$f \left[\begin{array}{ll} \text{PRED} & \text{`LOVE<SUBJ,OBJ>'} \\ \text{SUBJ} & g\,[\,] \\ \text{OBJ} & h\,[\,] \\ \text{GLUE} & \left\{ \left[\begin{array}{ll} \text{MEANING} & \text{LOVE} \\ \text{ARG1} & \left[\begin{array}{ll} \text{RESOURCE} & g \\ \text{TYPE} & e \end{array}\right] \\ \text{ARG2} & \left[\begin{array}{ll} \text{RESOURCE} & h \\ \text{TYPE} & e \end{array}\right] \\ \text{RESOURCE} & f \\ \text{TYPE} & t \end{array}\right] \right\} \end{array}\right]$$

[13]While this example is trivial, finding errors in more complex proofs can be difficult, especially when manually working with the GSWB.

[14]https://github.com/Mmaz1988/xle-glueworkbench-interface

More recently, a version with an alternative notation for meaning constructors has been developed[15] that is closer to their representation in formal semantic theory. The alternative notation is similar to that of GSWB but uses references to f-structure nodes, as in Figure 2 on the left. This is illustrated in (8).

(8)
```
[/x_e.[/y_e.P(x,y)]]:
((^SUBJ)_e -o ((^OBJ)_e -o ^_t))
```

While the notation is different, the implementation boils down to the idea of the original XLE+Glue. However, now, when loading a grammar in XLE, meaning constructors written as in (8) are automatically translated into AVM representations by a script, making the grammars leaner. Furthermore, such meaning constructors may be easier to read than the nested templates necessary to encode meaning constructors in the original approach.

This approach is, in principle, an implementation of the co-descriptive approach to Glue since the templates are generally called from the lexicon. The XLE+Glue repository provides several sample XLE grammars containing templates that produce the corresponding meaning constructors. These grammars exhibit the various parameters along which XLE+Glue can be tweaked: it allows for exploring different meaning languages (currently, first-order logic and λ-DRT), and it enables the user to specify meaning constructors in the f-structure or in a separate semantic structure. Furthermore, although the current paper presents XLE+Glue as a venue for exploring co-descriptive approaches to Glue, it is, in fact, more flexible, since the Glue AVMs corresponding to meaning constructors need not be specified in the lexicon. They could be specified via rewrite rules or, possibly, in other ways. However, since it is the only resource in this paper making a concrete proposal for exploring semantic co-description, it is unique in this regard.

On the technical side, XLE+Glue consists of an extension to the XLE user interface and a translation component that rewrites the specified meaning constructors into a format compatible with GSWB.[16] Thus, XLE+Glue is, essentially, an interface between XLE and GSWB.

[15] https://github.com/Mmaz1988/xleplusglue

[16] The original translation component was written in Prolog. For the new system, the scripts have been moved to a Java implementation.

3.3 Linguistic Graph Expansion and Rewriting

The Linguistic Graph Expansion and Rewriting (LiGER)[17] tool allows for the specification of rules that rewrite and expand f-structure nodes, as shown in Figure 2 on the right. The system is based on graph matching techniques, but also provides tools to check for certain LFG-specific relations such as (inside-out) functional uncertainty. The graphs are described in terms of queries inspired by corpus search engines, in particular the one designed for LFG within INESS (Rosén et al. 2012; https://clarino.uib.no/iness/). Before querying, the system translates f-structures into more general graph structures. This mechanism is inspired by the original XLE transfer system (Crouch et al. 2017, Ide and Bunt 2010), but it is applicable beyond the annotations provided by the XLE. For example, it provides an interface to the Stanford Universal Dependency parser (Manning et al. 2014). Generally speaking, it is mainly geared towards the analysis of directed (acyclic) graphs that underlie many syntactic analyses.

Figure 6 illustrates normalization from syntactic representations to directed graphs. Given this kind of normalization, the system can be combined with various linguistic resources to either specify structural correspondences or expand graphs with additional information. The primary use of the system is currently the specification of semantic rules inspired by the description-by-analysis tradition in Glue (Kaplan and Wedekind 1993). It combines insights from computational approaches, e.g., Crouch 2005 and Crouch and King 2006, with more recent theoretical approaches (Andrews 2008, 2010). The former employ a destructive approach during which a given f-structure is taken as input to a set of ordered rewrite rules. These rules incrementally consume parts of the f-structure to produce semantic constraints, sometimes involving intermediate representations and access to external resources (e.g., for lexical semantics). Thus, the inverse mapping from semantics to syntax is not trivially recoverable.[18] By contrast, the theoretical approach involves working towards a structure-preserving implementation, i.e., a monotonic approach to description-by-analysis, more clearly maintaining LFG's bi-directionality. This choice is

[17] https://github.com/Mmaz1988/abstract-syntax-annotator-web

[18] See Zarrieß and Kuhn (2010) for discussion.

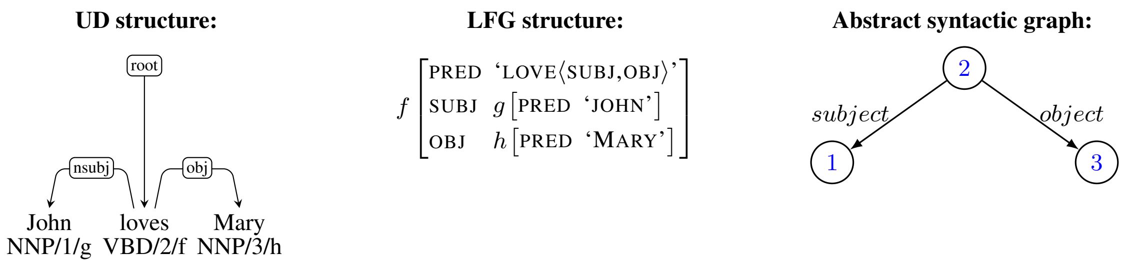

Figure 6: Parallelized syntax for: *John loves Mary*

not constrained by LiGER, but rather by how the system is used. Thus, it is well-suited to explore the notion of description-by-analysis.

LiGER is implemented in Java as an application and a web service in parallel, so it can be used in web-based applications and more traditional annotation pipelines. As indicated above, it is compatible with Universal Dependencies (as provided by Stanford CoreNLP) and XLE representations. It can also be used to call the corresponding parsers from their respective resources.

4 Use cases

At this stage of development, XLE+Glue and LiGER have not been widely used for broad coverage semantic parsing (but see Findlay et al. 2023 for a broad coverage use of the GSWB). However, they have already been employed for verification of theoretical LFG+Glue analyses (see §4.1), for a teaching grammar (see §4.2), and for research on ambiguity management (see §4.3).

4.1 Verification of theoretical analyses

The tools described above have been used to verify theoretical analyses. For example, GSWB has been employed in an investigation of scope interactions between nominal and verbal quantifiers (Zymla and Sigwarth 2019), LiGER in an analysis of Greek tense and aspect (Zymla and Fiotaki 2021), and XLE+Glue in an account of gapping (Przepiórkowski and Patejuk 2023).

In particular, Przepiórkowski and Patejuk 2023 propose a theoretical LFG+Glue analysis of gapping, as in English *Marge saw Lisa and Homer Bart*, with the second conjunct meaning 'Homer saw Bart'. The analysis crucially relies on Champollion's (2015) compositional treatment of event semantics and is relatively complex, to the extent that it is not trivial to manually verify its predictions for more complex cases, such as (9), which is expected to have the two readings in (10)–(11).

(9) Tracy introduced Lisa to Marge and Bart to Homer.

(10) $[\exists e.\, introduce(e) \wedge agent(e,t) \wedge theme(e,l) \wedge beneficiary(e,m)] \wedge [\exists e.\, introduce(e) \wedge \underline{agent(e,t)} \wedge \underline{theme(e,b)} \wedge beneficiary(e,h)]$

'Tracy introduced Lisa to Marge and Tracy introduced Bart to Homer.'

(11) $[\exists e.\, introduce(e) \wedge agent(e,t) \wedge theme(e,l) \wedge beneficiary(e,m)] \wedge [\exists e.\, introduce(e) \wedge \underline{agent(e,b)} \wedge \underline{theme(e,l)} \wedge beneficiary(e,h)]$

'Tracy introduced Lisa to Marge and Bart introduced Lisa to Homer.'

However, using XLE+Glue, the formal analysis was implemented as an XLE grammar and all reading were derived automatically. In the case of (9), they all turned out to be equivalent to (10) or (11).

4.2 Teaching grammar

A different application of the presented suite of Glue tools concerns a teaching grammar implementing analyses of some phenomena encountered in a grammar development class, especially tense and aspect.

Using GSWB and LiGER, the grammar produces DRT representations based on the Boxer tools exemplifying a Neo-Davidsonian event semantics. An example is shown in (12). There, *x1* refers to an event with two arguments, *x2* and *x3*. These are enumerated based on an argument hierarchy (Bresnan and Kanerva 1989). For the purpose of this paper, *arg1* generally refers to an agentive role, *arg2* refers to a theme/patient role, and *arg3* generally refers to a recipient/goal role.[19]

(12) Mary hugged a bear.

[19]Thus, the argument roles are comparable to those in the PropBank (Palmer et al. 2005), but they are not verb-specific.

```
 _____________
| x2 x3 x1    |
|-------------|
| bear(x2)    |
| x3 = Mary   |
| hug(x1)     |
| arg1(x1,x3) |
| arg2(x1,x2) |
|_____________|
```

Appendix B contains additional examples of DRSs produced by the grammar on the basis of meaning constructors derived by LiGER from f-structures.

For example, we have added to the grammar some basic LiGER rules for tense/aspect interpretation. In this case, the importance of LiGER lies in contextualizing tense/aspect features according to their morphosyntactic context and beyond. This is illustrated in example (13) below, where the interpretation of the embedded tense is constrained by the matrix tense. To put it concisely, the embedded tense must be evaluated relative to the matrix tense and it may only be evaluated as simultaneous or anterior to it as indicated by the two readings of (13) shown in (14)–(15). This is explained in more detail in Zymla 2017, 2018. Furthermore, this example illustrates differences between perfective (bounded) and imperfective (ongoing) grammatical aspect in the matrix clause and embedded clause respectively (see Zymla 2019 for details).

(13) Mary said that Susan was hugging a bear.

(14) Mary said: Susan is hugging a bear.

(15) Mary said: Susan was hugging a bear.

```
 ____________________________________________________
| x9 x8 x7 x6                                        |
|----------------------------------------------------|
| x9 = now                                           |
| before(x8,x9)                                      |
| x7 = Mary                                          |
| bounded(x6,x8)                                     |
| say(x6)                                            |
| arg1(x6,x7)                                        |
|        ___________________________________________ |
|       | x5 x4 x2                                  ||
|       |-------------------------------------------||
| say   | bear(x5)                                  ||
|       | x4 = Susan                                ||
|       | nonfut(x2,x6)                             ||
|       |   _______________       _______________   ||
|       | | x1            |     | x3            |   ||
|       | |---------------|     |---------------|   ||
|       | | ongoing(x1,x2)| ==> | partOf(x3,x1) |   ||
|       | |_______________|     | hug(x3)       |   ||
|       |                       | arg2(x3,x5)   |   ||
|       |                       | arg1(x3,x4)   |   ||
|       |                       |_______________|   ||
|       |___________________________________________||
|____________________________________________________|
```

As these examples show, the teaching grammar combined with LiGER allows us to capture important syntactic and semantic generalizations and, thus, explore substantial insights into the interplay between syntax and semantics. Importantly, the grammar also illustrates the distinction between co-description and description-by-analysis pointed out in the previous sections.

4.3 Exploring the cross-linguistic variability of semantic ambiguities

Moot and Retoré (2012) point out that semantic composition does not fall into one neatly categorized logic but rather moves on a spectrum of constrainedness of the underlying logic. Works like Gotham 2019, 2021, building on Barker 2022, explore subtle cross-linguistic differences in the flexibility of scope-taking semantic operators that highlight this issue. Constraints on semantic composition have been explored in the context of computational grammars and Glue semantics in Findlay and Haug 2022 and Zymla 2024 using the GSWB. This research not only improves the formal adequacy of LFG, but potentially also allows for more precise statements about the logic of composition more generally and its interaction with other modules of grammar, particularly, from a cross-linguistic perspective. Relatedly, Butt et al. (2024) present an approach to disambiguating questions via prosodic information that is computationally implemented in combination with XLE+Glue.

5 Summary

This paper has presented a suite of computational tools for Glue Semantics. It has established their relevance for exploring various important concepts, particularly how to embed a semantic component in the LFG projection architecture as implemented within XLE. In this regard, both a co-descriptive and a description-by-analysis approach have been presented, covering the two major proposals for semantic analysis in LFG.

The tools presented in this paper contribute to a growing ecosystem of LFG-based CL tools which are actively developed on multiple fronts. They receive regular updates and new features.

However, it is not only the LFG community that may benefit from these tools. Glue Semantics is compatible with various kinds of syntactic and semantic analyses. From the perspective of NLP, one of the most interesting prospects may be its compatibility with Universal Dependencies (Haug and Findlay 2023), but even beyond that, the tools presented in this paper provide an exciting avenue for research on formal semantics in computational linguistics.

References

Avery D Andrews. 2008. The role of PRED in LFG + Glue. In *Proceedings of the LFG08 Conference*, pages 47–67.

Avery D Andrews. 2010. Propositional glue and the correspondence architecture of LFG. *Linguistics and Philosophy*, 33(3):141–170.

Nicholas Asher and Hajime Wada. 1988. A computational account of syntactic, semantic and discourse principles for anaphora resolution. *Journal of Semantics*, 6(1):309–344.

Ash Asudeh. 2004. *Resumption as resource management*. Ph.D. thesis, Stanford University.

Ash Asudeh. 2006. Direct compositionality and the architecture of LFG. *Intelligent linguistic architectures: Variations on themes by Ronald M. Kaplan*, pages 363–387.

Ash Asudeh. 2022. Glue semantics. *Annual Review of Linguistics*, 8:321–341.

Ash Asudeh. 2023. Glue semantics. In (Dalrymple 2023), pages 651–697.

Ash Asudeh and Richard Crouch. 2002. Glue semantics for HPSG. In *Proceedings of the 8th international HPSG conference, Stanford, CA. CSLI Publications*.

Chris Barker. 2022. Rethinking scope islands. *Linguistic Inquiry*, 53(4):633–661.

V. Basile, J. Bos, K. Evang, and N. Venhuizen. 2012. A platform for collaborative semantic annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 92–96, Avignon, France.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing — Analyzing text with Python and the Natural Language Toolkit*. O'Reilly.

Patrick Blackburn and Johannes Bos. 2005. *Representation and inference for natural language: A first course in computational semantics*. Center for the Study of Language and Information Amsterdam.

Daniel G. Bobrow, Bob Cheslow, Cleo Condoravdi, Lauri Karttunen, Tracy Holloway King, Rowan Nairn, Valeria de Paiva, Charlotte Price, and Annie Zaenen. 2007. PARC's bridge and question answering system. In *Proceedings of the GEAF 2007 Workshop*, pages 1–22.

Johan Bos. 2008. Wide-coverage semantic analysis with Boxer. In *Semantics in text processing. step 2008 conference proceedings*, pages 277–286.

Joan Bresnan, Ash Asudeh, Ida Toivonen, and Stephen Wechsler. 2015. *Lexical-functional syntax*, volume 16. John Wiley & Sons.

Joan Bresnan and Jonni M Kanerva. 1989. Locative inversion in Chicheŵa: A case study of factorization in Grammar. *Linguistic inquiry*, pages 1–50.

Miriam Butt, Tina Bögel, Mark-Matthias Zymla, and Benazir Mumtaz. 2024. Alternative questions in Urdu: From the speech signal to semantics. In *Proceedings of the LFG'24 Conference*, Konstanz. PubliKon.

Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. 2002. The parallel grammar project. In *Proceedings of the 2002 Workshop on Grammar Engineering and Evaluation*, volume 15, pages 1–7. Association for Computational Linguistics.

Lucas Champollion. 2015. The interaction of compositional semantics and event semantics. *Linguistics and Philosophy*, 38(1):31–66.

Dick Crouch, Mary Dalrymple, Ronald M. Kaplan, Tracy Holloway King, John T. Maxwell III, and Paula Newman. 2017. *XLE Documentation*. Palo Alto Research Center.

Richard Crouch. 2005. Packed rewriting for mapping semantics to KR. In *Proceedings of the Sixth International Workshop on Computational Semantics (IWCS-6)*, pages 103–114, Tilburg.

Richard Crouch and Tracy Holloway King. 2006. Semantics via f-Structure rewriting. In *Proceedings of the LFG06 Conference*, pages 145–165, Stanford, CA. CSLI Publications.

Mary Dalrymple. 1999. *Semantics and syntax in Lexical Functional Grammar: The resource logic approach*. MIT Press.

Mary Dalrymple, editor. 2023. *Handbook of Lexical Functional Grammar*. Language Science Press, Berlin.

Mary Dalrymple, John Lamping, Fernando Pereira, and Vijay Saraswat. 1999. Quantification, anaphora, and intensionality. In Mary Dalrymple, editor, *Semantics and Syntax in Lexical Functional Grammar – The Resource Logic Approach*, pages 39–89.

Mary Dalrymple, John J. Lowe, and Louise Mycock. 2019. *The Oxford reference guide to Lexical Functional Grammar*. Oxford University Press, Oxford.

Mary Dalrymple, Agnieszka Patejuk, and Mark-Matthias Zymla. 2020. XLE+Glue – A new tool for integrating semantic analysis in XLE. In *Proceedings of the LFG'20 Conference, Australian National University*, Stanford, CA. CSLI Publications.

Jamie Findlay and Dag Haug. 2022. Managing scope ambiguities in Glue via multistage proving. In *Proceedings of the Lexical Functional Grammar Conference*, pages 144–163.

Jamie Y Findlay, Saeedeh Salimifar, Ahmet Yıldırım, and Dag TT Haug. 2023. Rule-based semantic interpretation for Universal Dependencies. In *Proceedings of the Sixth Workshop on Universal Dependencies (UDW, GURT/SyntaxFest 2023)*, pages 47–57.

Dan Flickinger, Stephan Oepen, and Emily Bender. 2017. Sustainable development and refinement of complex linguistic annotations at scale. In Nancy Ide and James Pustejovsky, editors, *Handbook of Linguistic Annotation*, pages 353–377. Springer.

Anette Frank. 1999. From parallel grammar development towards machine translation. A project overview. *Proceedings of Machine Translation Summit VII" MT in the Great Translation Era*, pages 134–142.

Anette Frank and Josef van Genabith. 2001. GlueTag: Linear logic based semantics for LTAG – and what it teaches us about LFG and LTAG. In *The Proceedings of the LFG'01 Conference*, pages 104–126, University of Hong Kong. CSLI Publications.

Jean-Yves Girard. 1987. Linear logic. *Theoretical Computer Science*, 50(1):1 – 101.

Matthew Gotham. 2018. Making logical form type-logical: Glue Semantics for minimalist syntax. *Linguistics and Philosophy 41(5)*, pages 411–556.

Matthew Gotham. 2019. Constraining scope ambiguity in LFG+Glue. In *Proceedings of the LFG'19 Conference*, pages 111–129, Stanford, CA. CSLI Publications.

Matthew Gotham. 2021. Approaches to scope islands in LFG+Glue. In *Proceedings of the LFG'21 Conference*, pages 146–166, Stanford, CA. CSLI Publications.

Matthew Gotham and Dag Trygve Truslew Haug. 2018. Glue semantics for Universal Dependencies. In *The Proceedings of the LFG'18 Conference*, pages 208–226, Stanford, CA. CSLI Publications.

Per-Kristian Halvorsen and Ronald M. Kaplan. 1988. Projections and semantic description in Lexical-Functional Grammar. In *Proceedings of the International Conference on Fifth Generation Computer Systems, FGCS 1988, Tokyo, Japan, November 28-December 2, 1988*, pages 1116–1122. OHMSHA Ltd. Tokyo and Springer-Verlag.

Dag TT Haug and Jamie Y Findlay. 2023. Formal semantics for dependency grammar. In *Proceedings of the Seventh International Conference on Dependency Linguistics (Depling, GURT/SyntaxFest 2023)*, pages 22–31.

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

Mark Hepple. 1996. A compilation-chart method for linear categorial deduction. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*, pages 537–542. Association for Computational Linguistics.

Nancy Ide and Harry Bunt. 2010. Anatomy of annotation schemes: Mapping to GrAF. In *Proceedings of the Fourth Linguistic Annotation Workshop*, pages 247–255.

Ray Jackendoff. 2010. The parallel architecture and its place in cognitive science. In Bernd Heine and Heiko Narrog, editors, *The Oxford Handbook of Linguistic Analysis*, pages 583–605. Oxford: Oxford University Press.

Hans Kamp and Uwe Reyle. 1993. *From discourse to logic: Introduction to modeltheoretic semantics of natural language, formal logic and Discourse Representation Theory*, volume 42. Springer Science & Business Media.

Ronald M Kaplan. 1995. Three seductions of computational psycholinguistics. *Formal Issues in Lexical-Functional Grammar*, 47.

Ronald M. Kaplan and Joan Bresnan. 1982. Lexical-Functional Grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173–281. MIT Press, Cambridge, MA.

Ronald M Kaplan and Jürgen Wedekind. 1993. Restriction and correspondence-based translation. In *Sixth Conference of the European Chapter of the Association for Computational Linguistics*.

Ronald M Kaplan and Jurgen Wedekind. 2019. Tractability and discontinuity. In *Proceedings of the International Lexical-Functional Grammar Conference*, pages 130–148.

Ewan Klein. 2006. Computational semantics in the Natural Language Toolkit. In *Proceedings of the Australasian Language Technology Workshop 2006*, pages 26–33.

Iddo Lev. 2007. *Packed computation of exact meaning representations*. Ph.D. thesis, Stanford University.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Moritz Meßmer and Mark-Matthias Zymla. 2018. The Glue Semantics Workbench: A Modular Toolkit for Exploring Linear Logic and Glue Semantics. In *Proceedings of the LFG'18 Conference, University of Vienna*, pages 249–263, Stanford, CA. CSLI Publications.

Richard Moot and Christian Retoré. 2012. *The logic of categorial grammars: A deductive account of natural language syntax and semantics*. Number 6850 in Lecture Notes in Computer Science. Springer, Heidelberg.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Adam Przepiórkowski and Agnieszka Patejuk. 2023. Filling gaps with Glue. In *The Proceedings of the LFG'23 Conference*, pages 223–240. PubliKon.

Victoria Rosén, Koenraad De Smedt, Paul Meurer, and Helge Dyvik. 2012. An open infrastructure for advanced treebanking. In *META-RESEARCH Workshop on Advanced Treebanking at LREC2012*, pages 22–29.

Sebastian Sulger, Miriam Butt, Tracy Holloway King, Paul Meurer, Tibor Laczkó, György Rákosi, Cheikh M Bamba Dione, Helge Dyvik, Victoria Rosén, Koenraad De Smedt, Agnieszka Patejuk, Özlem Çetinŏglu, I Wayan Arka, and Meladel Mistica. 2013. ParGramBank: The ParGram parallel treebank. In *ACL*, pages 550–560.

Jurgen Wedekind. 1988. Generation as structure driven derivation. In *Coling Budapest 1988 Volume 2: International Conference on Computational Linguistics*.

Jürgen Wedekind and Ronald M Kaplan. 2020. Tractable Lexical-Functional Grammar. *Computational Linguistics*, 46(3):515–569.

Sina Zarrieß and Jonas Kuhn. 2010. Reversing f-structure rewriting for heneration from meaning representations. In *Proceedings of the 15th International Conference on Lexical-Functional Grammar (LFG10)*, pages 479–499, Ottawa, Canada. CSLI Publications.

Mark-Matthias Zymla. 2017. Comprehensive annotation of cross-linguistic variation in the category of Tense. In *12th International Conference on Computational Semantics*.

Mark-Matthias Zymla. 2018. Annotation of the syntax/semantics interface as a bridge between deep linguistic parsing and TimeML. In *Proceedings 14th Joint ACL-ISO Workshop on Interoperable Semantic Annotation*, pages 53–59.

Mark-Matthias Zymla. 2019. Aspectual reasoning in LFG – A computational approach to grammatical and lexical aspect. In *Proceedings of the LFG'19 Conference, Australian National University*, pages 353–373, Stanford, CA. CSLI Publications.

Mark-Matthias Zymla. 2024. Ambiguity management in computational Glue semantics. In *Proceedings of the LFG'24 Conference*, pages 285–310, Konstanz, Germany. PubliKon.

Mark-Matthias Zymla and Alexandra Fiotaki. 2021. Perfective non-past in Modern Greek. In *Proceedings of the LFG'21 Conference, On-Line*, pages 332–352, Stanford, CA. CSLI Publications.

Mark-Matthias Zymla, Kascha Kruschwitz, and Paul Zodl. 2025. Semantic parsing and reasoning in LFG – the case of gradable adjectives. In *Proceedings of the BriGap-2 Workshop: Bridges and Gaps between Formal and Computational Linguistics*. To appear.

Mark-Matthias Zymla and Gloria Sigwarth. 2019. On the syntax/semantics interface in computational Glue Semantics: A case study. In *Proceedings of the LFG'19 Conference, Australian National University*, pages 374–392, Stanford, CA. CSLI Publications.

A Additional proofs

$$\frac{[X : m_e]^1 \quad \lambda x.\lambda y.\text{like}(x,y) : m_e \multimap (b_e \multimap f_t)}{\lambda y.\text{like}(X,y) : b_e \multimap f_t} \multimap_E$$

$$\frac{\lambda y.\text{like}(X,y) : b_e \multimap f_t \quad [Y : b_e]^2}{\text{like}(X,Y) : f_t} \multimap_E$$

$$\frac{\text{like}(X,Y) : f_t}{\lambda x.\text{like}(x,Y) : m_e \multimap f_t} \multimap_{I,1}$$

$$\frac{\lambda P.\forall x[\text{monkey}(x) \rightarrow P(x)] : (m_e \multimap f_t) \multimap f_t \quad \lambda x.\text{like}(x,Y) : m_e \multimap f_t}{\forall x[\text{monkey}(x) \rightarrow \text{like}(x,Y)] : f_t} \multimap_E$$

$$\frac{\forall x[\text{monkey}(x) \rightarrow \text{like}(x,Y)] : f_t}{\lambda y.\forall x[\text{monkey}(x) \rightarrow \text{like}(x,y)] : b_e \multimap f_t} \multimap_{I,2}$$

$$\frac{\lambda y.\forall x[\text{monkey}(x) \rightarrow \text{like}(x,y)] : b_e \multimap f_t \quad \lambda Q.\exists y[\text{banana}(y) \wedge Q(y)] : (b_e \multimap f_t) \multimap f_t}{\exists y[\text{banana}(y) \wedge \forall x[\text{monkey}(x) \rightarrow \text{like}(x,y)]] : f_t} \multimap_E$$

Figure 7: Glue proof: *Every monkey likes a banana* inverse scope

B Worked out examples

(16) Mary hugged a bear.

```
 _____________
| x2  x3  x1  |
|-------------|
| bear(x2)    |
| x3 = Mary   |
| hug(x1)     |
| arg1(x1,x3) |
| arg2(x1,x2) |
|_____________|
```

Produced meaning constructors:

```
{
lam(V,lam(X,lam(E,merge(app(V,E),drs([],[rel(arg2,E,X)]))))) :
    ((6_v -o 6_t) -o (4_e -o (6_v -o 6_t))) || noscope
lam(X,drs([],[eq(X,'Mary')])) : (8_e -o 8_t)
lam(X,drs([],[pred(bear,X)])) : (4_e -o 4_t)
lam(V,lam(X,lam(E,merge(app(V,E),drs([],[rel(arg1,E,X)]))))) :
    ((6_v -o 6_t) -o (8_e -o (6_v -o 6_t))) || noscope
lam(P,lam(Q,merge(drs([X],[]),merge(app(P,X),app(Q,X))))) :
    ((8_e -o 8_t) -o ((8_e -o 5_t) -o 5_t)) || noscope
lam(V,drs([],[pred(hug,V)])) : (6_v -o 6_t)
lam(P,lam(Q,merge(drs([X],[]),merge(app(P,X),app(Q,X))))) :
    ((4_e -o 4_t) -o ((4_e -o 5_t) -o 5_t))
lam(V,merge(drs([E],[]),app(V,E))) : ((6_v -o 6_t) -o 5_t)
}
```

F-structure:

```
"Mary hugged a bear"

      ⎡PRED    'hug<[1:Mary], [26:bear]>'                               ⎤
      ⎢       1⎡PRED 'Mary'                                            ⎤⎥
      ⎢SUBJ  66⎢CASE nom, GEND fem, NTYPE name, NUM sg, PERS 3         ⎥⎥
      ⎢      68⎣                                                       ⎦⎥
      ⎢      35⎡PRED 'bear'                                          ⎤  ⎥
      ⎢     102⎢                                                     ⎥  ⎥
      ⎢OBJ   26⎢SPEC [DET [PRED 'a']]                                ⎥  ⎥
    14⎢      34⎢CASE acc, DEF -, NTYPE count, NUM sg, PERS 3         ⎥  ⎥
    78⎢     104⎣                                                     ⎦  ⎥
   108⎢                                                                 ⎥
   122⎢TNS-ASP [MOOD indicative, PERF -_, PROG -_, TENSE past]          ⎥
   124⎣PASSIVE -                                                        ⎦
```

(17) Mary was hugged by a bear.

```
 _____________
| x3  x2  x1   |
|--------------|
| bear(x3)     |
| x2 = Mary    |
| hug(x1)      |
| arg1(x1,x3)  |
| arg2(x1,x2)  |
|______________|
```

Produced meaning constructors:

```
{
lam(V,merge(drs([E],[]),app(V,E))) : ((6_v -o 6_t) -o 5_t)
lam(V,lam(X,lam(E,merge(app(V,E),drs([],[rel(arg2,E,X)]))))) :
    ((6_v -o 6_t) -o (8_e -o (6_v -o 6_t))) || noscope
lam(X,drs([],[eq(X,'Mary')])) : (8_e -o 8_t)
lam(X,drs([],[pred(bear,X)])) : (4_e -o 4_t)
lam(V,lam(X,lam(E,merge(app(V,E),drs([],[rel(arg1,E,X)]))))) :
    ((6_v -o 6_t) -o (4_e -o (6_v -o 6_t))) || noscope
lam(P,lam(Q,merge(drs([X],[]),merge(app(P,X),app(Q,X))))) :
    ((8_e -o 8_t) -o ((8_e -o 5_t) -o 5_t)) || noscope
lam(V,drs([],[pred(hug,V)])) : (6_v -o 6_t)
lam(P,lam(Q,merge(drs([X],[]),merge(app(P,X),app(Q,X))))) :
    ((4_e -o 4_t) -o ((4_e -o 5_t) -o 5_t))
}
```

F-structure:

```
"Mary was hugged by a bear"

      PRED        'hug<[38:bear], [1:Mary]>'
                1 PRED 'Mary'
      SUBJ     87 CASE nom, GEND fem, NTYPE name, NUM sg, PERS 3
               89
               56 PRED 'bear'
              132
               47 SPEC [DET [PRED 'a']]
               55 DEF -, NTYPE count, NUM sg, PERS 3, PFORM by, PTYPE nosem
   26 OBL-AG  134
  101          38
   14          46
   15         138
  140
  150 TNS-ASP     [PERF -_, PROG -_, TENSE past]
  152 PARTICIPLE past, PASSIVE +
```

(18) Susan was given the bear by Mary.

```
 _____________
| x2  x3  x4  x1 |
|-------------|
| bear(x2)      |
| x3 = Mary     |
| x4 = Susan    |
| give(x1)      |
| arg3(x1,x4)   |
| arg1(x1,x3)   |
| arg2(x1,x2)   |
|_____________|
```

Produced meaning constructors:

```
{
lam(V,merge(drs([E],[]),app(V,E))) : ((4_v -o 4_t) -o 3_t)
lam(P,lam(Q,merge(drs([X],[]),merge(app(P,X),app(Q,X))))) :
    ((6_e -o 6_t) -o ((6_e -o 3_t) -o 3_t)) || noscope
lam(V,lam(X,lam(E,merge(app(V,E),drs([],[rel(arg3,E,X)]))))) :
    ((4_v -o 4_t) -o (8_e -o (4_v -o 4_t))) || noscope
lam(P,lam(Q,merge(drs([X],[]),merge(app(P,X),app(Q,X))))) :
    ((2_e -o 2_t) -o ((2_e -o 3_t) -o 3_t)) || noscope
lam(X,drs([],[pred(bear,X)])) : (2_e -o 2_t)
lam(V,lam(X,lam(E,merge(app(V,E),drs([],[rel(arg2,E,X)]))))) :
    ((4_v -o 4_t) -o (2_e -o (4_v -o 4_t))) || noscope
lam(X,drs([],[eq(X,'Susan')])) : (8_e -o 8_t)
lam(V,lam(X,lam(E,merge(app(V,E),drs([],[rel(arg1,E,X)]))))) :
    ((4_v -o 4_t) -o (6_e -o (4_v -o 4_t))) || noscope
lam(P,lam(Q,merge(drs([X],[]),merge(app(P,X),app(Q,X))))) :
    ((8_e -o 8_t) -o ((8_e -o 3_t) -o 3_t)) || noscope
lam(V,drs([],[pred(give,V)])) : (4_v -o 4_t)
lam(X,drs([],[eq(X,'Mary')])) : (6_e -o 6_t)
}
```

F-structure:

```
"Susan was given the bear by Mary"

    [PRED      'give<[87:Mary], [53:bear], [1:Susan]>'
    [         1[PRED 'Susan'
    [SUBJ   125[CASE nom, GEND fem, NTYPE name, NUM sg, PERS 3]
    [       127[
    [        72[PRED 'bear'
    [       163[DEF +, NTYPE count, NUM sg, PERS 3]
    [OBJ2    53[
    [        71[
    [       165[
    [        96[PRED 'Mary'
    [       177[GEND fem, NTYPE name, NUM sg, PERS 3, PFORM by, PTYPE nosem]
  26[       179[
 139[OBL-AG  87[
  14[        95[
  15[       183[
 188[
 202[TNS-ASP   [TENSE past]
 204[DATIVE-SHIFT +, PARTICIPLE past, PASSIVE +
```

(19) Mary hugged herself.

```
 _____________
| x2  x3  x1   |
|-------------|
| hug(x3)      |
| arg2(x3,x2)  |
| arg1(x3,x1)  |
| female(x2)   |
| x1 = x2      |
| x1 = Mary    |
|_____________|
```

Produced meaning constructors:

```
{
lam(X,drs([],[eq(X,'Mary')])) : (6_e -o 6_t)
lam(A,alfa(B,refl,pred(female,B),merge(app(A,C),drs([C],
    [pred(female,C),eq(B,C)])))) : ((2_e -o 3_t) -o 3_t)
lam(V,lam(X,lam(E,merge(app(V,E),drs([],[rel(arg1,E,X)]))))) :
    ((4_v -o 4_t) -o (6_e -o (4_v -o 4_t))) || noscope
lam(P,lam(Q,merge(drs([X],[]),merge(app(P,X),app(Q,X))))) :
    ((6_e -o 6_t) -o ((6_e -o 3_t) -o 3_t))
lam(V,drs([],[pred(hug,V)])) : (4_v -o 4_t)
lam(V,merge(drs([E],[]),app(V,E))) : ((4_v -o 4_t) -o 3_t)
lam(V,lam(X,lam(E,merge(app(V,E),drs([],[rel(arg2,E,X)]))))) :
    ((4_v -o 4_t) -o (2_e -o (4_v -o 4_t))) || noscope
}
```

F-structure:

```
"Mary hugged herself"

    [PRED    'hug<[1:Mary], [23:herself]>'                                      ]
    [          1[PRED 'Mary'                                              ]     ]
    [SUBJ     61[CASE nom, GEND fem, NTYPE name, NUM sg, PERS 3           ]     ]
    [         63[                                                         ]     ]
    [         23[PRED 'herself'                                                 ]
  14[OBJ      24[CASE acc, NTYPE pron, NUM sg, PERS 3, PRON-TYPE pers           ]
  73[         86[                                                               ]
  88[                                                                           ]
 102[TNS-ASP [MOOD indicative, PERF -_, PROG -_, TENSE past]                    ]
 105[PASSIVE -                                                                  ]
```

(20) Mary tried to hug a bear.

```
 _____________________
| x3                  |
|---------------------|
| x3 = Mary           |
|      _____________  |
|     | x2  x1      | |
|     |-------------| |
| try | bear(x2)    | |
|     | hug(x1)     | |
|     | arg1(x1,x3) | |
|     | arg2(x1,x2) | |
|     |_____________| |
|_____________________|
```

Produced meaning constructors:

```
{
lam(V,lam(X,lam(E,merge(app(V,E),drs([],[rel(arg2,E,X)]))))) :
    ((11_v -o 11_t) -o (9_e -o (11_v -o 11_t))) || noscope
lam(X,drs([],[pred(bear,X)])) : (9_e -o 9_t)
lam(X,drs([],[eq(X,'Mary')])) : (2_e -o 2_t)
lam(V,lam(X,lam(E,merge(app(V,E),drs([],[rel(arg1,E,X)]))))) : (
    (11_v -o 11_t) -o (2_e -o (11_v -o 11_t))) || noscope
lam(P,lam(Q,merge(drs([X],[]),merge(app(P,X),app(Q,X))))) :
    ((2_e -o 2_t) -o ((2_e -o 3_t) -o 3_t)) || noscope
lam(V,drs([],[pred(hug,V)])) : (11_v -o 11_t)
lam(X,lam(P,drs([],[try(app(P,X))]))) : (2_e -o ((2_e -o 10_t) -o 3_t))
lam(P,lam(Q,merge(drs([X],[]),merge(app(P,X),app(Q,X))))) :
    ((9_e -o 9_t) -o ((9_e -o 10_t) -o 10_t))
lam(V,merge(drs([E],[]),app(V,E))) : ((11_v -o 11_t) -o 10_t)
}
```

F-structure:

```
"Mary tried to hug a bear"

       [PRED     'try<[1:Mary], [29:hug]>'                                        ]
       |        1[PRED 'Mary'                                      ]              |
       |SUBJ   95|CASE nom, GEND fem, NTYPE name, NUM sg, PERS 3   |              |
       |       97[                                                 ]              |
       |         [PRED   'hug<[1:Mary], [55:bear]>'                              ]|
       |         |SUBJ   [1:Mary]                                                ||
       |      39 |      64[PRED 'bear'                                          ]||
       |     124 |     148|                                                     |||
       |XCOMP 154|OBJ  55 |SPEC [DET [PRED 'a']]                                |||
       |      29 |      63|CASE acc, DEF -, NTYPE count, NUM sg, PERS 3         |||
    14 |      37 |     150[                                                     ]||
   107 |     168[PASSIVE -, VFORM inf                                            ]|
   170 |                                                                          |
   173 |TNS-ASP  [MOOD indicative, PERF -_, PROG -_, TENSE past]                  |
   175 [VFORM    inf                                                              ]
```

(21) Mary saw the bear with the telescope

```
 ______________                 ______________
| x2 x3 x4 x1  |               | x3 x2 x4 x1  |
|--------------|               |--------------|
| bear(x2)     |               | bear(x3)     |
| x3 = Mary    |               | x2 = Mary    |
| telescope(x4)|               | telescope(x4)|
| with(x1,x4)  |               | with(x3,x4)  |
| see(x1)      |               | see(x1)      |
| arg1(x1,x3)  |               | arg2(x1,x3)  |
| arg2(x1,x2)  |               | arg1(x1,x2)  |
|______________|               |______________|
```

Produced meaning constructors:

```
{
lam(X,drs([],[pred(telescope,X)])) : (4_e -o 4_t)
lam(V,merge(drs([E],[]),app(V,E))) : ((6_v -o 6_t) -o 5_t)
lam(V,lam(X,lam(E,merge(app(V,E),drs([],[rel(arg2,E,X)]))))) :
    ((6_v -o 6_t) -o (9_e -o (6_v -o 6_t))) || noscope
lam(P,lam(Q,merge(drs([X],[]),merge(app(P,X),app(Q,X))))) :
    ((4_e -o 4_t) -o ((4_e -o 5_t) -o 5_t)) || noscope
lam(X,drs([],[eq(X,'Mary')])) : (11_e -o 11_t)
lam(V,lam(X,lam(E,merge(app(V,E),drs([],[rel(arg1,E,X)]))))) :
    ((6_v -o 6_t) -o (11_e -o (6_v -o 6_t))) || noscope
lam(P,lam(Q,merge(drs([X],[]),merge(app(P,X),app(Q,X))))) :
    ((11_e -o 11_t) -o ((11_e -o 5_t) -o 5_t)) || noscope
lam(X,drs([],[pred(bear,X)])) : (9_e -o 9_t)
lam(P,lam(Q,merge(drs([X],[]),merge(app(P,X),app(Q,X))))) :
    ((9_e -o 9_t) -o ((9_e -o 5_t) -o 5_t)) || noscope
lam(Y,lam(X,drs([],[rel(with,X,Y)]))) :
    (4_e -o (6_v -o 7_t))
lam(U,lam(V,lam(E,merge(drs([],[]),merge(app(U,E),app(V,E)))))) :
    ((6_v -o 7_t) -o ((6_v -o 6_t) -o (6_v -o 6_t)))
lam(V,drs([],[pred(see,V)])) : (6_v -o 6_t)
}

{
lam(X,drs([],[pred(telescope,X)])) : (5_e -o 5_t)
lam(V,merge(drs([E],[]),app(V,E))) : ((7_v -o 7_t) -o 6_t)
lam(U,lam(V,lam(E,merge(drs([],[]),merge(app(U,E),app(V,E)))))) :
    ((9_e -o 8_t) -o ((9_e -o 6_t) -o (9_e -o 6_t))) || noscope
lam(V,lam(X,lam(E,merge(app(V,E),drs([],[rel(arg2,E,X)]))))) :
    ((7_v -o 7_t) -o (9_e -o (7_v -o 7_t))) || noscope
lam(P,lam(Q,merge(drs([X],[]),merge(app(P,X),app(Q,X))))) :
    ((5_e -o 5_t) -o ((5_e -o 6_t) -o 6_t)) || noscope
lam(X,drs([],[eq(X,'Mary')])) : (11_e -o 11_t)
lam(V,lam(X,lam(E,merge(app(V,E),drs([],[rel(arg1,E,X)]))))) :
    ((7_v -o 7_t) -o (11_e -o (7_v -o 7_t))) || noscope
lam(P,lam(Q,merge(drs([X],[]),merge(app(P,X),app(Q,X))))) :
    ((11_e -o 11_t) -o ((11_e -o 6_t) -o 6_t)) || noscope
lam(X,drs([],[pred(bear,X)])) : (9_e -o 9_t)
lam(P,lam(Q,merge(drs([X],[]),merge(app(P,X),app(Q,X))))) :
    ((9_e -o 9_t) -o ((9_e -o 6_t) -o 6_t)) || noscope
lam(Y,lam(X,drs([],[rel(with,X,Y)]))) : (5_e -o (9_e -o 8_t))
lam(V,drs([],[pred(see,V)])) : (7_v -o 7_t)
}
```

F-structures:

```
"Mary saw the bear with the telescope"

    [PRED    'see<[1:Mary], [37:bear]>'
            1[PRED 'Mary'                                          ]
     SUBJ 136[CASE nom, GEND fem, NTYPE name, NUM sg, PERS 3       ]
          138[                                                     ]
           56[PRED 'bear'                                          ]
          172[CASE acc, DEF +, NTYPE count, NUM sg, PERS 3         ]
     OBJ   37[                                                     ]
           55[                                                     ]
          174[                                                     ]
              {   [PRED   'with<[86:telescope]>'
                         105[PRED 'telescope'                                 ]
                         193[CASE acc, DEF +, NTYPE count, NUM sg, PERS 3     ]
     ADJUNCT {    OBJ     86[                                                 ]
               71        104[                                                 ]
  14           85        195[                                                 ]
 148          199 PTYPE  sem                                                  ]}
 204
 218 TNS-ASP [MOOD indicative, PERF -_, PROG -_, TENSE past]
 220 PASSIVE -                                                               ]
```

```
"Mary saw the bear with the telescope"

    [PRED    'see<[1:Mary], [37:bear]>'
            1[PRED 'Mary'                                          ]
     SUBJ 136[CASE nom, GEND fem, NTYPE name, NUM sg, PERS 3       ]
          138[                                                     ]
             [PRED    'bear'
                      {   [PRED   'with<[86:telescope]>'
                                 105[PRED 'telescope'                              ]
                                 193[CASE acc, DEF +, NTYPE count, NUM sg, PERS 3  ]
     OBJ      ADJUNCT {   OBJ     86[                                              ]
           56          71        104[                                              ]
          172          85        195[                                              ]
           37         199 PTYPE  sem                                               ]}
  14       55
 148      201 CASE acc, DEF +, NTYPE count, NUM sg, PERS 3                         ]
 204
 218 TNS-ASP [MOOD indicative, PERF -_, PROG -_, TENSE past]
 220 PASSIVE -                                                                    ]
```

(22) Mary said that Susan was hugging a bear.

```
 ________________________________________________
| x9 x8 x7 x6                                    |
|------------------------------------------------|
| x9 = now                                       |
| before(x8,x9)                                  |
| x7 = Mary                                      |
| bounded(x6,x8)                                 |
| say(x6)                                        |
| arg1(x6,x7)                                    |
|      __________________________________________ |
|     | x5 x4 x2                              | |
|     |-----------------------------------------| |
| say | bear(x5)                              | |
|     | x4 = Susan                            | |
|     | nonfut(x2,x6)                         | |
|     |  _______________      ______________   | |
|     | | x1            |    | x3           |  | |
|     | |---------------|    |--------------|  | |
|     | | ongoing(x1,x2) | ==> | partOf(x3,x1) |  | |
|     | |_______________|    | hug(x3)      |  | |
|     |                      | arg2(x3,x5)  |  | |
|     |                      | arg1(x3,x4)  |  | |
|     |                      |______________|  | |
|     |_________________________________________| |
|________________________________________________|
```

Produced meaning constructors:

```
{
//Liger
lam(S,lam(T,drs([],[rel(ongoing,T,S)]))) : (207_s -o (209_s -o 205_t))
lam(S,lam(T,drs([],[rel(bounded,T,S)]))) : (208_s -o (210_s -o 206_t))
lam(M,lam(P,lam(S,drs([],[imp(merge(drs([Z],[]),app(app(M,S),Z)),app(P,Z))])))) :
    ((207_s -o (209_s -o 205_t)) -o ((10_s -o 6_t) -o (11_s -o 6_t)))
lam(M,lam(P,lam(S,merge(drs([Z],[]),merge(app(app(M,S),Z),app(P,Z)))))) :
    ((208_s -o (210_s -o 206_t)) -o ((19_s -o 18_t) -o (8_s -o 18_t)))
lam(T,lam(T2,drs([],[rel(before,T,T2)]))) : (8_s -o (9_s -o 8_t))
lam(T,lam(T2,drs([],[rel(nonfut,T,T2)]))) : (11_s -o (12_s -o 11_t))
lam(T,lam(P,lam(S,merge(drs([R],[]),merge(app(app(T,R),S),app(P,R)))))) :
    ((11_s -o (12_s -o 11_t)) -o ((11_s -o 6_t) -o (12_s -o 6_t)))
lam(T,lam(P,lam(S,merge(drs([R],[]),merge(app(app(T,R),S),app(P,R)))))) :
    ((8_s -o (9_s -o 8_t)) -o ((8_s -o 18_t) -o (9_s -o 18_t)))
//Grammar
lam(X,drs([],[eq(X,'Susan')])) : (14_e -o 14_t)
lam(X,drs([],[eq(X,'Mary')])) : (17_e -o 17_t)
lam(P,lam(Q,merge(drs([X],[]),merge(app(P,X),app(Q,X))))) :
    ((5_e -o 5_t) -o ((5_e -o 6_t) -o 6_t))
lam(X,drs([],[pred(bear,X)])) : (5_e -o 5_t)
lam(P,lam(Q,merge(drs([X],[]),merge(app(P,X),app(Q,X))))) :
    ((17_e -o 17_t) -o ((17_e -o 18_t) -o 18_t)) || noscope
lam(V,lam(X,lam(E,merge(app(V,E),drs([],[rel(arg2,E,X)]))))) :
    ((7_v -o 7_t) -o (5_e -o (7_v -o 7_t))) || noscope
lam(P,merge(drs([T],[eq(T,now)]),app(P,T))) :
    ((9_s -o 18_t) -o 18_t) || noscope
lam(P,lam(Q,merge(drs([X],[]),merge(app(P,X),app(Q,X))))) :
    ((14_e -o 14_t) -o ((14_e -o 6_t) -o 6_t)) || noscope
lam(V,lam(X,lam(E,merge(app(V,E),drs([],[rel(arg1,E,X)]))))) :
    ((7_v -o 7_t) -o (14_e -o (7_v -o 7_t))) || noscope
lam(V,drs([],[pred(hug,V)])) : (7_v -o 7_t)
lam(P,lam(X,lam(S,merge(drs([],[pred(say,S),rel(arg1,S,X)]),drs([],[say(app(P,S))])))))) :
    ((12_s -o 6_t) -o (17_e -o (19_s -o 18_t)))
lam(V,lam(S,merge(drs([E],[rel(partOf,E,S)]),app(V,E)))) :
    ((7_v -o 7_t) -o (10_s -o 6_t))
}
```

F-structure:

```
"Mary said that Susan hugged a bear"

    [PRED    'say<[1:Mary], [23:hug]>'
    [        1[PRED 'Mary'
    [SUBJ  123[CASE nom, GEND fem, NTYPE name, NUM sg, PERS 3]
    [      125[
    [         [PRED    'hug<[58:Susan], [83:bear]>'
    [         [      58[PRED 'Susan'
    [         [SUBJ 155[CASE nom, GEND fem, NTYPE name, NUM sg, PERS 3]
    [         [     157[
    [         [      92[PRED 'bear'
    [COMP  71 [     191[
    [     167 [OBJ   83[SPEC [DET [PRED 'a']]
    [     197 [      91[CASE acc, DEF -, NTYPE count, NUM sg, PERS 3]
    [     211 [     193[
    [      23 [
 14 [      57 [TNS-ASP [MOOD indicative, PERF -_, PROG -_, TENSE past]
135 [     213 [COMP-FORM that, PASSIVE -
215 [
218 [TNS-ASP [MOOD indicative, PERF -_, PROG -_, TENSE past]
220 [ROOT    +
```